Entity Matching using Deep Neural Networks: From Discriminative Pre-trained Language Models to Generative Large Language Models

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Ralph Peeters
aus Bensheim

Mannheim, 2025

Dekan:	Professor Dr. Claus Hertling, Universität Mannheim
Referent:	Professor Dr. Christian Bizer, Universität Mannheim
Korreferent:	Professor Dr. Heiko Paulheim, Universität Mannheim
Korreferent:	Professor Dr. Felix Naumann, Universität Potsdam

Tag der mündlichen Prüfung: 10. März 2025

Acknowledgements

Looking back on the long and challenging journey toward completing this dissertation, I recognize that this work is not mine alone, but rather the result of invaluable support and encouragement from many people who stood by me and share credit for its completion.

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Dr. Christian Bizer, for his continuous support, guidance, and mentorship throughout my doctoral studies. His insightful discussions, valuable advice, and patience during challenging phases of my research have been instrumental, and I am genuinely grateful for such dedicated mentorship. My sincere thanks also extend to the other professors of the Data and Web Science Group: Prof. Dr. Heiko Paulheim, Prof. Dr. Simone Ponzetto, and Prof. Dr. Rainer Gemulla, whose stimulating discussions and valuable input have significantly enriched my work over the years. Additionally, special thanks go to Prof. Dr. Goran Glavaš, who ignited my interest in deep learning during my master's studies, thank you for the collaboration and support. I also extend my appreciation to Prof. Dr. Felix Naumann for serving on my defense committee.

My heartfelt thanks go to my colleagues and friends: Dr. Anna Primpeli, for your continuous support, insightful discussions before and during this journey, being the best office mate I could have wished for, and for all the wonderful experiences beyond our research. Dr. Sven Hertling, Dr. Alexander Diete, and Manuel Fink, for warmly welcoming me, teaching me the ropes, and sharing many enjoyable discussions, banter, and memorable moments. Alexander Brinkmann, Dr. Chia-Chien Hung, Dr. Jovita Lukasik, and Keti Korini, thank you for enriching my experience with stimulating conversations, shared struggles, and mutual encouragement. Alex, special thanks to you for being an exceptional office mate and friend, I truly feel fortunate to have shared these years with you. Additionally, a heartfelt thank you goes to Aaron Steiner, whose consistently positive nature brightened the final stages of my doctoral work. Last but certainly not least, thank you to my dear friend Dr. Christian Czech for picking me up when I stumbled just before the finishing line.

Finally, but most profoundly, I thank my family, my parents Heinz and Dorothea, and my sisters Sylvia and Sonja, for their unwavering support and unconditional love throughout this entire process. Your constant belief in me provided essential strength during difficult times, significantly contributing to the successful completion of this dissertation. Sonja, you took the brunt of it, thank you for always being there for me. Special thanks also to Yuki Long for our wonderful shared experiences and for broadening my horizons during my final year, making the concluding stretch so much easier.

Thank you all from the bottom of my heart.

Abstract

Entity matching is the task of identifying records that refer to the same entity across different datasets. It is a critical step in the data integration process. Supervised entity matching methods typically frame the problem as a binary classification task between record pairs. These methods require labeled record pairs, consisting of matches and non-matches, for training. Key challenges in entity matching include high heterogeneity among records referring to the same entity, scarcity of training data, and the continuous emergence of unseen entities in real-world applications.

This thesis introduces two novel benchmarks for product matching, created using semantically annotated product identifiers on the Web as distant supervision. These benchmarks, sourced from thousands of e-shops, are among the largest and most diverse publicly available product matching datasets. They enable a fine-grained evaluation of entity matching methods across different entity matching challenges.

The thesis presents two new neural approaches for entity matching based on pre-trained language models, which achieve state-of-the-art results on multiple benchmarks. Unlike existing methods, both approaches exploit entity group information alongside binary matching labels during training. The first method, JointBERT, employs a dual-objective fine-tuning strategy. The second method, R-SupCon, uses supervised contrastive learning and establishes new state-of-the-art results on multiple benchmarks, proving particularly effective on smaller training sets. In addition, the thesis explores the usefulness of multilingual Transformers for improving product matching performance in low-resource languages.

This work further investigates generative large language models for entity matching, comparing them with pre-trained language models. The investigations include an analysis of prompting techniques, such as zero-shot inference, in-context learning, and rule-based prompting, as well as fine-tuning for entity matching. The results highlight the potential of large language models to match or exceed the performance of fine-tuned pre-trained language models, while requiring no or minimal amounts of training data. Additionally, the experiments demonstrate better generalization to unseen entities compared to pre-trained language models.

The thesis also examines the explainability of matching decisions, introducing two methods for aggregating local explanations into global insights. The first method, based on LIME explanations, is broadly applicable to matching classifiers. The second method uses large language models to produce structured explanations that can be automatically parsed and aggregated. Finally, the thesis introduces a method for automating error analysis using large language models. This approach allows for the automatic generation of error classes, which can help data engineers in the process of improving entity matching pipelines.

Zusammenfassung

Entity Matching ist die Aufgabe, Datensätze zu identifizieren, die sich auf dieselbe Entität in unterschiedlichen Datenquellen beziehen. Es handelt sich um einen kritischen Schritt im Datenintegrationsprozess. Entity Matching Methoden, die auf überwachten maschinellen Lernverfahren basieren, betrachten die Aufgabe als binäre Klassifikation von Datensatzpaaren. Diese Methoden benötigen gelabelte Datensatzpaare, sowohl Matches als auch Non-Matches, für das Training. Zu den wichtigsten Herausforderungen zählen die hohe Heterogenität von Datensätzen, die sich auf dieselbe Entität beziehen, die Knappheit an Trainingsdaten sowie das kontinuierliche Auftreten unbekannter Entitäten in Anwendungen in der Praxis.

Diese Arbeit führt zwei neue Benchmarks für das Produkt-Matching ein, die mittels semantisch annotierter Produktkennungen aus dem Web als distant supervision erstellt wurden. Diese Benchmarks, die aus tausenden von E-Shops stammen, gehören zu den größten und vielfältigsten öffentlich verfügbaren Datensätzen für Produkt-Matching und ermöglichen eine feingranulare Evaluation von Entity Matching-Systemen für verschiedene Entity Matching Herausforderungen.

Die Arbeit präsentiert zwei neue neuronale Ansätze für Entity Matching auf Basis vortrainierter Sprachmodelle, die auf mehreren Benchmarks Werte auf dem Stand der Technik erreichen. Im Gegensatz zu bestehenden Methoden nutzen beide Ansätze während des Trainings neben den binären Matching-Labels auch Entitätsgruppierungen. Die erste Methode, JointBERT, verwendet eine Fine-tuning-Strategie mit zwei Zielen. Die zweite Methode, R-SupCon, nutzt überwachtes Contrastive Learning und erzielt Werte auf dem Stand der Technik auf mehreren Benchmarks, insbesondere bei kleineren Trainingssätzen. Zusätzlich untersucht die Arbeit die Nützlichkeit von multilingualen Transformern zur Verbesserung der Produkt-Matching Leistung in ressourcenarmen Sprachen.

Es werden große generative Sprachmodelle für Entity Matching untersucht und mit vortrainierten Sprachmodellen verglichen. Prompting-Techniken, wie Zero-Shot-Inferenz, In-Context Learning und regelbasiertes Prompting, sowie das Finetuning für Entity Matching werden analysiert. Die Ergebnisse zeigen, dass große Sprachmodelle ohne, oder mit nur minimalen Mengen von Trainingsdaten, die Leistung von vortrainierten Sprachmodellen erreichen oder übertreffen können und insbesondere bessere Generalisierung für unbekannte Entitäten ermöglichen.

Schließlich wird die Erklärbarkeit von Matching-Entscheidungen untersucht, wobei zwei Methoden zur Aggregation lokaler Erklärungen eingeführt werden, um globale Einblicke zu gewinnen. Die erste Methode basiert auf LIME-Erklärungen und ist universell anwendbar, während die zweite große Sprachmodelle nutzt, um strukturierte und automatisch parsbare Erklärungen zu erzeugen. Abschließend wird eine Methode vorgestellt, die es ermöglicht, die Fehleranalyse mittels großer Sprachmodelle zu automatisieren und Fehlerklassen automatisch zu generieren, was Dateningenieuren dabei helfen kann Entity Matching-Prozesse zu verbessern.

Contents

Chapter 1

Introduction

1.1 Motivation

Entity matching, the process of identifying records that refer to the same entity in different datasets, is a critical task in data integration [Christen, 2012b,Christophides et al., 2020, Barlaug and Gulla, 2021]. In an era where data grows exponentially, organizations increasingly depend on accurate data integration to drive their analytics, decision-making, and strategic initiatives.

In today's connected world, entity matching plays a crucial rule in many areas. In healthcare, entity matching can be used to consolidate patient records from various hospitals and clinics to provide a unified view, improving patient care and clinical research. In e-commerce, entity matching can be used to merge product offers from multiple platforms to deliver a holistic overview of offers to customers. Companies must integrate data between various departments and locations [Doan et al., 2012] for various applications such as analytics or customer relationship management. In the area of finance, the integration of financial data is crucial for supporting investment decisions.

Furthermore, entity matching is vital in government operations, where it can help ensure consistency and accuracy in public record management, such as voting registries and tax records. In academia and research, entity matching facilitates the integration of datasets from different studies or experiments, enabling meta-analyses and increased data utility for scientific exploration. This application is particularly relevant in fields such as genomics [Gomez-Cabrero et al., 2014], where data from multiple studies can be combined to improve the understanding of genetic patterns and influences.

Despite advances in machine learning and natural language processing, which have significantly influenced current entity matching methods [Christophides et al., 2020, Barlaug and Gulla, 2021], entity matching remains a challenging problem due to the diversity, volume, and variability of datasets from different sources. Figure 1.1 shows an example of the entity matching task and its result for multiple datasets referencing persons. The following paragraphs explain the challenges and characteristics of the entity matching task along this example.

The entity matching process [Christen, 2012b, Christophides et al., 2020] involves finding all pair-wise correspondences (matches) between the records of all datasets. After discovering the correspondences, they are used to group records into clusters of records with each cluster containing all records referencing the same entity (right side of Figure 1.1).

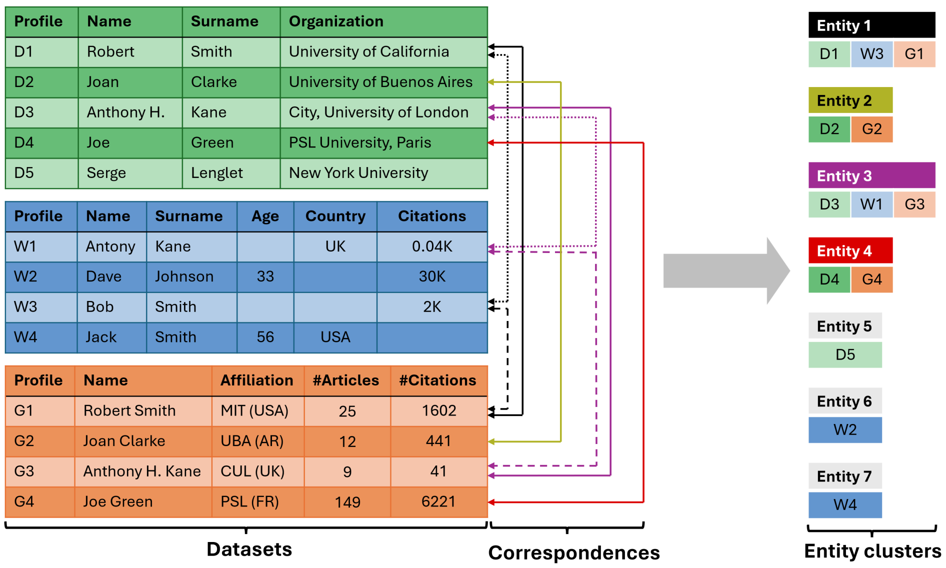


Figure 1.1: An example of entity matching between three datasets. Identifying all pairs of matching records results in a set of correspondences. These correspondences are subsequently grouped into clusters that contain all records describing the same entity. Adapted from [Papadakis et al., 2021].

Surface Form Heterogeneity: One of the main challenges in finding correspondences between datasets is the heterogeneity among the surface forms of attribute values. For example, the person *Anthony H. Kane* appears with the middle initial in the top dataset, but without it and with a spelling error (*Antony*) in the middle dataset, complicating the identification of a match between these records. Abbreviations are another source of surface form heterogeneity. The attributes *organization* and *affiliation* in the top and bottom datasets exemplify this form of heterogeneity. Furthermore, attributes can be represented using different data types, such as numerals for the age or citation counts, which can increase ambiguity due to different units of measurement. An example is the *citations* attribute in the middle and bottom dataset.

Schema Heterogeneity: This type of heterogeneity refers to all differences in the schemata of the datasets. Useful information for finding correspondences may be represented in a compound, below first normal form attribute instead of multiple attributes across different datasets. Examples are the *name* in the bottom dataset versus *name* and *surname* in the other datasets, as well as the *affiliation* in the bottom dataset that includes additional information about the country, which is found in the separate attribute *country* in the middle dataset.

Missing Values: Missing values in the dataset attributes (middle dataset) can lead to additional ambiguity, given that these attributes are relevant for finding correspondences. The *citations* and *country* attributes in the middle dataset have missing values which complicates finding correspondences with the bottom dataset that contains this information as part of the *affiliation* and *#citations* attributes.

Value Conflicts: Conflicting values for attributes of records referencing the same entity are another source of heterogeneity that complicate finding correspondences. Sources for this type of heterogeneity can be errors made when values are entered into a dataset, e.g. due to human error, a wrong birthdate may be entered when a new person is added to a dataset. Furthermore, the process of extracting data from databases can be error-prone and lead to value conflicts. Similarly, errors in the schema matching step, which is performed before entity matching in the data integration pipeline (see Section 2.2), can lead to value conflicts. Value conflicts can further arise because of attribute values changing over time due to the intrinsic nature of certain attributes, such as the natural process of aging or the number of citations increasing over the years. The records for the same entity *Robert/Bob Smith* in the middle and bottom datasets show different amounts of citations, whereas the affiliation in the top and bottom datasets also differs for this person as he may have changed his place of work.

The arrows in Figure 1.1 show the complete pair-wise correspondences between the records, which are the result of performing the entity matching task as a pair-wise matching task. After finding these correspondences, the final entity matching step uses the pair-wise correspondences to group records into clusters referencing the same entity. The right part of Figure 1.1 shows the final result of this clustering, with one cluster for each entity containing all records that refer to this entity. The entity matching process is described in more detail in Section 2.3.

Another important concept in entity matching is the concept of *head* and *tail entities*. Entity 1 (Robert Smith) and Entity 3 (Anthony H. Kane) are described by three records each in the final clusters in Figure 1.1. The clusters for the other persons contain only two or a single record. Entities described by many records across datasets are termed *head entities*, while those covered by few datasets are termed *tail entities*. For instance, common products are head entities, whereas niche items are tail entities. The head and tail distribution impacts the entity matching task, especially when applying supervised learning methods [Peeters and Bizer, 2021], as the amount of available training examples is often more limited for tail entities, resulting in less diversity in representations, which can lead to difficulty learning the relevant patterns for the correct matching decision.

Traditional methods for entity matching [Christen, 2012b, Konda et al., 2016], while effective if datasets are structured, struggle to perform effectively when applied to datasets characterized by textuality, heterogeneity, and ambiguity [Mudgal et al., 2018]. The introduction of pre-trained language models (PLMs), such as BERT [Devlin et al., 2019] and RoBERTa [Liu et al., 2019b], has provided new avenues for tackling these challenges. Their subsequent adoption by the data integration community has shown remarkable improvements, specifically for textual matching tasks [Li et al., 2020]. However, these models require thousands to tens of thousands of task-specific training data pairs [Li et al., 2020], which are often not available. Furthermore, they struggle with generalizing to entities unseen during training, which is a common scenario in dynamic and evolving application domains [Peeters et al., 2024b].

The advent of large language models (LLMs), such as the GPT and Llama series, has significantly advanced computational natural language understanding by demonstrating strong zero-shot and few-shot performance [Brown et al., 2020, Zhao et al., 2023]. Early studies of these models for entity matching [Narayan et al., 2022, Peeters et al., 2025, Zhang et al., 2023] show that when adequately prompted or fine-tuned, they can potentially offer robust solutions for entity matching without the extensive need for labeled data.

Entity matching presents challenges such as high heterogeneity among records referencing the same entity, and the continuous emergence of unseen entities (e.g., new products). Addressing these challenges requires publicly available benchmarks that explicitly model these difficulties and enable fine-grained evaluation and comparison of matching methods. The manual collection and annotation of record pairs for training and evaluating entity matching systems is costly and time-intensive [Gokhale et al., 2014]. Current publicly available benchmarks for entity matching do not fulfill the stated requirements as many are small, exhibit limited diversity, and none explicitly model the mentioned challenges [Primpeli and Bizer, 2020, Peeters et al., 2024b].

This thesis contributes to closing this research gap by introducing two novel benchmarks for product matching. The benchmarks are created using semantically annotated product identifiers on the Web as distant supervision to automatically collect pairs of records and associated labels without requiring large-scale manual annotation. Both benchmarks offer diverse data from thousands of e-shops, making them two of the largest and most diverse publicly available product matching benchmarks [Peeters et al., 2024b]. Furthermore, both benchmarks allow for evaluating entity matching systems along explicitly modeled matching challenges.

The thesis introduces two PLM-based methods for neural entity matching that improve on previous state-of-the-art methods by using entity group information during the training procedure in addition to pair-wise matching and non-matching entity descriptions. The proposed and existing methods are compared on the presented benchmarks and a range of widely used entity matching benchmarks, highlighting the strengths and weaknesses of each method. The presented methods outperform existing methods in matching scenarios where entities have been seen during training. The thesis further evaluates multilingual Transformers for product matching in low-resource scenarios.

This thesis investigates using generative large language models for entity matching. These investigations include comparing the effectiveness of prompting techniques, including zero-shot inference, in-context learning, rule-based prompting, and fine-tuning. These experiments highlight the potential of LLMs to match or exceed the performance of fine-tuned PLMs while requiring no or only minimal amounts of training data. Additionally, the experiments demonstrate better generalization to unseen entities compared to PLMs.

While neural methods for entity matching have advanced the field [Barlaug and Gulla, 2021], record pair-specific matching decisions of these models cannot be easily explained due to the black-box nature of neural networks [Sun et al., 2021]. As a result, the explainability of neural methods has become a research focus in the field of entity matching [Di Cicco et al., 2019, Baraldi et al., 2023, Paganelli et al., 2022]. Many current methods for explaining entity matching model decisions focus on explanations for decisions for single record pairs [Di Cicco et al., 2019], termed local explanations. Other methods focus on explaining matching decisions of specific types of models, like PLMs [Baraldi et al., 2023, Paganelli et al., 2022].

This thesis contributes two novel methods for aggregating local explanations into global insights to provide a comprehensive understanding of model behavior. The first method, based on LIME [Ribeiro et al., 2016] explanations, is broadly applicable to matching classifiers. The second method uses large language models to produce structured explanations that can be automatically parsed and aggregated.

Error analysis is an integral part of the process of improving entity matching pipelines. This task consists of collecting single erroneous matching decisions and aggregating them into meaningful error classes that allow insights into the situations in which a matching model fails. Due to the creativity required to create meaningful error classes from matching errors, humans have traditionally performed this task. The last part of this thesis presents a method for using LLMs to generate meaningful error classes and classify erroneous matching decisions automatically.

1.2 Contributions

This section outlines the contributions of this thesis to the state-of-the-art of entity matching in data integration. The thesis contributes novel entity matching benchmarks based on Schema.org annotations found on the public Web, novel methods for supervised entity matching, and novel methods for explaining entity matching decisions and analyzing matching errors. Specifically, the contributions of this thesis are:

1. **Novel Benchmarks for Entity Matching:** This thesis contributes two novel benchmarks, created using product identifiers found in Schema.org annotations as distant supervision, and their evaluation to the field of benchmarks for entity matching in data integration. Many widely used publicly available benchmarks for entity matching exhibit a range of shortcomings, including small size and lack of diversity regarding matching record pairs [Peeters et al., 2024b]. Furthermore, they do not allow an evaluation of specific matching dimensions like the number of corner cases or unseen entities during inference. The first benchmark contains over thirty thousand matching pairs, making it the largest entity matching benchmark for the product domain at the time of its creation. It contains a test set with one specific matching challenge assigned to each pair, allowing for a more fine-grained evaluation, which is unavailable in previous benchmarks. The second benchmark, as an evolution of these ideas and with a similar size to the first, allows for the detailed and separate evaluation of the dimensions number of corner cases, number of unseen entities in the test set, and development set size. It further allows studying the interactions between these three dimensions. Both benchmarks expand the current set of publicly available benchmarks as two of the largest available benchmarks while supporting the investigation of specific matching challenges.

2. **Cross-Lingual Fine-Tuning for Entity Matching:** The Transformer architecture has been successfully used in multiple entity matching methods [Barlaug and Gulla, 2021] focusing on mono-lingual matching. In contrast, its application on cross-lingual entity matching tasks has not been investigated. This thesis contributes a comparison of mono- and multi-lingual Transformer models on a cross-lingual product matching task to the field of supervised entity matching. The investigated task consists of matching English and German product offers for phones from the Web. The results demonstrate that supplementing German training data with English training data improves product matching performance in German.

3. **JointBERT - Method for Dual-Objective Fine-Tuning for Entity Matching:** The task of entity matching using supervised machine learning methods has traditionally been approached by training a binary classifier for comparing two records [Fellegi and Sunter, 1969, Christen, 2012b, Christophides et al., 2020]. Given a pair of records, the learned model decides whether they match. Datasets used for training these models can contain multiple records describing the same entity. These groups of records are known in some cases. For example, GTIN or MPN numbers for products, DUNS numbers for companies, or ISBN numbers for books can be used to identify such groups. Existing matching methods do not directly exploit the knowledge that multiple records refer to the same entity and only learn from pairs of records. This thesis contributes the JointBERT method to the field of supervised entity matching, which uses these groups as part of a second objective

during model training. JointBERT is a multi-objective method as, in addition to the binary matching objective, it is trained with the secondary objective of predicting the individual entity groups to which both records in a pair belong. The evaluation of the method shows that this dual-objective architecture improves matching results in entity matching scenarios where multiple records for the same entity are available.

4. **R-SupCon - Supervised Contrastive Learning for Entity Matching:** The Transformer architecture is often used in existing methods for entity matching. However, it is almost exclusively trained using a cross-entropy loss [Li et al., 2020, Yao et al., 2022]. This thesis contributes the method R-SupCon to the field of supervised entity matching, which applies supervised contrastive learning during the training procedure. The evaluation shows that a supervised contrastive loss improves results over previous methods that use a cross-entropy loss if only a small amount of training pairs are available and the task does not contain unseen entities.

5. **Prompt Engineering and Fine-tuning for Generative Large Language Models for Entity Matching:** Existing state-of-the-art entity matching systems [Barlaug and Gulla, 2021] are often based on smaller PLMs like BERT and RoBERTa. This thesis contributes a comparison of the effectiveness of various prompting techniques using LLMs to the field of entity matching. These include zero-shot and few-shot prompting as well as prompting with handwritten and automatically learned rules. The thesis further contributes an evaluation of fine-tuning LLMs for entity matching. The experiments compare hosted closed-source and local open-source LLMs and show they can reach or exceed the matching performance of fine-tuned PLMs with no or only a limited amount of training pairs. Additionally, the experiments show better generalization to unseen entities compared to PLMs.

6. **Novel Methods for Explaining Entity Matching Decisions and Analyzing Matching Errors:** This thesis contributes two novel methods for aggregating instance-level explanations into global-level model insights to the field of explaining entity matching model decisions. The first method is based on generating global insights by aggregating importance/relevancy weights via domain-specific word classes and is applicable to any matching classifier that can be explained with the LIME [Ribeiro et al., 2016] method. The second method presented in this thesis leverages an LLM to generate structured explanations for its decisions that can be automatically parsed and aggregated into global-level insights as a ranked list of attributes with average importance scores that the LLM bases its decisions on. The thesis further contributes a method to automatically generate error classes from LLM-generated structured explanations of erroneous matching decisions. Beforehand, human annotators had to perform this creative task manually.

1.3 Outline

This section provides an overview of the structure of this thesis. The thesis consists of five parts. The first part introduces the foundational concepts of data integration, entity matching, and semantic annotations on the Web. The second part describes how Schema.org annotations can be used as distant supervision to automatically create labeled pairs for entity matching and introduces two benchmarks derived from Schema.org annotations. The third part describes novel neural methods and their evaluation on the entity matching task. The fourth part presents methods for explaining model decisions in entity matching and automating error analysis.

PART I: Foundations

The first part of the thesis introduces foundational concepts of data integration, entity matching, and semantic annotations on the Web. These concepts form the basis for all created benchmarks, methods, and experiments described in the following chapters.

Chapter 2: Data Integration and Entity Matching: This chapter presents the data integration process with a focus on the entity matching step, which is the main topic of this thesis. The sections define basic concepts like *entity* and *entity matching*, introduce the entity matching workflow, similarity metrics, evaluation metrics, and the representation of records as embeddings using Word2Vec and language models, specifically PLMs, and LLMs. The last section of the chapter defines additional terminology used throughout the thesis.

Chapter 3: Semantic Annotation on the Web: This chapter presents the concept of semantic annotations on the Web. The sections discuss the history of semantic annotations and present various formats for annotating on the Web. Schema.org is introduced, which is the main vocabulary for semantic annotations used in the following sections of this thesis. Finally, two table corpora created from Schema.org annotations during the work on this thesis are introduced, which can be used as a source of domain data and for the creation of benchmarks, amongst other things.

PART II: The Semantic Web as Supervision for Product Matching

The second part of the thesis describes a process for using Schema.org annotations in the Common Crawl to automatically cluster product offers referring to the same product by leveraging annotated product identifiers on web pages. It further introduces the WDC LSPM benchmark created from this clustering as well as an experimental evaluation of the benchmark for training product matchers using Schema.org data. These experiments include an analysis of performance on unseen entities, additional fine-tuning for such entities, the impact of noise in the training data, and a large-scale fine-tuning experiment for the BERT Transformer using millions of product offer pairs automatically generated from Schema.org annotations. In the second chapter of this part, the WDC Products benchmark is introduced, an evolution of the WDC LSPM benchmark sourced from more recent data. It includes multiple variants for investigating three matching challenges.

Chapter 4: Semantic Annotations as Training Data for Product Matching: This chapter presents a process for using product identifiers in Schema.org annotations found in the Common Crawl to automatically cluster product offers referring to the same product. This clustering is subsequently used to create the WDC LSPM benchmark, which was the largest and most diverse product matching benchmark compared to publicly available benchmarks at the time of its creation The benchmark addresses the need for larger and more diverse entity matching benchmarks. The WDC LSPM benchmark is subsequently used to train and evaluate product matching systems, showing the usefulness of automatically created training data from Schema.org annotations. These experiments include the maintenance of matchers for new unseen products and the impact of increasing label noise in the training data. Finally, millions of the automatically generated training pairs from the clustering are used to intermediately train the BERT Transformer, further increasing its performance on the WDC LSPM benchmark, which underlines the usefulness of the clustering for automatically creating large amounts of training data.

Chapter 5: WDC Products: A Multi-Dimensional Entity Matching Benchmark: This chapter introduces the WDC Products benchmark, created from a clustering of Schema.org annotated products in a more recent version of the Common Crawl compared to WDC LSPM. The benchmark has a similar size compared to WDC LSPM making it one of the largest publicly available benchmarks. WDC Products originates from more sources than LSPM, resulting in higher diversity than any publicly available product matching benchmark. WDC Products further provides multiple variants of the benchmark that allow a separate investigation along three matching challenges and how these challenges interact with each other. These challenges are (1) the amount of corner cases, (2) the number of unseen entities in the test set, and (3) the amount of available development set pairs. This novel structure of the benchmark makes it a unique contribution to the field of entity matching, as no existing publicly available benchmarks allow for this kind of evaluation along multiple matching challenges.

PART III: Deep Neural Networks for Entity Matching

The third part of the thesis presents methods for entity matching based on neural networks. The chapters of this part discuss cross-language learning for entity matching and introduce the PLM-based dual-objective method JointBERT and the contrastively trained R-SupCon method. Both methods use entity group information in combination with pair-wise labels during training, resulting in state-of-the-art performance on multiple entity matching benchmarks. The final chapter of this part investigates prompting and fine-tuning of LLMs. It shows that this model type can achieve or exceed the performance of PLMs with no or only a limited amount of training examples and further achieves a better matching performance on unseen entities than PLMs.

Chapter 6: Cross-Language Learning for Entity Matching: This chapter presents an evaluation of mono- and multi-lingual PLMs like mBERT[1] and XLM-RoBERTa [Conneau et al., 2020] on cross-lingual product matching. The experiments show that multi-lingual Transformer models can achieve high matching performance on a low-resource language, German in these experiments, by supplementing the available German training data with additional training data from the high-resource language English compared to training with only the available German training data.

Chapter 7: JointBERT: Dual-Objective Fine-tuning for Entity Matching: This chapter presents JointBERT, a dual-objective fine-tuning method for entity matching. Compared to existing single objective methods for entity matching based on PLMs, JointBERT is trained with a secondary objective predicting the entity groups both records in a training pair refer to in addition to the binary match/non-match objective. The experimental evaluation shows that JointBERT outperforms single-objective methods in entity matching scenarios where multiple records are available per entity in the training set.

Chapter 8: R-SupCon: Supervised Contrastive Learning for Entity Matching: This chapter presents R-SupCon, a method for entity matching based on PLMs that is trained with a supervised contrastive loss. The method uses available entity group information or derives it from available labeled training pairs in combination with a source-aware sampling strategy to avoid label noise. In the first step, the PLM RoBERTa is pre-trained with this method. In the second step, the trained model is frozen, and a classification layer is fine-tuned for the final classification of record pairs as match or non-match. The experiments show that this method improves the state-of-the-art on a set of entity matching benchmarks compared to previous methods in seen scenarios and is especially effective if training data is limited.

Chapter 9: Prompt Engineering and Fine-tuning of Large Language Models for Entity Matching: This chapter compares the effectiveness of prompting techniques for entity matching using LLMs, including zero-shot and few-shot

[1] https://github.com/google-research/bert/blob/master/multilingual.md

prompting, prompting with written rules, and automatically learned rules. The chapter further presents results for fine-tuning LLMs for entity matching. The experimental results show that LLMs can achieve the same performance as PLMs with no or only a limited amount of training pairs and also achieve a better matching performance on unseen entities than PLMs.

PART IV: Explainability and Error Analysis for Entity Matching

The fourth and final part of the thesis presents methods for explaining model decisions of entity matching models and a method for automating error analysis with LLM-based matchers. The former methods are based on aggregating explanations for matching decisions for single record pairs into global insights. The first presented method aggregates domain-specific word classes in combination with importance/relevancy weights generated by a method based on LIME [Ribeiro et al., 2016] explanations. The second presented method leverages an LLM to create structured explanations for its decision, which can subsequently be automatically parsed and aggregated to global insights. The final chapter of this part introduces a method for using LLMs to automatically generate meaningful error classes for error analysis based on erroneous matching decisions and associated structured explanations, which can help data engineers improve entity matching pipelines.

Chapter 10: Explaining Model Decisions for Entity Matching: This chapter presents two methods for aggregating explanations of model decisions for single record pairs into global insights. The first method is based on local Mojito [Di Cicco et al., 2019] explanations, an adaptation of LIME [Ribeiro et al., 2016] for entity matching. The explanations assign an importance score to each token in the attributes of a record pair. The proposed method categorizes words into a set of domain-specific word classes, which subsequently allows for the aggregation of importance scores to a global level, showing the aptitude of PLM-based entity matching models for recognizing and weighting important word classes like model numbers compared to a previous RNN-based method. The second method leverages an LLM to generate structured explanations for its decisions, containing attributes and associated importance scores. This structure can subsequently be exploited to automatically parse the explanations and aggregate importance scores globally by attributes similar to the first method.

Chapter 11: Automatic Error Analysis for Entity Matching: This chapter presents a method for using an LLM to automatically perform the creative task of creating a set of error classes for error analysis. The LLM uses the matching errors made together with the associated structured explanations to create these classes. Human annotators had to perform this task beforehand. The experiments in this chapter further show that the LLM can sort the errors into the correct error classes with high accuracy, making this method useful for supporting data engineers working on entity matching pipelines.

Chapter 12: Conclusion: This chapter summarizes the contributions of this thesis and discusses the research impact of this thesis by discussing research publications that have used the benchmarks and methods described in this thesis. The last section of this chapter presents directions for future work.

1.4 Published Work

The research work presented in this thesis has previously been presented at international conferences and workshops. Each chapter of this thesis explicitly mentions the relevant publications in its introduction. The following list presents an overview of all publications that have contributed to this thesis.

- **The Semantic Web as Supervision for Product Matching**
  - **[Primpeli et al., 2019]** Primpeli, A., Peeters, R., and Bizer, C. (2019). The WDC Training Dataset and Gold Standard for Large-Scale Product Matching. In *Workshop on e-Commerce and NLP, Companion Proceedings of the 2019 World Wide Web Conference*, pages 381–386.
  - **[Bizer et al., 2019]** Bizer, C., Primpeli, A., and Peeters, R. (2019). Using the Semantic Web as a Source of Training Data. *Datenbank-Spektrum*, 19(2):127–135.
  - **[Peeters et al., 2020a]** Peeters, R., Bizer, C., and Glavaš, G. (2020a). Intermediate Training of BERT for Product Matching. In *CEUR Workshop Proceedings*, volume 2726, pages 2–6.
  - **[Peeters et al., 2020b]** Peeters, R., Primpeli, A., Wichtlhuber, B., and Bizer, C. (2020b). Using schema.org Annotations for Training and Maintaining Product Matchers. In *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics*, pages 195–204.
  - **[Peeters et al., 2024a]** Peeters, R., Brinkmann, A., and Bizer, C. (2024a). The Web Data Commons Schema.org Table Corpora. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1079–1082.
  - **[Peeters et al., 2024b]** Peeters, R., Der, R. C., and Bizer, C. (2024b). WDC Products: A Multi-Dimensional Entity Matching Benchmark. In *Proceedings of the 27th International Conference on Extending Database Technology*, pages 22–33.
- **Methods based on Pre-Trained Language Models for Entity Matching**
  - **[Peeters and Bizer, 2021]** Peeters, R. and Bizer, C. (2021). Dual-Objective Fine-tuning of BERT for Entity Matching. *Proceedings of the VLDB Endowment*, 14(10):1913–1921.

- **[Peeters and Bizer, 2022a]** Peeters, R. and Bizer, C. (2022a). Cross-Language Learning for Product Matching. In *Companion Proceedings of the Web Conference 2022*, pages 236–238.
- **[Peeters and Bizer, 2022b]** Peeters, R. and Bizer, C. (2022b). Supervised Contrastive Learning for Product Matching. In *Companion Proceedings of the Web Conference 2022*, pages 248–251.

• **Prompt Engineering and Fine-tuning of Generative Large Language Models for Entity Matching**

- **[Peeters and Bizer, 2023]** Peeters, R. and Bizer, C. (2023). Using ChatGPT for Entity Matching. In *Proceedings of the 27th Conference on Advances in Databases and Information Systems*, pages 221–230.
- **[Peeters et al., 2025]** Peeters, R., Steiner, A., and Bizer, C. (2025). Entity Matching using Large Language Models. In *Proceedings of the 28th International Conference on Extending Database Technology*.

Part I

Foundations

Chapter 2

Data Integration and Entity Matching

2.1 Background

Information systems are ubiquitous in today's connected and fast-moving society. They are essential for the creation of business value, the management of data in the health sector, and the support of research at scientific institutions and universities. As a result, large amounts of data are generated worldwide. In 2020, approximately 64 zettabytes of data were created, consumed, and stored, with projections for 2025 reaching 175 zettabytes of global data usage [Reinsel et al., 2018]. Although only 2% of the 64 zettabytes used in 2020 were stored and retained in 2021, the resulting amount of stored data is still greater than 1 zettabyte.

In addition to handling these large amounts of data, different types of heterogeneity between datasets from different data sources complicate data linkage between sources [Elmagarmid et al., 2007, Christen, 2012b]. The latter is a highly relevant process for any organization trying to extract value from data. For example, larger companies maintain various databases across different departments. A holistic view of the data requires the integration of the different datasets into one view or dataset for analytics purposes and to support strategic or operational decisions. This process, which consists of multiple steps, is called data integration.

This chapter introduces the basic concepts and methods of the data integration process with a special focus on the entity matching step, which is the main topic of this thesis' contributions. Section 2.2 introduces the full data integration process for consolidating multiple datasets into one single integrated view or dataset. Section 2.3 focuses specifically on the entity matching step of the data integration process and introduces relevant definitions and evaluation metrics. Section 2.4 clarifies additional terminology used throughout the thesis.

2.2 The Data Integration Process

The data integration process aims to consolidate data from multiple sources into a single integrated, complete, and concise view or dataset [Bleiholder and Naumann, 2009]. For a dataset to be complete per this definition, all the entities described in the original datasets must also be contained in the integrated dataset. For a dataset to be concise, only one record in the integrated dataset should describe each entity. Formally, the terms record and entity are defined as [Christophides et al., 2015, Papadakis et al., 2021]:

Definition 1 ***(Record)*** *A record r_i of a dataset D is defined as $r_i = \{(a_{i_j}, v_{i_j}) : a_{i_j} \in N, v_{i_j} \in V\}$, with N the set of attribute names, and V the set of attribute values in D with $r_i \in D$.*

Definition 2 ***(Entity)*** *An entity e_n is represented by a set of records, i.e. $e_n = \{r_1, ..., r_n\}, with\{r_1, ..., r_n\}$ describing the same object.*

This thesis also uses the term *entity description* which is not formally defined in the literature. For the purposes of this thesis, an *entity description* refers to a record that is characterized by high textuality in the values of its attributes. In entity matching use-cases, for example, when matching products across e-shops, the entities are often described by highly textual attributes instead of a more structured format as found in relational databases.

The term dataset used in this section refers to any kind of collection of records referencing entities that are described by the same set of attributes. These sets of attributes do not necessarily overlap between different datasets, nor do values referencing the same attribute of the same entity across datasets have to be exactly the same. These circumstances show the main challenge of the data integration process, which is resolving different types of heterogeneity between datasets from different data sources along the steps of the process. These kinds of heterogeneity can be divided into three types [Özsu and Valduriez, 2020]:

- **Syntactical Heterogeneity** is the different representation of data in datasets from different sources. For example, data may use different encodings, such as ASCII or Unicode, across different data formats, such as CSV or XML.
- **Structural Heterogeneity** refers to differences in how the data is represented across data sources with respect to the schemata of the underlying data structures. For example, the name of a person can be described by a single attribute *name* in one dataset while it is represented by *first name* and *last name* in another.
- **Semantic Heterogeneity** refers to all differences concerning the meaning of the data and schema values. Examples are synonyms, i.e., values that have the same meaning but different surface forms, or homonyms, referencing values that have the same surface form but actually different meanings in other datasets.

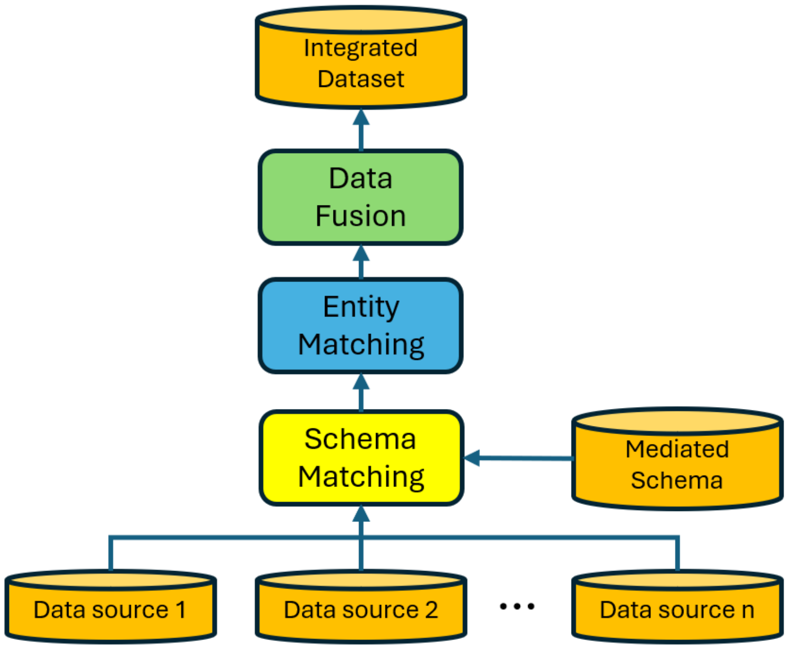


Figure 2.1: The data integration process, adapted from [Lehmberg, 2019].

The data integration process aims to bridge all these types of heterogeneity along its steps to create a concise and complete unified representation of the data. Figure 2.1 gives an overview of the data integration process and its steps. The first two steps of the process, schema matching and entity matching, are matching steps that involve finding overlaps across data sources.

Schema Matching

The first step in the process, commonly referred to as *Schema Matching* or *Schema Alignment* [Rahm and Bernstein, 2001], deals with structural heterogeneity between datasets from different data sources. Specifically, the outcome of the schema matching task is a unified schema for the integrated view or dataset that represents the attributes found in the different data sources. To achieve this, attribute columns from all schemata need to be normalized and matched across the datasets, so that each dataset from each source is consequently represented by the same schema. Methods and research directions for the schema matching problem are discussed in various surveys [Rahm and Bernstein, 2001, Bernstein et al., 2011, Gal, 2011]. Formally, schema matching between data sources is defined as [Rahm and Bernstein, 2001]:

Definition 3 ***(Schema Matching)*** *Given multiple schemata S_A, S_B, ... S_N, for multiple data sources A, B, ... N, find the correspondences $C_S \subseteq A \times B \times ... \times N$ that map each attribute in each data source to its semantically equivalent attribute in the other data sources.*

Entity Matching

The second matching step, after the alignment of the schemata, is *entity matching*, often also referred to as *entity resolution*, *record linkage* or *identity resolution* [Fellegi and Sunter, 1969, Christen, 2012b, Christophides et al., 2015, Barlaug and Gulla, 2021]. Christophides et al. [Christophides et al., 2015] make a distinction between the terms entity resolution and entity matching, the latter being a substep of the former in their definition. Entity resolution encompasses the entity matching step and an optional upstream blocking or filtering step (more details are given in Section 2.3) in their definition. For the purposes of this thesis, both terms are used interchangeably and refer to the full process that includes both the blocking and matching steps.

The goal of the entity matching step is to match all records across datasets that reference the same entity and deals with resolving heterogeneity across records from different sources to achieve that goal. Entity matching is the main focus of this thesis, as all contributions relate to this step of the data integration pipeline. Entity matching methods, research directions, and the relevant definitions are described in further detail in Section 2.3. Entity matching between multiple sources can be defined as [Christophides et al., 2015]:

Definition 4 ***(Entity Matching)*** *Given multiple data sources A, B, ..., N, let $E_{Desc} = \{r_1, ..., r_x\}$ be the set of entity descriptions contained in all sources. Further, let $M : E_{Desc} \times E_{Desc} \to \{true, false\}$ be a binary matching function. An entity matching of E_{Desc} aims to create a partitioning $P = \{p_1, ..., p_n\}$ of E_{Desc} where all $p \in P$ refer to distinct real-world objects and each p contains all entity descriptions referring to p from E_{Desc}.*

Entity matching is often preceded by a blocking or filtering step [Papadakis et al., 2020], which reduces the necessary amount of comparisons between pairs of records as this number grows quadratically with the number of records in each dataset.

Both matching steps, schema matching, and entity matching, in a pair-wise formulation, require defining a binary decision function M that maps the comparison of column/record pairs to a binary output space. Let p be a pair of records or a pair of attributes from different datasets, then for both tasks it is necessary to find a decision function M such that:

$$M(p) \to \begin{cases} 1 & \text{if matching} \\ 0 & \text{otherwise} \end{cases}$$

Defining the decision function M is difficult due to heterogeneity in the values and attributes of the datasets as presented above. It can further happen that the semantics are not fully captured by schema and records [Miller et al., 2000] but are implicit. The next section discusses these sources of difficulty for finding the decision function together with examples for the entity matching task in detail.

Data Fusion

The final step of the data integration process is data fusion [Naumann et al., 2006, Bleiholder and Naumann, 2009, Dong and Naumann, 2009]. This step requires aligned schemata and matching record correspondences between all datasets to create an integrated, complete, and concise view or dataset. The main challenge of the data fusion step is fusing the records referencing the same entity across all datasets. As a result, data fusion also faces the problem of semantically heterogeneous values similar to the previous steps in the data integration pipeline. Although it is decided which values for which attribute refer to the same entity, it is still to be decided which of the different representations of these equivalent values should be selected for the fused records in the fused dataset in a step called *conflict handling*.

The conflict handling step of data fusion can be approached using three different strategies according to Bleiholder and Naumann [Bleiholder and Naumann, 2009]:

- **Conflict Ignorance:** Data fusion with conflict ignorance leaves the decision of what values to use or discard to the application or user. Source datasets can be merged using a union operator for all values without discarding anything in this strategy. Assuming a perfect matching result in the schema matching and entity matching steps, the resulting view or dataset of this type of conflict handling will be complete but not necessarily concise.
- **Conflict Avoidance:** Data fusion with conflict avoidance employs functions that make a decision for which value to keep and which to discard without comparing the actual values. As a result, when employing this strategy, it is unclear if there is a conflict between values in the first place, as this does not play a role for the decision. An example of conflict avoidance are functions that always choose the value from a preferred source.
- **Conflict Resolution:** Data fusion with conflict resolution considers the actual values of the records to be fused and employs a variety of conflict resolution functions in case all values of matching records are not equal. These are deciding functions like choice by majority vote or mediating functions like taking the mean value of all conflicting values for one record.

After data fusion, the original datasets are represented by a single view or dataset. Assuming perfect results in the matching and fusion steps, this unified representation is concise and complete, although this is rarely achievable in real-world scenarios due to errors along the integration pipeline. Such errors can further propagate along the pipeline, making each subsequent step more difficult. For example, wrong correspondences generated during the entity matching step can lead to erroneous conflict resolution if values from actual non-matching records are fused.

This thesis focuses on benchmarks, methods and evaluations specifically for the entity matching step of the data integration pipeline. The following section introduces entity matching in more detail, presenting the process, similarity metrics, neural language model representations and relevant evaluation metrics.

2.3 Entity Matching

The goal of the entity matching step of the data integration pipeline is to find all entity descriptions referring to the same real-world entity in two data sources, or across many sources. Different entity matching settings are defined as follows in the literature [Christophides et al., 2020]:

- **Dirty EM:** Refers to the deduplication of a dataset from a single source that contains duplicates.
- **Clean-Clean EM:** Refers to the setting of matching records between two datasets from two sources that are themselves duplicate-free.
- **Multi-Source EM:** In this setting, datasets from more than two sources need to be matched.

Although these definitions encompass a range of possible matching scenarios, they do not cover all settings. For example, *Dirty-Clean* and *Dirty-Dirty* scenarios with exactly two datasets are not covered by these definitions but are found in some of the widely used benchmark datasets for evaluating entity matching (see Chapter 8 for examples). Furthermore, multi-source entity matching can be considered in the *dirty* and *clean* scenarios similar to the two-source definitions. This thesis redefines these entity matching scenarios to encompass a wider range of real-word scenarios:

- **Dirty EM:** Refers to the setting of matching records in a dataset from a single data source that contains duplicates.
- **Clean/Dirty Two-source EM:** Refers to the setting of matching records between datasets from two sources that are internally duplicate-free (clean) or one or both are not internally deduplicated (dirty).
- **Clean/Dirty Multi-source EM:** Refers to the setting of matching records between datasets from more than two sources that are internally duplicate-free (clean) or one or more are not internally deduplicated (dirty).

From the mentioned scenarios, the contributions of this thesis are evaluated based on the clean/dirty two-source and clean/dirty multi-source scenarios.

2.3.1 Workflow

Figure 2.2 shows the workflow of the entity matching step in data integration. The input is made up of one or more datasets, which have aligned schemata as a result of the previous schema matching step. The inputs first go through a pre-processing procedure, followed by a blocking step. After the blocking step, the entity matching part of the workflow comprises the record pair comparison and record pair classification steps which result in a set of correspondences between the datasets [Christen, 2012b]. The final step performs a clustering of these correspondences into entity clusters [Hassanzadeh et al., 2009, Saeedi et al., 2017], with ideally all records referring to the same entity ending up in the same cluster. The following paragraphs give an overview of each step with a special focus on the entity matching part of the workflow.

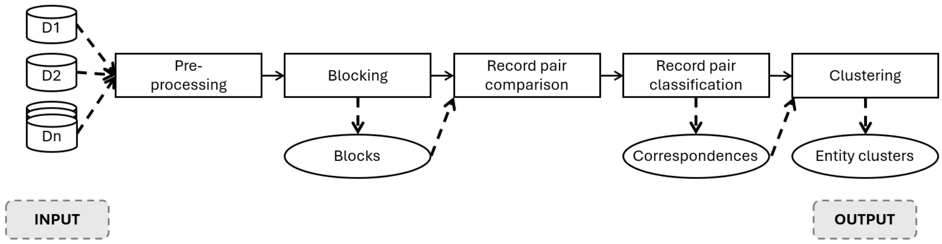


Figure 2.2: The entity matching workflow. Adapted from [Christen, 2012b].

Pre-Processing Step

The pre-processing step involves processing of the input records in a way that facilitates the later stages of the workflow. Pre-processing usually includes data transformation and standardization steps like value normalization, lower-casing, and string normalization to reduce heterogeneity among the values between datasets. The purpose of the pre-processing step is to facilitate finding a binary decision function for matching as described above. An overview of common pre-processing methods is available in [Christen, 2012b, Elmagarmid et al., 2007, Sarawagi, 2008].

Data transformation procedures include (1) removal of unwanted characters and stopwords, (2) expansion of abbreviations and correction of misspellings, and (3) segmentation of longer attributes into shorter attributes [Christen, 2012b]. An example of the latter is the separation of full address information into single attributes, e.g., *street*, *housenumber*, *city*, etc.

Data standardization steps ensure that the data in all datasets conforms to a pre-defined standard [Elmagarmid et al., 2007], e.g. a certain representation for addresses or dates is enforced across all datasets to facilitate comparison in the next steps.

Although most of these pre-processing steps have proven valuable for the entity matching workflow [Barlaug and Gulla, 2021], recent entity matching methods for textual data are based on pre-trained language models. These language models provide their own pre-processing and tokenization functions that assume no prior in-depth processing in an effort to keep data processing equivalent to the pre-processing applied to the pre-training data of the models [Devlin et al., 2019]. In these cases, applying extensive pre-processing beforehand may lead to lower performance of the following steps, e.g., lower-casing when using a cased language model will impact the resulting embeddings and skew contextual information possibly hurting matching performance (see Section 2.3.3).

Blocking Step

The next step in the entity matching workflow is the blocking step [Christophides et al., 2015]. This step is optional and aims to reduce the computational complexity of the following entity matching step. As entity matching is defined as a matching task between pairs of records, with increasing amounts of records across datasets, the cross product of pairs of records grows quadratically and becomes computationally intractable [Christen, 2012b, Christophides et al., 2020].

The blocking step is applied to avoid comparing every possible pair combination among the datasets. To this end, blocking methods usually provide an indexing function that is applied to every record and returns a blocking key [Bilenko et al., 2006] that is then used by an equality function that checks if two records are found under a common blocking key. As a result, only record pairs with a common blocking key pass the output of the blocking function and are compared in the entity matching step.

While easing the computational complexity problem, this step introduces the potential to negatively impact the following entity matching and data fusion steps. Finding good blocking keys is a non-trivial problem [Papadakis et al., 2020] and leads to a trade-off between eased computational requirements and missing true matches among the datasets. If two matching records do not end up in the same block, e.g., due to heterogeneity in their values that cannot be resolved during pre-processing, this record pair will never be matched and subsequently not fused in the later stages thus leading to redundancy in the final integrated view or dataset.

Christen [Christen, 2012b] divides blocking methods into four families: (1) standard blocking, (2) sorted neighborhood blocking, (3) q-gram-based indexing, and (4) canopy clustering. In recent years, deep learning-based blocking methods have become more prominent, which often do not follow the paradigm of blocking keys but instead directly leverage methods similar to deep entity matching systems [Papadakis et al., 2022]. These are based on distributional representations in the embedding space combined with a nearest-neighbor search in that space to find pairs of records to be compared. An overview of blocking methods can be found in the surveys of O'Hare et al. [O'Hare et al., 2019], Papadakis et al. [Papadakis et al., 2020] and Steorts et al. [Steorts et al., 2014]. Papadakis et al. further performed a comparison of blocking key-based and nearest-neighbor-based approaches [Papadakis et al., 2023b]. More recent approaches based on deep learning have turned to using contrastive learning for blocking [Brinkmann et al., 2024, Mugeni and Amagasa, 2022].

This thesis does not investigate the blocking problem, as all contributions focus on the matching step after the blocking has already been performed. All benchmark datasets used in this thesis already provide pair-wise training, validation and testing splits after blocking.

Matching Step

The matching step is the central part of the entity matching workflow. It consist of two steps, (1) record pair comparison and (2) record pair classification [Christen, 2012b]. It takes a set of record pairs as input and must provide a decision function that assigns either *matching* or *non-matching* to each pair. For this purpose, the matching algorithm needs to compare both records using some form of feature representation, resulting, e.g., in a similarity vector for each record pair [Christen, 2012b] in the record pair comparison step. The final output of this step, is a set of correspondences based on the identified matches between record pairs.

The record pair classification step makes a decision for each record pair given the similarity representation of the pair [Christen, 2012b]. Depending on the matching method, this classification step can be a separate step or can be integrated into the method. The former is usually the case for rule-based and unsupervised entity matching methods, while supervised methods perform the classification step together with the record pair comparison as part of the model [Christen, 2012b, Barlaug and Gulla, 2021]. Supervised non-neural methods usually train a machine learning model on a labeled set of matching and non-matching record pairs and automatically learn the similarity representation of a record pair and the decision threshold for record classification from manually created features as part of the model from the training data [Christen, 2012a]. Commonly used models for this task are decision trees, support vector machines, logistic regression and random forests [Elfeky et al., 2002, Christen, 2008, Konda et al., 2016].

For supervised neural methods, the process is similar to non-neural methods with the difference that the feature engineering step is part of the model architecture (see Section 2.3.3). Some models create similarity vectors attribute-wise before combining them [Mudgal et al., 2018] while more recent methods based on language models of the Transformer family [Brunner and Stockinger, 2020, Li et al., 2020, Peeters and Bizer, 2021] usually work with the serialization of complete records or record pairs even at the lowest level of the network as part of the contextualization mechanism of these models (see Section 2.3.3). The final classification step of the combined similarity vectors of a record pair in Transformer-based methods like BERT [Devlin et al., 2019] or RoBERTa [Liu et al., 2019b] is commonly handled using a simple feed-forward layer or network as the head of the model, resulting in a decision for match or non-match. Barlaug and Gulla [Barlaug and Gulla, 2021] provide an overview of a selection of neural methods for entity matching. For generative large language model-based methods, the matching decision is instead parsed from the generated natural language answer of the model following a prompt [Narayan et al., 2022] without an additional classification layer. The following paragraphs give a short overview of select methods from each category.

Early Methods: The works cited as foundational work for the matching task are the works of Newcombe and Kennedy [Newcombe and Kennedy, 1962] and Fellegi and Sunter [Fellegi and Sunter, 1969] who formulated some of the first methods for linking heterogeneous entities between datasets from different sources. The work of Newcombe and Kennedy is based on the manual curation of linkage rules over a series of iterations to improve the quality of results. The authors define the entity matching task as a Bayesian inference problem. Fellegi and Sunter build on this work, formalize the task, and propose a statistical model supporting the generation of matching rules.

Unsupervised Methods: Early unsupervised methods aggregate the resulting similarity vector into a single score and define a threshold value for classification into matches and non-matches [Bilenko and Mooney, 2003, Cohen and Richman, 2002, Monge and Elkan, 1996] while others rely on hand-crafted matching rules [Hernández and Stolfo, 1995,Lim et al., 1996] or clustering algorithms [Elfeky et al., 2002]. Monge and Elkan [Monge and Elkan, 1996, Monge, 1997] propose a matching algorithm based on string similarity metrics combined with a threshold to separate matching from non-matching record pairs. Dey et al. [Dey et al., 1998] propose the linear combination of similarities of each attribute in a record pair combined with a decision threshold for the matching decision. Cohen [Cohen, 2000] proposes the use of TFIDF vectors together with the cosine similarity metric for distance-based matching. More recent applications of unsupervised methods to entity matching are the work of Zhu et al. [Zhu et al., 2016] who apply a graph-based approach for matching on RDF-based graphs. Wu et al. [Wu et al., 2020] propose ZeroER based on Gaussian mixture models as an unsupervised method for entity matching.

Supervised Methods: In early supervised machine learning methods, the similarity vector of the record pair comparison step is created by applying various similarity metrics (see Section 2.3.2) attribute-wise to generate a similarity representation of each attribute of the compared records, which are then combined to a final similarity vector representation. Cochinwala et al. [Cochinwala et al., 2001] propose some of the first supervised learning techniques for entity matching based on the CART algorithm [Breiman, 2017] for generating classification trees, a linear combination of parameters and a nearest neighbor approach, resulting in the best performance for the tree-based approach. Bilenko et al. [Bilenko and Mooney, 2003] use an SVM with separate features for each matching attribute and show that this performs better than treating the entire record as one large textual attribute. Cohen and Richman [Cohen and Richman, 2002] propose a learnable distance function from training data as part of their supervised method. Konda et al. [Konda et al., 2016] propose Magellan, a framework for blocking and entity matching based on learning supervised classifiers on string and numerical similarity features between attributes. Another direction of supervised methods leverages active learning to minimize the amount of required labels for learning by keeping a human in the loop who subsequently labels sets of record pairs selected by the methods based on various heuristics [Sarawagi and Bhamidipaty, 2002, Isele and Bizer, 2013, Chen et al., 2019, Kasai et al., 2019, Bogatu et al., 2021, Huang et al., 2022, Primpeli and Bizer, 2022]. Meduri et al. [Meduri et al., 2020] give an overview of active learning methods for entity matching.

First Wave Neural Methods (RNN Era): Deep learning-based methods for entity matching do not rely on explicit feature engineering but instead perform it as part of their learning process in the subsymbolic space [Goodfellow et al., 2016]. These methods convert each record attribute-wise [Mudgal et al., 2018] or record-wise [Li et al., 2020] into a vector representation based on the tokens contained in their attributes and subsequently create features internally as part of learning the parameters of the layers of the deep neural network. The record representations are merged into a pair representation close to the output layer of the network. The foundational works for neural methods in entity matching are DeepER [Ebraheem et al., 2018] and Deepmatcher [Mudgal et al., 2018] who were the first to apply siamese neural networks to the entity matching task and showed an increase of matching performance for neural networks especially for textual entity matching tasks. Both methods convert records attribute-wise to embeddings using methods such as Word2Vec [Mikolov et al., 2013b] and fastText [Bojanowski et al., 2017]. The methods combine these embeddings into representations of each record as part of the network, and subsequently perform a classification step using feed-forward networks to decide if they match. Although the Deepmatcher framework offers various neural network designs, including one of the first attention-based architectures for entity matching, the recurrent neural network is found to perform consistently best in a wide range of entity matching tasks [Peeters et al., 2020b, Li et al., 2020]. Further examples from the early era of neural methods for entity matching are Seq2SeqMatcher [Nie et al., 2019] and Hi-EM [Zhao and He, 2019].

Second Wave Neural Methods (PLM Era): The second wave of neural methods for entity matching based on the Transformer architecture and pre-trained language models started with the work of Brunner and Stockinger [Brunner and Stockinger, 2020] who were the first to apply fine-tuning of PLMs to the task of entity matching showing increased performance compared to the neural methods of the RNN era. Li et al. [Li et al., 2020] proposed Ditto based on fine-tuned PLMs with additional data augmentation functionality which became the state-of-the-art for several years. Wang et al. [Wang et al., 2023] proposed Sudowoodoo based on a self-supervised contrastive learning method for low-resource matching scenarios. PromptEM [Wang et al., 2022c] moved towards the prompting paradigm of the LLM era (see below) by steering the models more towards a natural language answer to the question if a record pair matches compared to the earlier approaches that serialized both records and used the [CLS] token of the Transformer for classification. The authors show that this method can perform better, arguing that the serialization of the task is closer to the natural language pre-training of the PLMs. Mugeni et al. [Mugeni et al., 2023] showed that fine-tuning adapters instead of the full PLM can achieve the same or better result while saving on time and compute resources.

Third Wave Neural Methods (LLM Era): The third wave of neural methods is still in its infancy at the time of writing and is predominantly based on the use of generative large language models for entity matching [Narayan et al., 2022, Fan et al., 2024b, Sisaengsuwanchai et al., 2023, Zhang et al., 2024a, Wang et al., 2024, Wadhwa et al., 2024]. Narayan et al. [Narayan et al., 2022] laid the foundation with their first experiments using a GPT-based model for entity matching showing matching performance comparable to fully fine-tuned PLMs with only a few in-context examples as part of the prompt. While LLMs are technically pre-trained language models, they are not referenced as PLMs in the literature. The term PLM is used to refer to smaller models like BERT, whereas the term LLM is commonly used to refer to autoregressive language models with more than a billion parameters. This thesis follows the same naming scheme when referring to PLMs and LLMs. Concurrent work and the work presented in this thesis (see Chapter 9 for a deeper discussion of related work) show that LLMs have a high potential for the entity matching task as they show good performance for two of the shortcomings of PLM-based approaches. Their strong zero-shot and few-shot capabilities lead to requiring only a few labeled examples to reach or exceed the matching performance of fully fine-tuned PLMs. Furthermore, they consistently perform well even on unseen entities (see Chapter 9).

Domain Adaptation: Highly related to the concept of unseen entities (see Section 2.4) these methods focus on training models on one or more benchmark datasets and try to subsequently transfer the trained models to other benchmarks (which may be from the same or a different matching domain) without or with only a few additional training pairs of the target domain. Methods investigating domain adaptation can stem from any of the previously presented categories, while many of the recently proposed methods are based on neural networks [Jin et al., 2021, Loster et al., 2021, Kirielle et al., 2022, Trabelsi et al., 2022, Tu et al., 2022, Bai et al., 2023, Sun et al., 2024, Xu and Wang, 2024].

Clustering Step

The clustering step [Hassanzadeh et al., 2009, Saeedi et al., 2017] in entity matching follows the identification of matching record pairs as correspondences and aims to consolidate these correspondences into unified clusters that represent individual entities across datasets. This step often employs graph-based methods, where nodes represent records and edges denote pair-wise matches. A common approach is to apply transitive closure, ensuring that if Record A matches Record B, and Record B matches Record C, then all three records are grouped in a single cluster, even if a direct match was not initially identified between A and C. Such transitive relations help capture the latent connections between records, thus increasing the robustness of the clustering.

This clustering step presents additional challenges, as errors in the initial matching phase, such as false positives (incorrect matches), may propagate, leading to clusters that erroneously merge records of distinct entities. Likewise, false negatives (missed matches) can result in fragmented clusters, which fail to capture the entirety of an entity's records across datasets.

The outcome of the clustering step is a partitioning of all records into distinct clusters, with each cluster ideally corresponding to a unique entity. This consolidated view is essential for achieving the conciseness and completeness objectives of data integration, ensuring that each entity is represented only once in the integrated dataset.

For *clean* entity matching scenarios, the clustering step relies on enforcing the one-to-one correspondence between records of different data sources. Unique Mapping Clustering [Böhm et al., 2012, Lacoste-Julien et al., 2013] sorts all edges in the correspondence graph by decreasing weight (e.g., record similarity), then iterates through records and only considers the top edge as a true correspondence. This approach is similar to solving the stable marriage problem [McVitie and Wilson, 1970]. The Hungarian algorithm [Kuhn, 1955] is another example of a clustering approach for clean entity matching scenarios. More clustering algorithms for clean entity matching are presented in [Papadakis et al., 2023a].

The work of Saeedi et al. [Saeedi et al., 2017] compares various methods to perform the clustering step, which are suited for dirty entity matching, in their FAMER framework. Methods based on connected components [Junghanns et al., 2017] identify all connected subgraphs within the graph of correspondences and perform clustering based on the records that are part of these subgraphs. For example the transitive closure method described above is a simple example for a method based on connected components. While straightforward, this method has the potential of propagating erroneous matches as well as fragmenting clusters due to missing matches.

Center clustering [Hassanzadeh and Miller, 2009] assigns entities to clusters based on edge weights (the weights are the similarities of the two connected records as output by the matching step), prioritizing edges with high similarity. Initially, it focuses on edges with high similarity, making it effective in maintaining quality clusters. Merge Center [Hassanzadeh and Miller, 2009] is a variation of Center clustering that merges clusters if a record of one cluster is close to the center of another cluster. Although it can improve recall, it risks reducing precision due to merging similar but different entities.

Star clustering [Aslam et al., 2004] creates clusters by assigning entities with high node degrees as centers, which act as local hubs around which clusters are formed. The algorithm can lead to overlapping clusters, which requires additional post-processing steps to resolve these overlaps. Star-2 clustering [Saeedi et al., 2017], an alternative version of Star clustering, bases center selection on the average similarity degree of outgoing edges, rather than the count.

Correlation clustering [Bansal et al., 2004, Gionis et al., 2007, Chierichetti et al., 2014, Pan et al., 2015] is an approach that uses not only positive but also negative edge weights based on similarity to determine clusters, aiming to maximize internal agreements while minimizing disagreements between clusters. An example of an approximate version of Correlation Clustering used for scalability is CCPivot [Chierichetti et al., 2014], where multiple vertices are selected as initial cluster centers or *pivots* in each iteration, reducing the number of rounds needed to cluster all vertices.

Draisbach et al. [Draisbach et al., 2019] present Maximum Clique Clustering and Extended Maximum Clique Clustering which are based on extracting maximum cliques from the components of a graph iteratively where each maximum clique represents on entity cluster. They further present Global Edge Consistency Gain, which reclassifies edges within the clusters to maximize the consistency of the transitive relationship implied by the edge connections, thus refining the cluster boundaries more accurately. The authors compare their methods to a selection of the approaches mentioned above and additional clustering methods like Markov Clustering [Van Dongen, 2000] and GCCluster [Wang et al., 2016].

2.3.2 Similarity Metrics

Similarity metrics are used to calculate the similarity of single attribute values, complete records, or to compare embedding representations of the former. As a result, they are essential to any kind of matching task, e.g. in this thesis similarity metrics are used as part of matching algorithms as well as for the selection of corner case records (see Section 2.4 and Chapter 5). This section introduces a set of relevant similarity metrics that are commonly used throughout the data integration process and partly in the experiments of this thesis. The presented similarity metrics are an integral part of non-neural entity matching systems. Although neural networks perform feature engineering as part of their internal layers (as discussed in Section 2.3.3), most non-neural methods base the features of their matching classifiers on a set of similarity measures including, but not limited to, those presented below [Konda et al., 2016, Bilenko and Mooney, 2003]. For example, the Magellan entity matching system [Konda et al., 2016] may use one or multiple similarity metrics for each of the attributes in a record pair as input to a machine learning classifier. The representation of a record pair in Magellan is thus a multidimensional vector of similarities. This notion of multi-dimensional vectors and similarities is an integral concept that finds continuation in the use of embeddings in neural entity matching systems (see Section 2.3.3).

Vector Similarity

Vector similarity allows for the comparison of numerical vectors in embedding spaces. This can, for example, be a simple bag-of-words representation of two records using word occurrence or TFIDF weighting or the comparison of embeddings created by a neural network (see Section 2.3.3). A commonly used metric for comparing numerical vectors is cosine similarity, which measures similarity using the angle between two vectors. Given two vectors $\vec{A}$ and $\vec{B}$, cosine similarity is defined as:

$$\text{sim}_{cosine}(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}||\vec{B}|} \tag{2.1}$$

String Similarity

Much of the data in real-world use cases contains string values either as single words, multiple words, or even full sentences. For example, for the disambiguation of persons, it may be required to match attributes consisting of the first, middle, and last names of a person, which may appear in any kind of ordering across different datasets. Depending on the structure of the data, the choice of the optimal similarity metric is key. String similarity metrics are commonly divided into categories: character-based, token-based, hybrid, and phonetic metrics [Elmagarmid et al., 2007].

Character-based Metrics: These metrics compare strings character-wise e.g. by applying edit operations to one string to convert it into the other. Fewer necessary operations for this conversion result in a higher similarity being assigned. A well-known edit-based metric is the Levenshtein [Levenshtein et al., 1966] similarity metric. It first calculates the distance between two strings by counting the number of deletions, insertions, and substitutions necessary to convert one string into the other. Once the distance $\text{dist}_{lev}(a, b)$ is obtained, the Levenshtein similarity is defined as:

$$\text{sim}_{lev}(a, b) = 1.0 - \frac{\text{dist}_{lev}(a, b)}{\max(|a|, |b|)} \tag{2.2}$$

Other well-known character-based metrics are, for example, the Jaro [Jaro, 1989] similarity and its extension, the Jaro-Winkler [Winkler, 1990] similarity, which were proposed for the matching of names and put a higher importance on matching the beginning of strings.

Token-based Metrics: These metrics work at the word or n-gram level instead of the character level and are better suited to compare longer strings consisting of multiple words that may appear in different order, compared to character-based metrics [Elmagarmid et al., 2007]. Commonly used token-based metrics are Jaccard [Jaccard, 1912], Overlap [Christen, 2012a] and CosineTFIDF [Cohen et al., 2003].

The Jaccard [Jaccard, 1912] similarity measures the intersection between two sets of words or n-grams of two strings A, B over the union of their tokens. If the intersection is measured in relation to the size of the smaller set, it is Overlap similarity [Christen, 2012a]. They are defined as:

$$\text{sim}_{jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.3}$$

$$\text{sim}_{overlap}(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)} \tag{2.4}$$

Hybrid Metrics: This family of similarity metrics combines character-based and token-based metrics by allowing a degree of variability when estimating the equality of tokens. This allows them to combine the advantages of both by handling typographical errors within words or n-grams and differing orderings inside a string [Doan et al., 2012]. Examples of this type of metric are Monge-Elkan [Monge and Elkan, 1996], generalized Jaccard [Naumann and Herschel, 2022] and SoftTFIDF [Cohen et al., 2003]. Hybrid metrics usually comprise an inner character-based similarity metric and an outer token-based metric. For example, the generalized Jaccard similarity may contain an inner Levenshtein similarity that compares tokens and considers them matching if their edit distance is below a user-defined threshold θ [On et al., 2006]. Given a set of matching tokens $|\text{match}(A, B)|$ decided by an inner similarity metric sim_{inner} between the set of tokens between two strings, generalized Jaccard is defined as:

$$\text{match}(A, B) = \{(a_i, b_j) | a_i \in A \wedge b_j \in B : \text{sim}_{inner}(a_i, b_j) \geqslant \theta\} \tag{2.5}$$

$$\text{sim}_{gen.jac}(A, B) = \frac{|\text{match}(A, B)|}{|A| + |B| - |\text{match}(A, B)|} \tag{2.6}$$

Numerical Similarity

The previously presented metrics are suitable for the comparison of strings, but not other data types. Another example of a frequently occurring data type are numerals. For these cases, specialized metrics for numbers can be applied, for example, by calculating the absolute or percentage difference between two numbers and normalizing by a maximum absolute difference extracted from the data or defined by the user [Christen, 2012a] or by calculating a relative difference between the two values [Konda et al., 2016]. These similarities are defined for two numbers u, v as:

$$\text{sim}_{numabs} = 1.0 - \frac{|u - v|}{\max(u, v)} \tag{2.7}$$

$$\text{sim}_{numrel} = 1.0 - \frac{2 \times |u - v|}{u + v} \tag{2.8}$$

Numerical similarity can also be used to cover additional data types such as time, date, or geographic data by first converting and normalizing the respective type into a numerical format followed by application of the presented metrics or other metrics specific to the data type [Christen, 2012a].

2.3.3 Neural Language Representation

The previous section presented various similarity metrics for e.g. engineering features for the comparison of two records in a record pair. In neural methods for entity matching, neural networks perform feature engineering and similarity comparison as part of the network [Goodfellow et al., 2016, Barlaug and Gulla, 2021]. All neural methods for entity matching can theoretically be trained in an *end-to-end* fashion from inputs over feature engineering to outputs, as long as each layer is differentiable and thus trainable with the backpropagation algorithm.

Neural methods are based on tokenizing string inputs into words or tokens and subsequently converting them into embedding representations using the networks themselves [Devlin et al., 2019] or an external embedding representation as input [Mikolov et al., 2013b, Bojanowski et al., 2017]. The embedding representations used for words and tokens for entity matching can also be learned as part of the neural models, but it has been experimentally proven to be more effective to initialize them from an existing word modeling vocabulary such as fastText or a fully pre-trained language model like BERT [Barlaug and Gulla, 2021].

Neural entity matching models of the first era (see Section 2.3.1) which use embeddings as numerical representations of the input tokens but do not consist of a full language model like those of the second era, usually use an external model to create the input embeddings to the actual matching model [Ebraheem et al., 2018, Mudgal et al., 2018].

These embeddings are based on simple language models from the Word2Vec family [Mikolov et al., 2013b, Mikolov et al., 2013a], which are trained on textual corpora like Wikipedia. Word2Vec is based on the distributional assumption that similar words appear in similar contexts. The language model is trained as a simple feed-forward network in two configurations. The continuous bag-of-words (CBOW) model is trained by using left and right context words to predict a masked center word, while the skip-gram model is trained to predict the surrounding context words from the center word. The authors have shown that this training on large textual corpora like Wikipedia results in a trained feed-forward network that

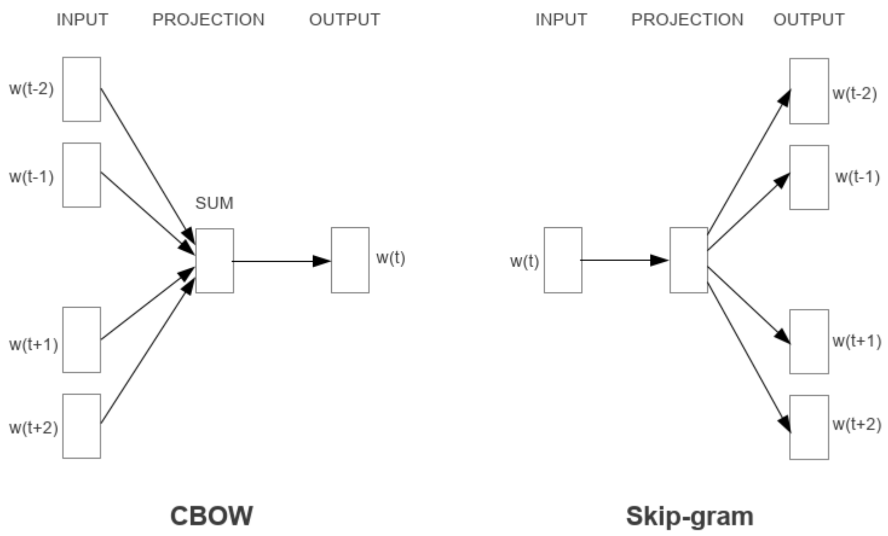


Figure 2.3: The CBOW and Skip-gram variations of the Word2Vec model [Mikolov et al., 2013a].

can embed words while semantically similar words are located in close proximity in the resulting embedding space. The two variations of the Word2Vec model are depicted in Figure 2.3.

The fastText [Bojanowski et al., 2017] model extends on the idea of Word2Vec by not training on words but sub-word tokens, which allows for the representation of words unseen during training by aggregating the word-specific sub-word tokens that are part of the vocabulary of fastText. Compared to the local context learning of Word2Vec and fastText, GloVe [Pennington et al., 2014] word embeddings are created based on global co-occurrence statistics across the whole training corpus.

The research done during the first era of neural methods for entity matching has shown that this notion of distributional similarity among word embeddings enhances the entity matching task compared to randomly initializing embeddings [Ebraheem et al., 2018, Mudgal et al., 2018, Barlaug and Gulla, 2021]. The methods of the first era use these language representations in the form of externally learned, fixed embeddings on the input level of the network. The higher layers of the network are then tuned based on the entity matching task, learning more entity matching specific features as part of the network. This is done for each input record/attribute separately or with cross-attention to the other record/attribute in a record pair [Mudgal et al., 2018]. The final layers of the network then aggregate the separate embeddings to a similarity representation of the record pair, e.g. via summarization, difference, dot product, or cosine similarity. This final similarity representation, based on the original language embeddings, is then passed through a classifier to obtain the matching decision.

Figure 2.4 shows an example of this layered architecture starting with em-

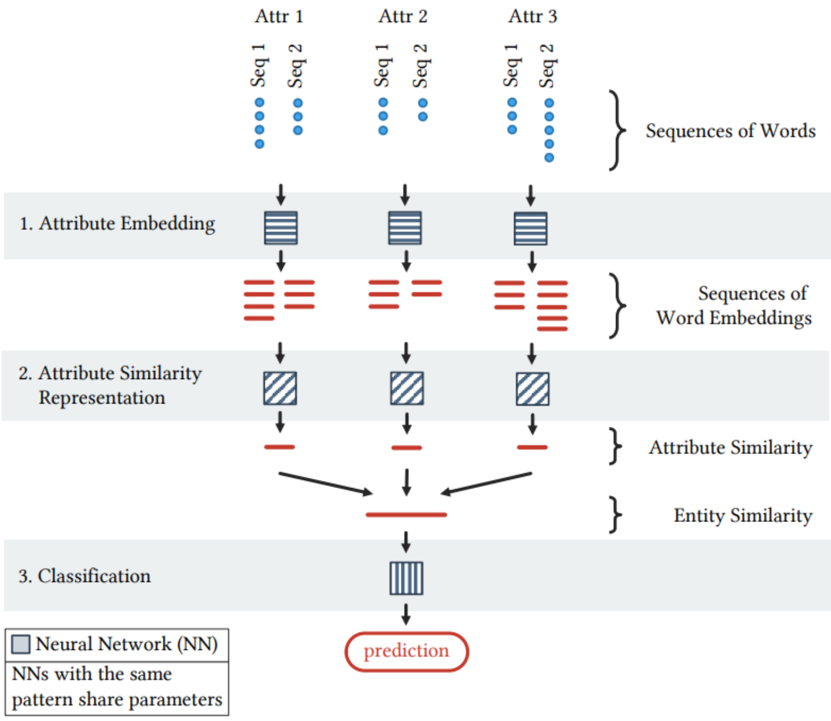


Figure 2.4: The layered Deepmatcher training architecture [Mudgal et al., 2018].

bedding attributes separately using, e.g., fastText embeddings and consequently aggregating them to attribute similarity representations and finally a record pair similarity as input to the final classifier. This structure of aggregating attribute similarities to a record similarity and subsequent classification is analogous to the process found in methods previously used in non-neural matching systems like Magellan [Konda et al., 2016] (see Section 2.3.2).

Entity matching methods of the first era are usually not trained end-to-end as their input embeddings (e.g., fastText) are fixed during training [Barlaug and Gulla, 2021]. Section 4.7.1 of this thesis shows that allowing full end-to-end training for these models can improve the matching performance.

The neural entity matching methods of the second and third eras [Brunner and Stockinger, 2020, Li et al., 2020, Wang et al., 2022c, Narayan et al., 2022, Wang et al., 2024] which are based on the Transformer [Vaswani et al., 2017] architecture make use of architecture-specific embedding functionality in contrast to using an externally learned language representation as they are based on PLM and LLM language models at their core. In existing entity matching methods, they are commonly trained in a full end-to-end fashion which includes fine-tuning the input embeddings for the matching task.

Autoencoding PLM models like BERT [Devlin et al., 2019] and RoBERTa [Liu et al., 2019b], for example, use a WordPiece or SentencePiece[1] tokenizer to split words in record pairs into sub-tokens that are part of the Transformers original vocabulary from the pre-training stage. There is no direct separation between attributes of a record in this format, as the full record is passed as a string of tokens, although additional tokens can be added to inform the model of the structure of the data [Li et al., 2020]. The vocabularies of the PLMs, which are mappings from tokens to multi-dimensional embeddings, are created in such a way that even single characters can be represented, which means that these models can represent any kind of word made up of the known set of characters and tokens. The embeddings of the vocabulary tokens themselves are learned as part of the pre-training procedure. For example, BERT is pre-trained on the English version of Wikipedia as well as BookCorpus data using multi-task learning in the form of masked language modeling and next sentence prediction [Devlin et al., 2019]. Input embeddings PLMs and LLMs often consist of an aggregation of multiple embeddings, for example, the token embeddings themselves and a positional embedding encoding the location of the token in comparison to others in the sentence.

The subsequent fine-tuning of these models, for example, for the entity matching task, includes the fine-tuning of the embedding layer itself, which is another differentiation point of these methods from the first era of neural methods where Word2Vec or fastText embeddings were fixed and not adapted as part of the model training procedure [Mudgal et al., 2018]. As part of the fine-tuning, the embeddings of the input layer are propagated through the network while contextualized on all other tokens as part of the bi-directional self-attention layers [Vaswani et al., 2017] of autoencoders. In the last layer of the Transfomer encoder, either the embedding representation of a summary token that represents the full sequence or the aggregation of all token embeddings is used as input to a final classification layer that outputs the final matching decision.

Although PLMs can be used to embed each record in a pair of records separately and consequently combine the representations of each through, e.g. the dot product, which can be subsequently passed to a classifier, it has been shown that concatenating both records as input to the PLM leads to better performance due to the self-attention mechanism applying to all tokens across records [Barlaug and Gulla, 2021]. Figure 2.5 shows the end-to-end training architecture of the Ditto [Li et al., 2020] entity matching system. Both records are converted to string representations, concatenated, tokenized, and propagated through the Transformer layer. At the last layer, the contextualized similarity representation of the record pair (the prepended [CLS] token) is further passed into a classification layer that decides if the record pair is a match. The Ditto architecture is trained in a true end-to-end fashion, with all layers including the embedding layer being adapted for the matching task.

Large language models of the third era are based on the same Transformer architecture as PLMs but were pre-trained using an autoregressive decoder-style

[1] https://github.com/google/sentencepiece

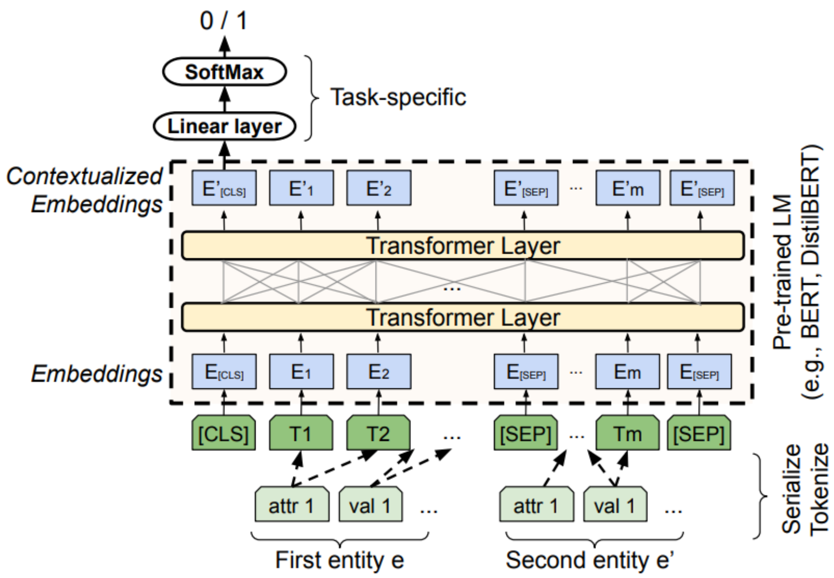


Figure 2.5: The Ditto end-to-end training architecture [Li et al., 2020].

language modeling objective, specifically next word prediction, on a large scale. General pre-training is followed by additional instruction fine-tuning [Sanh et al., 2022] on a multitude of tasks, and optional reinforcement learning from human feedback [Ouyang et al., 2022] to further adapt the model to human preferences. The output of entity matching systems using these models is based on the generation of natural language instead of using similarity embeddings in combination with a classifier as in PLMs. The input to these models is realized via natural language prompting consisting of a task description as well as one or more record pairs for which a matching decision is to be made [Narayan et al., 2022, Zhang et al., 2023, Wang et al., 2024]. The model then generates a natural language answer conditioned on the input prompt and the knowledge stored in the model parameters.

2.3.4 Evaluation Metrics

The evaluation practices used to evaluate the extent to which a method was successful in performing the entity matching task make use of standard measures from machine learning and have found wide adoption in the data integration community [Christen and Goiser, 2007]. The pair-wise entity matching task is defined on pairs of records that must be assigned either a *matching* or *non-matching* label. When evaluating entity matching systems, these pairs of records and their corresponding labels are available as record/label tuples in the form of (r_1, r_2, y_{12}). The task of an entity matching system is then to predict the correct label $y_{ij} \in \{0, 1\}$ for each pair of records. The label predicted by the method, $\hat{y}_{ij}$, is then compared to the true label y_{ij} and sorted into one of the following categories:

- **True Positives (TP):** A matching pair of records $(r_a, r_b, y_{ab} = 1)$ that was predicted to be matching $(r_a, r_b, \hat{y}_{ab} = 1)$.
- **False Positives (FP):** A non-matching pair of records $(r_c, r_d, y_{cd} = 0)$ that was predicted to be matching $(r_c, r_d, \hat{y}_{cd} = 1)$.
- **True Negatives (TN):** A non-matching pair of records $(r_c, r_d, y_{cd} = 0)$ that was predicted to be non-matching $(r_c, r_d, \hat{y}_{cd} = 0)$.
- **False Negative (FN):** A matching pair of records $(r_a, r_b, y_{ab} = 1)$ that was predicted to be non-matching $(r_a, r_b, \hat{y}_{ab} = 0)$.

After sorting each prediction into one of the four categories, summary predictions can be calculated from them. The metrics most commonly used for binary classification problems are *accuracy*, *precision*, *recall* and *F1-score*.

Accuracy measures how many of the overall number of pairs of records were classified correctly and is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.9}$$

Precision measures how many predicted matches are actually true matches:

$$Precision = \frac{TP}{TP + FP} \tag{2.10}$$

Recall measures the fraction of correctly classified matches compared to all true matches:

$$Recall = \frac{TP}{TP + FN} \tag{2.11}$$

The F1-score is the harmonic mean between the two measures precision (P) and recall (R) and is defined as:

$$F1 = \frac{2PR}{P + R} \tag{2.12}$$

From the presented measures, precision, recall and F1-score are commonly used in the literature [Christen, 2012a, Christophides et al., 2015, Barlaug and Gulla, 2021] as the pair-wise entity matching task, due to its nature, has an intrinsic imbalance between matching and non-matching records. Depending on the size of the datasets, the overall number of matches is several orders of magnitude smaller than the number of non-matches. Using accuracy for evaluation in this scenario may lead to overestimation of the actual performance of entity matching systems, as well as difficulty in evaluating differences between matching systems. As a result, entity matching performance is usually measured by calculating precision, recall and F1 on the matching class.

2.4 Additional Terminology

This section introduces additional terminology used throughout the thesis, which has either not been clearly defined in entity matching literature or is adapted from specific machine learning terminology to the entity matching domain.

Corner cases

Corner cases are matching and non-matching record pairs that are close to the decision boundary between the two classes *match* and *non-match*. Such examples define the decision boundary between classes and are commonly referred to as hard positives and hard negatives in machine learning literature [Simo-Serra et al., 2015, Schroff et al., 2015, Zhan et al., 2021].

For the entity matching task, this hardness can be expressed based on textual and numerical similarity of record pairs using similarity metrics such as those presented in Section 2.3.2. Textual metrics like Jaccard can be directly applied to the original tokens of records. Further, embedding-based methods can be used to embed each record and subsequently calculate, for example, cosine similarity between embeddings combined with nearest-neighbor search in the embedding space to find corner cases.

Corner case pairs selected in this manner exhibit the property of resembling a pair of the respective other class. For example, in the context of product matching, a *positive corner case* refers to a pair of matching product offers that exhibit dissimilarities in their surface forms, which are usually the result of heterogeneity introduced by different vendors, for example, mentioning different product features in the offers or using different abbreviations or units of measurement. A *negative corner case* consists of two non-matching product offers whose textual representations are highly similar, e.g., may differ only in a single product feature.

Evaluation Set Naming

This section clarifies the naming of the various evaluation sets used throughout the thesis and to what parts of the evaluation process they refer (see [Bishop and Nasrabadi, 2006, James et al., 2013, Goodfellow et al., 2016]).

Training Set: The training set contains all record pairs that are used for directly training supervised machine learning-based entity matching models, i.e. these record pairs and their labels are shown to the model as examples for matches and non-matches during the training procedure.

Validation Set: The validation set contains all record pairs that are used for model selection and hyperparameter optimization of the models. These pairs are not shown directly to the model but are used as a means to evaluate changes in model performance when parameters of the training change. These record pairs are not part of the training set and are as such unknown to the models.

Development Set: The development set consists of all pairs that are part of the training set and the validation set. When using the term development set, it is not necessary that the training and validation subsets are already defined, as these may be created at a later stage by splitting the development set. The term development set refers to all pairs of records that are used for training or for model selection. The development set is only used in entity matching systems based on supervised learning. Unsupervised methods or supervised systems that are applied zero-shot (see below) do not use the development set but are directly evaluated on the test set.

Test Set: The test set encompasses all pairs of records that are not part of the development set. The purpose of this set is an unbiased evaluation of trained models on unknown data that have not been used for training or for model selection. The test set is the main reference point for evaluating entity matching systems, as the performance on this set most closely resembles the expected unbiased performance during deployment on new record pairs.

Generalization to Unseen Entities

The generalization phenomenon in machine learning [Dziugaite et al., 2020,Musavi et al., 1994] refers to the observable performance loss when transferring a model from development data to the withheld test data. Due to a shift in the data distribution, a performance loss can be measured by, for example, comparing the evaluation metrics on the validation and test sets. In the case of entity matching, this effect is usually measured by comparing the performance metrics introduced above on the validation and test sets.

The special case of generalization to *seen* or *unseen* entities describes the strength of the distribution shift from development to test data. For example, in the product matching domain, the smallest shift happens when the model is tasked to decide for match or non-match on a pair of product records that refer to products for which example pairs have already been seen during training. A shift in structure or semantics may occur, but the pair of records is close to the training distribution. This is further referenced as a *seen* scenario in this thesis. The *unseen* scenario happens when a trained product matching model needs to decide on a match or non-match for record pairs of products for which no matching or non-matching pairs were seen in the training set. In this case, the domain is the same, and it is expected that patterns learned during training will transfer to an extent, as products usually exhibit a similar structure in their descriptions. But if the same model needs to decide whether two records from the publication domain refer to the same publication, while the matching task is defined as the same, the actual wording and structural and semantic patterns differ from the patterns learned for matching products during training. In this regard, generalization to unseen entities is also related to the field of transfer learning or domain adaptation in entity matching (see Section 2.3.1).

This thesis defines an *unseen entity* as an entity that is contained as part of the records in the test set but no records of that entity appear in the development set. Unseen entities are also called out-of-distribution entities [Yang et al., 2024]. In the context of pair-wise entity matching, a test pair may contain records of (1) zero, (2) a single, or (3) two unseen entities. If the first case is true for the entire test set, this is referenced as a *fully seen* matching scenario. If the third case is true for the entire test set, it is a *fully unseen* matching scenario. Any other combination of pairs from the three cases is named a *partly unseen* scenario.

The concept of seen and unseen entities is integral to the entity matching task, as in many domains additional previously unknown entities can appear either due to not including them as part of the development set at creation time, applying matchers on domains that they were not specifically trained for, or due to e.g. the introduction of new entities over time. For example, new products constantly appear in online shops, new publications are written, and new companies are established. The challenge of handling unseen entities is discussed at multiple points throughout the thesis (see Chapters 4 through 9, excluding Chapter 6) and is one of the matching challenge dimensions of the WDC Products benchmark presented in Chapter 5.

Chapter 3

Semantic Annotations on the Web

3.1 Background

The Web has become an indispensable resource that redefines the way information is disseminated, accessed, and used. As the Web has grown, so too has the complexity of managing and interpreting the vast amounts of information it holds. The Semantic Web, a visionary concept proposed by Tim Berners-Lee [Lassila et al., 2001], aims to address this challenge by making web data more interpretable to machines. This enhanced understanding facilitates more efficient data sharing, reuse, and interoperability across various systems.

At its core, the Semantic Web extends the traditional Web by embedding metadata within web content, thereby enabling machines to interpret the context and meaning of the data. This paradigm shift is underpinned by various standards and technologies, including the Resource Description Framework (RDF) [Klyne, 2004], the Web Ontology Language (OWL), and the SPARQL Protocol and RDF Query Language (SPARQL) [Seaborne and Prud'hommeaux, 2008], which collectively support machines in performing sophisticated data integration and inference tasks.

Central to the implementation of the Semantic Web is the concept of semantic annotations, facilitated by domain-independent markup formats and annotation vocabularies. Semantic annotations involve enriching web content with metadata that provides explicit information about the data's semantics. This process typically involves tagging elements of web content with ontological terms, which describe their meaning in a structured, machine-readable format [Guha et al., 2016]. Semantic annotations thus bridge the gap between human-readable content and machine-interpretable data, enabling automated agents to interpret and manipulate the information.

Utilizing markup formats and vocabularies to semantically annotate web data enhances the accessibility, crawlability, and searchability of webpage content [Mika, 2015]. This practice is beneficial for various web applications, such as web search, price comparison, and reservation engines [Guha et al., 2016]. The use of semantically annotated data has become increasingly crucial for web data publishers as more web applications make use of it. To ensure consistent semantic terminology, major search engines such as Bing, Google, Yahoo and Yandex collaborated in 2011 to create the Schema.org[1] vocabulary. The adoption of semantic annotations on the Web has grown substantially over the years. In 2010, 5.7% (147 million) of the examined webpages included semantically annotated data, and this figure increased to 49. 9% (1.7 billion) by 2020 [Brinkmann et al., 2023a], according to the Web Data Commons project [2].

Schema.org annotations play a central role in the contributions of this thesis. Chapter 4 presents a benchmark and evaluation based on using Schema.org marked-up product data on the Web to automatically generate training data for entity matching methods on a large scale. To this end, product identifiers found in Schema.org annotations of products are used to group them into clusters of product offers referring to the same product automatically. Chapter 5 presents a second benchmark that is created from such a clustering.

This chapter is structured as follows. Section 3.2 introduces the main markup formats, while Section 3.3 presents the Schema.org vocabulary. Finally, Section 3.4 presents how Schema.org annotations have been used to create two large table corpora as part of this thesis.

3.2 Markup Formats

Markup formats are essential tools that web developers use to embed structured data directly into HTML documents. These formats allow for the precise definition of data elements on a webpage, making it easier for search engines to index and display content in a rich and meaningful manner. This section explores various markup formats, including Microdata, RDFa, Microformats, and JSON-LD, examining their structures, use cases, and implementations.

3.2.1 Microdata

Microdata[3] is a specification used to nest metadata within existing content on web pages. This format leverages HTML5 attributes to annotate content, making it easier for search engines to parse and understand the data. Microdata utilizes a vocabulary such as Schema.org to define the types and properties of items. The following attributes are essential components of Microdata:

- *itemscope*: Creates a new item and indicates that the HTML element contains information about an item.
- *itemtype*: Specifies the type of item using a URL that defines the vocabulary (e.g., Schema.org).

[1] https://schema.org/
[2] https://webdatacommons.org/structureddata/
[3] https://www.w3.org/standards/history/microdata/

- *itemprop*: Indicates that the element contains the value of the specified property for the item.
- *itemid*: Provides a global identifier for the item.

Figure 3.1 shows an example of a product marked up with Microdata.

```
<div itemscope itemtype="http://schema.org/Product">
  <span itemprop="name">Amazing Product</span>
  <span itemprop="price" content="19.99">$19.99</span>
  <span itemprop="description">This is an amazing product.</span>
</div>
```

Figure 3.1: An example product marked up with Microdata.

3.2.2 RDFa

Resource Description Framework in Attributes[4] (RDFa) is a standard for embedding rich metadata within web documents. RDFa extends HTML5 by providing additional attributes to annotate elements with structured data. RDFa enables interoperability by using URIs to denote the type and relationship of data elements. This format supports various vocabularies, including Schema.org and Dublin Core[5]. The following attributes are essential components of RDFa:

- *vocab*: Defines a default vocabulary for the document or element.
- *typeof*: Specifies the resource type using a URL.
- *property*: Indicates that the element contains the value of the specified property.
- *content*: Provides the value of a property when the content is not human-readable.
- *resource*: Defines a URI for the subject resource.

Figure 3.2 shows an example of a product marked up with RDFa.

```
<div vocab="http://schema.org/" typeof="Product">
    <span property="name">Amazing Product</span>
    <span property="price" content="19.99">$19.99</span>
    <span property="description">This is an amazing product.</span>
</div>
```

Figure 3.2: An example product marked up with RDFa.

[4] https://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/
[5] https://www.dublincore.org/

3.2.3 Microformats

Microformats [6] are a set of simple open data formats built on existing and widely adopted standards. They are designed to be easy to understand and implement, using class names and attributes to embed metadata into HTML content. Microformats focus on providing a human-readable and machine-processable structure without requiring significant changes to existing HTML code. Key attributes and conventions in Microformats include:

- *class*: Utilizes specific class names to indicate properties of the data (e.g., h-product, p-name, p-price).
- *rel*: Defines a relationship between the current document and another document.

Figure 3.3 shows an example of a product marked up with Microformats.

```html
<div class="h-product">
    <span class="p-name">Amazing Product</span>
    <span class="p-price" content="19.99">$19.99</span>
    <span class="p-description">This is an amazing product.</span>
</div>
```

Figure 3.3: An example product marked up with Microformats.

3.2.4 JSON-LD

JavaScript Object Notation for Linked Data[7] (JSON-LD) is a method of encoding Linked Data using JSON. JSON-LD allows for the embedding of metadata within a script tag, separate from the HTML content. This approach provides a clean and flexible way to add structured data to web pages without altering the HTML structure. JSON-LD is favored by search engines and is recommended by Google for structured data. Key attributes in JSON-LD include:

- *@context*: Provides the context for the data, usually a URL pointing to the schema definition.
- *@type*: Defines the type of the item.
- *@id*: Provides a unique identifier for the item.
- *property names*: Specific properties are defined within the JSON structure, such as name, price, and description.

Figure 3.4 shows an example of a product marked up with JSON-LD.

[6] https://microformats.org/2005/06/20/welcome
[7] https://www.w3.org/community/json-ld/

```
<script type="application/ld+json">
{
    "@context": "http://schema.org",
    "@type": "Product",
    "name": "Amazing Product",
    "price": "19.99",
    "description": "This is an amazing product."
}
</script>
```

Figure 3.4: An example product marked up with JSON-LD.

3.2.5 Comparison of Markup Formats

Each markup format has its unique strengths, making them suitable for different scenarios in web development. Microdata and RDFa are similar in that they embed structured data directly within HTML tags, creating a close association between content and its metadata. This close integration can be advantageous for developers who prefer a more declarative approach, where the data and its structure are easily visible and maintained within the HTML itself. This method also allows for easier updates and immediate visibility of the structured data in the source code.

Microformats, while adhering to a simpler methodology, employ specific class names to encode metadata. This approach reduces complexity and is more accessible for developers who are looking for straightforward implementation without needing to learn extensive new syntax. Microformats excel in simplicity and human-readability, making it easy to adopt and integrate into existing web pages with minimal changes.

JSON-LD stands out due to its separation of structured data from HTML content. By embedding JSON-LD within a script tag, developers can keep their HTML clean and focused purely on presentation while all the structured data is maintained in a well-organized JSON format. This separation simplifies the HTML structure and allows for more complex data relationships and easier updates, as the JSON data can be dynamically generated and manipulated without altering the HTML. JSON-LD is particularly favored by search engines like Google[8], as it provides a robust and flexible way to implement structured data that can be easily processed and indexed.

A critical advantage of JSON-LD over the other formats is its ability to handle more complex data structures and relationships. Since JSON is a widely used format for data interchange, integrating JSON-LD with other systems and APIs is often more straightforward. Additionally, JSON-LD's clear separation from HTML content reduces the risk of errors during web development and maintenance, as the structured data can be managed independently from the web page's design and content. While Microdata and RDFa can also represent complex relationships through nested structures, they require careful attention to HTML syntax and attributes.

[8]https://developers.google.com/search/docs/appearance/structured-data/intro-structured-data

3.3 Schema.org Vocabulary

Schema.org is a collaborative, community-driven initiative founded by Google, Microsoft, Yahoo, and Yandex with the goal of creating a unified vocabulary for structured data on the Web [Guha et al., 2016, Kanza et al., 2018]. Launched in June 2011, Schema.org provides a collection of schemas, or structured data formats, that webmasters can use to mark up their pages in ways recognized by major search providers. This effort helps search engines better understand and display content, enhancing the quality of search results and making it easier for users to find the information they need. Applications that use Schema.org data include Google Shopping, Google for Jobs, and Google Dataset Search[9].

3.3.1 Development of Schema.org

The development of Schema.org stemmed from the need for a standardized approach to structured data on the Web. Prior to Schema.org, various markup languages and vocabularies existed, leading to inconsistencies and inefficiencies in how search engines indexed and presented web content. Early attempts like Yahoo's SearchMonkey in 2008 and Google Rich Snippets in 2009 led to the establishment of data silos [Mika, 2015]. By consolidating multiple vocabularies into a single, extensible framework, Schema.org aimed to streamline the process of adding structured data to web pages and ensure compatibility across different search engines.

Schema.org's development has been a collaborative and iterative process involving input from a wide range of stakeholders, including webmasters, developers, and search engine providers. In its original iteration, Schema.org comprised 297 classes/types and 187 relations/properties [Guha et al., 2016]. By May 2024, the Schema.org vocabulary has been released in version 27[10]. At its inception, Schema.org was designed to be extensible in two primary ways, which were formally defined in 2015 as hosted and external extensions [Guha et al., 2016]. Hosted extensions are developed in collaboration with the broader community and subject matter experts and are integrated into the core Schema.org vocabulary. In contrast, external extensions are managed independently and are not incorporated into the core Schema.org structure. A notable example of an external extension is the GS1[11] web vocabulary, which provides detailed product data descriptions.

3.3.2 Structure of the Schema.org Vocabulary

The Schema.org vocabulary is structured as a hierarchy of types and properties designed to cover a broad spectrum of web content. The key components of its structure are:

[9] https://developers.google.com/search/docs/appearance/structured-data/search-gallery
[10] https://schema.org/docs/releases.html
[11] https://www.gs1.org/voc/

Product

A Schema.org Type

Thing › Product

Any offered product or service. For example: a pair of shoes; a concert ticket; the rental of a car; a haircut; or an episode of a TV show streamed online.

Property	Expected Type	Description
Properties from Product		
additionalProperty	PropertyValue	A property-value pair representing an additional characteristic of the entity, e.g. a product feature or another characteristic for which there is no matching property in schema.org. Note: Publishers should be aware that applications designed to use specific schema.org properties (e.g. https://schema.org/width, https://schema.org/color, https://schema.org/gtin13, ...) will typically expect such data to be provided using those properties, rather than using the generic property/value mechanism.
aggregateRating	AggregateRating	The overall rating, based on a collection of reviews or ratings, of the item.
brand	Brand or Organization	The brand(s) associated with a product or service, or the brand(s) maintained by an organization or business person.
gtin12	Text	The GTIN-12 code of the product, or the product to which the offer refers. The GTIN-12 is the 12-digit GS1 Identification Key composed of a U.P.C. Company Prefix, Item Reference, and Check Digit used to identify trade items. See GS1 GTIN Summary for more details.
gtin13	Text	The GTIN-13 code of the product, or the product to which the offer refers. This is equivalent to 13-digit ISBN codes and EAN UCC-13. Former 12-digit UPC codes can be converted into a GTIN-13 code by simply adding a preceding zero. See GS1 GTIN Summary for more details.

Figure 3.5: Part of the Schema.org documentation for the Product class.

Types: Types are the core building blocks of Schema.org, representing entities such as *Person*, *Organization*, *Product*, *Event*, and many others. Each type defines a set of properties that are relevant to that entity. Types are organized in a hierarchical manner, allowing for the inheritance of properties. For example, *Person* is a subtype of *Thing*, meaning it inherits properties from Thing.

Properties: Properties describe attributes of types. For example, a *Person* type has properties such as *name*, *birthDate*, and *address*. Properties can be specific to a type or shared across multiple types. The vocabulary includes data types for properties, specifying what kind of values are expected (e.g., *Text*, *Number*, *Date*, *URL*).

Data Types: Schema.org defines several data types to ensure that properties have well-defined formats. These include basic types like *Text*, *Number*, and *Boolean*, as well as more complex types like *Date*, *Time*, *URL*, and *StructuredValue*.

Enumeration: Some properties have a fixed set of values, known as enumerations. For example, the *BookFormatType* enumeration for a *Book* type includes values like *Paperback*, *Hardcover*, and *EBook*.

Extensibility: Schema.org is designed to be extensible. The core vocabulary can be expanded with extensions to cover more specific needs or emerging domains. Extensions allow for the integration of specialized vocabularies into the main Schema.org framework without disrupting its overall structure.

Figure 3.5 shows an example of the documentation of the Schema.org class *Product*. The properties related to GTIN numbers are used in this thesis to automatically group product offers on the Web into clusters of the same product and subsequently generate training data, as well as benchmarks (see Chapters 4 and 5) from this clustering. The next section presents two large table corpora from Schema.org annotations that were created as part of the work on this thesis.

3.4 The WDC Schema.org Table Corpora

Schema.org has found broad adoption as millions of websites have started to use the Schema.org vocabulary to provide structured data within their pages. At the time of writing, approximately 50% of all web pages contain Schema.org annotations [Brinkmann et al., 2023a].

The Web Data Commons (WDC) project[12] regularly extracts Schema.org data from the Common Crawl[13], the largest public web corpus. This extraction is the basis for the creation of the benchmarks presented in this thesis. The CommonCrawl is released monthly and typically contains approximately 3 billion HTML pages originating from over 30 million different websites (hosts). The WDC project uses the extracted data to calculate statistics about the adoption of Schema.org on the Web [Brinkmann et al., 2023a]. In addition, it publishes the extracted data in the form of N-Quads[14], a provenance-enabled graph data format.

This section presents a process of how this extracted data can be directly used for applications that require tabular data without requiring users to sort through large amounts of N-Quads for the data they seek. These tables can further be used to create additional benchmarks similar to those presented in this thesis. The WDC Schema.org Table Corpora are created from the extracted data by (1) grouping the data by website, (2) removing incomplete entities that were extracted from listing pages, and (3) deduplicating it. The resulting relational tables are presented in a JSON format. This section presents two releases of the WDC Schema.org Table Corpus: The 2020 release consists of 4.2 million relational tables that together contain $\sim$292 million rows of data. The corpus was generated from the WDC 2020 JSON-LD and Microdata extraction. The second corpus contains 5 million tables with $\sim$361 million rows of data generated from the WDC 2023 extraction.

The tables in the corpora belong to 44 different Schema.org classes, with *Product*, *LocalBusiness*, and *Event* being the most widely used classes. As the tables are

[12] http://webdatacommons.org/structureddata/

[13] https://commoncrawl.org/

[14] https://www.w3.org/TR/n-quads/

generated from Schema.org annotations, all tables that describe entities of a specific type use the same set of attributes to describe these entities, i.e. all tables use a shared schema. However, since the data originates from more than 4.3 million different websites (hosts) across the Web, the actual data values are heterogeneous in terms of the value format, the unit of measurement, and the language. The code to reproduce the creation process presented in the following is available on GitHub[15]

The work presented in this section has been previously published in [Peeters et al., 2024a].

3.4.1 Creation Process

This section describes the process of creating the WDC Schema.org Table Corpora. The 2023 release of the corpus is used to illustrate the process in terms of number of tables, number of rows, and the required amount of computation.

Extracting Data from the Common Crawl

The WDC project has developed a parsing framework for extracting structured data from the Common Crawl [Meusel et al., 2014]. This framework runs in the AWS cloud and supports the parallel processing of multiple (W)ARC files. To extract JSON-LD, Microdata, RDFa, and Microformats data from the HTML pages contained in the (W)ARC files, the framework uses the Any23 parser library[16]. For the 2023 release, 250 AWS spot instances with 8 $\times$ 3.2 GHz CPUs and 16 GB RAM were used for the extraction, which altogether required 4,602 machine hours. The extracted corpus consists of 97 billion RDF quads (N-Quads). Webmasters primarily use JSON-LD and Microdata syntaxes to annotate web pages with Schema.org terms[17]. Consequently, the extracted JSON-LD and Microdata data is merged to form class-specific subsets for selected Schema.org classes. The subsets consist of all entities of a specific class and entities of other classes present on the same page and contain 39 billion RDF quads[18]. Five days of compute time on a local shared server equipped with 96 $\times$ 3.6 GHz CPU cores and 1024 GB RAM were necessary to create the Schema.org subsets.

Grouping by Host

In the second step, all entities, corresponding attributes, and attribute values are converted from RDF quads to tabular format and grouped by host. As this process involves converting a graph structure into single relational tables, which requires flattening the graph, the following decisions are made. If an attribute contains

[15]https://github.com/wbsg-uni-mannheim/schemaorg-tables
[16]https://github.com/apache/any23
[17]https://webdatacommons.org/structureddata/
[18]https://webdatacommons.org/structureddata/schemaorg/

child entities instead of a literal value, all child entities and their attributes are extracted as a list. However, only literal values are considered for the attributes of child entities, dismissing any child entity attributes further down the hierarchy. For example, a web page about a movie might annotate the name of the movie and details of the actors who appear in the movie, including their names and their spouses. In this case, the movie name and a list of actors are extracted. For each actor, only the actor's name is extracted because it is a literal value. Child entities further down the hierarchy, such as the actor's spouse, are omitted. After this step, the 2023 version of the table corpus contains ∼7.5 million tables with a total of ∼1.4 billion rows.

Removal of Listing Pages and Sparse Entities

Listing pages contain concise information about entities that are described in more detail on other pages. In order to have attribute-rich entity descriptions in the corpus, descriptions originating from listing pages must be excluded. Other pages provide detailed descriptions of one entity and brief descriptions of other entities as part of navigation elements or advertisements. The objective is to extract only the main entity from such pages. The following heuristic is applied to exclude these sparse entities. The entity is extracted if a web page contains only one relevant entity with at least three attributes. For web pages that contain multiple entities, the attribute values of each entity are concatenated, and the mean absolute deviation (MAD) of each entity is calculated based on the length of the concatenated attribute values. Entities with at least three attributes and concatenated attribute value lengths greater than the median plus three times the MAD (positive outliers) are extracted. If a web page marks up multiple entities without outliers, those entities are dismissed as originating from a listing page. Applying the heuristics reduces the corpus size to ∼5 million tables and ∼429 million rows.

Content-based Deduplication

Content-based deduplication removes exactly equal entity descriptions that originate from different web pages of the same host. This process is applied to all attributes except for *schema.org/url*, which is excluded for top and second-level attributes as it may differ and lead to false positives during deduplication. Only attributes with a density above 25% are kept for the final table for each host, and all sparse attributes are dismissed to avoid extremely sparse tables. After content-based deduplication, the 2023 release of the WDC Schema.org Table Corpus contains ∼5 million tables containing all together ∼361 million rows of data. It took 10 days of compute time on a local shared server equipped with 96 × 3.6 GHz CPU cores and 1024 GB RAM to create the 2023 release from the extracted RDF quads resulting from the first step.

3.4.2 Content of the 2023 Corpus

This section presents profiling information for the 2023 WDC Schema.org Table Corpus. The corpus comprises approximately 5 million tables, containing over 361 million rows of data in total. The tables cover 42 Schema.org classes and originate from over 4.33 million websites (hosts).

Tables by Class

The statistics on the number of tables per class, their rows, and the average number of attributes for a selection of Schema.org classes are presented in Table 3.1. The selected Schema.org classes demonstrate the breadth of the corpus, ranging from classes with extensive tables, such as *Product*, to those with fewer tables, such as *Dataset*. In addition to the statistics for the complete corpus (Overall), Table 3.1 provides separate statistics for the largest 100 tables and tables with at least three rows (Minimum 3). For example, the corpus contains over one million *LocalBusiness* tables describing altogether 8 million business entities with on average 4 attributes, such as *name*, *address*, *telephone*, or *average rating*. By distinguishing between the Top 100 and Minimum 3 tables, it is visible that the Top 100 tables account for 1% to 11% of all rows for the three most popular classes: *Product*, *LocalBusiness*, and *Event*.

Table 3.1: Number of tables and rows for selected Schema.org classes in millions (M) and thousands (k).

	Overall			Top100	Minimum 3	
Class	**Tables**	**Rows**	**Avg. Attr.**	**Rows**	**Tables**	**Rows**
Product	3M	288M	5	4M	2M	283M
LocalBusiness	1M	8M	4	903k	65k	6M
Event	368k	15M	8	1M	261k	14M
Restaurant	60k	716k	7	348k	7k	313k
JobPosting	58k	3M	7	543k	37k	3M
Recipe	41k	4M	11	619k	33k	3M
Question	38k	4M	6	2M	21k	2M
Hotel	22k	2M	6	794k	9k	880k
Book	14k	3M	6	944k	10k	2M
Movie	6k	2M	7	481k	5k	1M
SportsEvent	4k	579k	6	296k	3k	281k
Hospital	2k	40k	6	31k	396	7k
Dataset	2k	364k	7	286k	1k	77k

Attributes

The WDC Schema.org Table Corpus is constructed using Schema.org annotations. As a result, all tables share the same set of attributes, while the attributes present in a specific table depend on the annotations included by the corresponding host in its web pages. Table 3.2 shows a selection of attributes that appear in the tables for the classes *Product*, *LocalBusiness* and *Movie*. Common attributes such as *name* and *description* are present in many tables across multiple classes. Other attributes, such as *productID* and *genre*, are more class-specific and less frequently used, indicating that a long-tail distribution for such attributes exists in the corpus. For both head and long-tail attributes, the tables exhibit a high average value density of 95%. This shows that if hosts use a Schema.org term, they do so consistently. Some attributes are entity identifiers that can be used to link entities across tables, for example, to derive training data for entity matching tasks. Examples of such attributes are *SKU, productID, MPN* and *GTIN13* for the class *Product* as well as the *telephone* number for *LocalBusiness*.

Table 3.2: Fraction of tables containing specific attributes.

Product		**LocalBusiness**		**Movie**	
Attribute	**in % of tables**	**Attribute**	**in % of tables**	**Attribute**	**in % of tables**
name	100	name	99	name	96
offers	96	address	97	description	73
description	86	telephone	91	director	63
sku	57	aggregaterating	19	datecreated	55
brand	33	geo	16	aggregaterating	38
image	28	pricerange	12	duration	36
category	11	email	11	actor	36
aggregaterating	9	description	11	genre	26
productid	7	openinghoursspec	10	datepublished	25
mpn	7	url	8	image	15
gtin13	3	image	7	url	13

3.4.3 Comparison to Previous Table Corpora

Various table corpora have been created in recent years. Table 3.3 lists table corpora and shows statistics on their number of tables (Tabs.), the average number of rows (Avg. # Rows) and attributes (Avg. # Attr.), and whether all tables in the corpus use a single shared schema. The WDC Web Table Corpus [Lehmberg et al., 2016] and the Dresden Web Table corpus [Eberius et al., 2015] extract relational HTML tables from web pages in the Common Crawl. These web tables [Zhang and Balog, 2020] have been used in related work on table search [Chapman et al., 2020] and table augmentation [Cafarella et al., 2018]. The WikiTables table corpus contains tables that were extracted from Wikipedia [Bhagavatula et al., 2013]. VizNet [Hu et al., 2019] consists of tables that were chosen for benchmarking visualization methods. Open Data Portal Watch [Mitlöhner et al., 2016] contains tabular data that was collected from open data portals. Table 3.3 shows that the tables in the WDC Web Table Corpus, the Dresden Web Table Corpus, WikiTables, and VizNet have a relatively small number of rows. Compared to the web tables corpora, the WDC Schema.org table corpora and GitTables [Hulsebos et al., 2023] contain, on average, more rows. GitTables [Hulsebos et al., 2023] consists of tables that are extracted from CSV files shared on GitHub. The tables in all the referenced table corpora do not use a single shared schema, but each table uses a different, proprietary schema. As a post-processing step, the tables in GitTables are annotated with semantic types from DBpedia and Schema.org using an automated annotation method [Hulsebos et al., 2023].

Table 3.3: Related table corpora.

Table Corpus	Single Schema	Tabs.	Avg. # Rows	Avg. # Attr.
Dresden Web Tables Corpus [Eberius et al., 2015]	×	59M	17	6
WDC Web Tables Corpus 2015 [Lehmberg et al., 2016]	×	90M	14	5
WikiTables [Bhagavatula et al., 2013]	×	15M	15	6
VizNet [Hu et al., 2019]	×	31M	17	3
Open Data Portal Watch [Mitlöhner et al., 2016]	×	1M	17	14
GitTables [Hulsebos et al., 2023]	×	2M	209	25
WDC Schema.org Corpus 2020	✓	4M	89	5
WDC Schema.org Corpus 2023	✓	5M	72	5

3.4.4 Applications

This section describes various applications of the WDC Schema.org Table Corpora. The corpora can be used for benchmarking, as a source of training data, to analyze the adoption of semantic web technologies, and as a source of domain data. This following paragraphs give an overview of some possible applications.

Source of Training Data

Its structuredness, the common schema, and shared entity identifiers make the corpus a source of (pre-)training data for table representation learning as well as data integration. The corpus also contains 4 million question-answer pairs originating from 38,000 websites (see row *Question* in Table 3.2), which could be used to fine-tune large language models or as background knowledge for retrieval-augmented question answering. In addition, the corpus can also be used as a structured pre-training resource for table representation learning methods [Deng et al., 2020] or for fine-tuning large language models for structured data tasks [Li et al., 2023a].

Analyzing the Adoption of Semantic Web Technologies

Detailed statistics on which host uses which Schema.org terms are published together with the corpora. From a web science perspective, these statistics together with the data itself can be used to analyze the adoption of the Schema.org vocabulary within specific application domains, as well as on the Web in general [Guha et al., 2016, Brinkmann et al., 2023a].

Source of Domain Data

The corpus can further be used as a large source of domain data. For example, if a user wants to assemble a list of shops or hotels in a city, the 1 million local business tables in the corpus with 8 million rows in total could prove a useful starting point. If the user wants to analyze the skills that are currently in demand on the job market, they may use the 3 million job postings in the corpus for their analysis.

Benchmarking

The table corpus is a useful resource for evaluating table annotation, schema matching, entity matching, and data retrieval methods due to its shared schema and the presence of entity identifiers such as GTINs, MPNs, or phone numbers in many tables. For example, the SOTAB table annotation benchmark [Korini et al., 2022] was constructed by selecting a subset of tables from the 2020 version of the corpus, removing attribute labels from the tables, and having the annotation systems predict the removed labels. The SOTAB benchmark was used in the 2023 edition of the SemTab challenge[19]. A second benchmark that uses tables from the 2020 version of the corpus is the WDC Schema Matching Benchmark[20] that requires instance-based schema matching methods to discover correspondences between table columns.

The next part of the thesis presents an evaluation of using Schema.org annotations as supervision for automatically creating training sets for product matching and introduces two benchmarks based on this process.

[19]https://sem-tab-challenge.github.io/2023/
[20]https://webdatacommons.org/structureddata/smb/

Part II

The Semantic Web as Supervision for Product Matching

Chapter 4

Semantic Annotations as Training Data for Product Matching

4.1 Introduction

Deep neural networks have become part of the methods for all steps of the data integration pipeline, including schema matching [Deng et al., 2020, Wang et al., 2023], information extraction [Yang et al., 2022], and entity matching [Mudgal et al., 2018, Li et al., 2020, Wang et al., 2022c].

Most publicly available training corpora and benchmarks for entity matching are limited with respect to the amount and diversity of matching entity description pairs available for training [Peeters et al., 2024b]. This chapter evaluates the potential of using semantic annotations from a large number of websites as a source of training data for supervised product matching methods.

This chapter contributes a novel benchmark, WDC LSPM, and associated experimental analysis to the field of supervised entity matching, showing that manual labeling of training data for matching entities in the domain of products can be replaced by relying exclusively on Schema.org annotations gathered from the public Web. Many product offers on the Web are accompanied by product identifiers, such as GTINs [1] or EANs. A GTIN is a unique and internationally recognized identifier for products, with EANs being a type of GTIN with a specific length. Each GTIN is unique to a product and helps distinguish it from millions of other products on the global market. As these identifiers are marked up using Schema.org annotations as part of product offers, it is possible to cluster these offers by identifier. The result is a clustering of product offers from different e-shops referring to the same product, which can subsequently be leveraged to create training pairs for product matching methods automatically. The clustering is created from Schema.org annotations found in the WDC extraction of the Common Crawl (see Section 3.4). The specific clustering used in this chapter consists of more than 20 million pairs of offers referring to the same products. The offers were extracted from 43 thou-

[1] https://www.gs1.org/standards/id-keys/gtin

sand e-shops that provide Schema.org annotations, including some form of product identifiers, such as manufacturer part numbers (MPNs), global trade item numbers (GTINs), or stock-keeping units (SKUs). At the time of creation, the dataset is orders of magnitude larger than the training sets of the publicly available datasets Walmart-Amazon [Gokhale et al., 2014], Amazon-Google [Köpcke et al., 2010], and Abt-Buy [Köpcke et al., 2010] that are widely used to evaluate entity matching methods (also see Table 4.1 and Section 5.4).

This chapter introduces the process for automatically generating a large-scale training set for product matching from Schema.org annotations as the basis for the generation of the WDC LSPM benchmark. Afterwards, it evaluates the usefulness of the WDC LSPM benchmark by using it to train non-neural and neural product matchers to a high level of performance using a range of experiments.

Using Schema.org product identifier annotations, such as GTINs or MPNs, as distant supervision for product matching carries a certain level of inherent noise. Therefore, it is important to explore the learning method's label noise resistance. A specialized set of experiments shows that the Deepmatcher [Mudgal et al., 2018] system can be reliably trained with the inherent noise of the automatically created training set, but quickly loses performance when the level of label noise is further increased.

Finally, millions of additional training pairs from the clustering are used to perform large-scale intermediate training of the PLM BERT [Devlin et al., 2019], leading to increased performance on the WDC LSPM benchmark when compared to fine-tuning on thousands of training pairs with the state-of-the-art method Ditto [Li et al., 2020], further supporting the usefulness of Schema.org annotations as a source of training data also on million scale. These experiments build the foundation for the WDC Products benchmark presented in Chapter 5 and all the following chapters, since the created benchmarks are used as part of all the following experimental evaluations.

The contributions of this chapter are summarized as follows:

- **WDC Product Data Corpus and Gold Standard for Large-Scale Product Matching (WDC LSPM):** A highly textual product matching benchmark derived from Schema.org annotations in the Common Crawl of November 2017 using annotated product identifiers on webpages to cluster product offers referring to the same product. The benchmark provides training, validation, and test splits for four product categories. This benchmark compliments publicly available product matching benchmarks due to its large size and multi-source nature, resulting in more available and heterogeneous product descriptions per product compared to previous publicly available benchmarks.

- **Evaluation of WDC LSPM to Train and Maintain Product Matchers:** An analysis of the usefulness of the created benchmark for training various neural and non-neural matching methods is performed. This evaluation includes an investigation of the impact of label noise on the matchers and the possibility of maintaining matchers for unseen entities and their performance on such entities. The results show that neural product matchers trained and maintained with the Schema.org data in the WDC LSPM benchmark can achieve a high performance of more than 90% F1 in all product categories covered by the benchmark and are robust to the average label noise inherent to the automatically generated pairs.

- **Evaluation of Schema.org Training Data for the Large-Scale Intermediate Training of BERT:** The usefulness of the LSPM benchmark and the original clustering of product offers is further supported by subjecting the PLM BERT [Devlin et al., 2019] to large-scale intermediate training with millions of automatically labeled training examples, resulting in improved performance compared to fine-tuning with only thousands of examples. The additional inclusion of a masked language modeling (MLM) objective together with the binary matching objective in a multi-task training setup with millions of automatically created training pairs results in further improvements to matching performance.

The first and second contributions are joint work with Anna Primpeli and Benedikt Wichtlhuber published in [Peeters et al., 2020b]. The author contributed the LSPM benchmark and all entity matching experiments apart from those described in Section 4.7.2 which were performed by Benedikt Wichtlhuber. The process presented in Section 4.4 and the resulting corpus in Section 4.5 are the work of Anna Primpeli. They are included in this thesis as they are the basis for the creation of both presented benchmarks. The third contribution [Peeters et al., 2020a] is joint work with Goran Glavaš, who supported the author with best practices for the BERT Transformer architecture and its intermediate training.

Section 4.2 presents related work on the properties of publicly available benchmarks for entity matching and the usage of semantic annotations on the Web. Section 4.3 discusses the semantic annotations found in the Common Crawl of November 2018 which is the basis for the following clustering process. Section 4.4 presents the process of exploiting Schema.org annotations to collect large amounts of training data for entity matching automatically. Section 4.5 presents the WDC Training Dataset for Large-Scale Product Matching as the result of this process. Section 4.6 presents the process of building the WDC LSPM benchmark from this dataset. Section 4.7 presents the experimental results for the evaluation of the Schema.org generated training data and the WDC LSPM benchmark. Section 4.8 concludes the chapter with a discussion of the results and contributions.

4.2 Related Work

The first part of this section gives an overview of the properties of existing entity matching benchmarks showing the need for larger and more diverse benchmarks. The second part of this section presents various studies on the adoption of markup languages that have been performed, showing a steady increase over the years. This increased adoption of semantic annotations makes the creation of the corpus and benchmarks presented in the following sections and chapters a possibility.

Properties of Existing Entity Matching Benchmarks and Corpora:

Various benchmark datasets [Primpeli and Bizer, 2020] have been proposed and used to compare entity matching methods since the inception of the field. Due to the importance of benchmarks for evaluating entity matching systems and driving the entity matching field forward, some researchers have focused on analyzing and categorizing publicly available benchmarks along various dimensions [Primpeli and Bizer, 2020,Graf et al., 2022]. Table 4.1 gives an overview of publicly available entity matching benchmark datasets and their statistics, including: domain, number of sources from which the data originates, number of entities/records, and number of positive pairs (record pairs referring to the same entity).

The Alaska and Ember benchmarks are very recent additions to the set of entity matching benchmarks and were chronologically proposed after the work presented in this chapter. The remaining benchmarks in the table have become a gold standard for evaluating entity matching systems for many years and are widely used in the literature [Primpeli and Bizer, 2020, Barlaug and Gulla, 2021].

When comparing the statistics, it is evident that these benchmarks originate from a limited number of sources, usually two, and are limited by the amount of matches (per entity) that are available for training and evaluating machine learning systems. These properties lead to limited heterogeneity among the entity records and hinder learning entity-specific patterns due to the low amount of available matches per entity. These circumstances prompt the need for corpora and benchmarks with larger amounts of training data, especially for training neural network-based entity matching systems that usually require a large and diverse amount of training data to perform at peak effectiveness [Mudgal et al., 2018, Li et al., 2020, Peeters et al., 2024b].

As the process of labeling record pairs as positives or negatives is highly labor intensive and costly [Gokhale et al., 2014] at scale, but at the same time considered a very mundane task, this has prompted research into using automatically collected training data from semantic annotations, as presented in this chapter. Section 5.2 presents a more in-depth discussion of the benchmarks presented in Table 4.1 and compares them to the two benchmarks proposed in this thesis.

Table 4.1: Comparison of existing benchmark datasets and corpora for entity matching.

Benchmark	Domain	# Sources	# Entities	# Records	# Matches	# Non-Matches	Avg. Matches per Entity
			Leipzig Database Group				
Abt-Buy	Product	2	1,012	1,081/1,092	1,095	-	1.08
Amazon-Google	Product	2	995	1,363/3,226	1,298	-	1.30
DBLP-ACM	Bibliogr.	2	2,220	2,614/2,294	2,223	-	1.00
DBLP-Scholar	Bibliogr.	2	2,351	2,616/64,263	5,346	-	2.27
			DuDe				
Restaurants	Company	2	110	533/331	112	-	1.02
Cora	Bibliogr.	1	118	1,879	64,578	268,082	547.27
			Magellan				
Walmart-Amazon	Product	2	846	2,554/22,074	1,154	-	1.36
Company	Company	2	28,200	28,200/28,200	28,200	84,432	1.00
Beer	Product	2	68	4,345/3,000	68	382	1.00
iTunes-Amazon	Product	2	120	6,906/55,932	132	407	1.10
			Alaska				
Camera	Product	24	103	3,865	157,157	-	1,525.80
Monitor	Product	26	242	2,283	13,556	-	56.02
			Chinese Academy of Sciences				
Ember (Chinese)	Product	1	350	6,245	5,053	206,296	14.44

Usage of Semantic Annotations on the Web:

The major search engine companies Google, Yahoo!, Microsoft, and Yandex extract structured data from their web crawls, but do not make the data publicly available for commercial reasons. However, there have been several studies on the adoption of markup languages:

Mika and Potter [Mika and Potter, 2012] analyze the adoption of markup languages based on Web crawls of the Bing search engine from 2011 and 2012. They state that search engines at that time were biased towards popular head sites with textual content and were not specifically looking for marked up data in webpages. They report an adoption of over 30% for the websites in their sample that use some form of semantic annotation, with RDFa and Microdata being the most preferred choices of syntax.

Meusel et al. [Meusel et al., 2015] studied the adoption of the Schema.org vocabulary during the period 2012 to 2014 using early extractions from the Web Data Commons (WDC) project[2]. They find that half of the elements in the Schema.org standard are not observed in any part of the examined corpora, while deprecations in the standard have often found widespread adoption. The authors further state that homogeneity in the adoption of the Schema.org vocabulary increases the more relevant the marked-up data is for large search engines like Google due to their usage in rich search snippets. Another reason for the increased homogeneity is

[2]https://webdatacommons.org/

observed in the usage of parts of the vocabulary in widely deployed content management systems.

Guha et al. [Guha et al., 2016] present an updated analysis of Microdata adoption with a particular focus on the Schema.org vocabulary. They summarize their findings in three lessons learned: (1) if asymmetry exists between publishers and consumers, the complexity should be put with the smaller number as to continue using their learned workflows, (2) long specifications are not followed as developers tend to copy and edit examples instead, and (3) complexity should be added over time as needed as adoption is driven by basic layers first.

Kanza et al. [Kanza et al., 2018] analyze the social processes around the creation and maintenance of the Schema.org vocabulary. They find that GitHub is the main platform for creating and editing functionality, while the main use of the community mailing list is for clarifications. The authors further state that 10% of the users perform 80% of the work, with most users being "lurkers" that do not start discussions or interact with the group.

Alrashed et al. [Alrashed et al., 2021] find that 61% of the hosts in a Google crawl that provides *Schema.org/Dataset* annotations do not describe datasets and propose a classifier to identify such low-quality annotations. They state that the providers often do not use the correct typing for artifacts on a webpage if the Schema.org category is more general, as is the case for the dataset category.

Finally, the Web Almanac[3] has tracked the adoption of semantic annotations on the Web based on a crawl by the HTTP archive. The aforementioned WDC project[4] performs a yearly extraction of the Common Crawl and examines the adoption of semantic annotations each year[5], and makes the data available as raw annotations on HTML pages and in various other formats to interested users. This effort was presented in 2014 by Meusel et al. [Meusel et al., 2014] and recently by Brinkmann et al. [Brinkmann et al., 2023a] who found that the adoption of Schema.org terms has risen to 50% of all webpages in the Common Crawl over the years. This widespread adoption makes the use of Schema.org annotations to automatically create training data a lucrative prospect.

The clustering of products based on Schema.org annotations and the WDC LSPM benchmark for entity matching used in this chapter, as well as the WDC Products benchmark presented in Chapter 5, are all based on WDC extractions, which will be named at the corresponding locations in this thesis. The data from two of these extractions is also available in a relational table format as previously presented in Section 3.4.

[3]https://almanac.httparchive.org/en/2022/structured-data
[4]https://webdatacommons.org/
[5]https://webdatacommons.org/structureddata/

4.3 Schema.org Annotated Product Offers in the Common Crawl

The Web Data Commons (WDC) project[6] monitors the adoption of Schema.org annotations on the Web by analyzing the Common Crawl[7], a series of public Web corpora, each containing several billion HTML pages [Meusel et al., 2014, Brinkmann et al., 2023a]. Table 4.2 shows the number of pay level domains (PLDs) in the CommonCrawl that use product-related Schema.org terms in 2017 compared to 2013. The absolute number of websites, the richness of the descriptions, and the number of websites annotating product identifiers (lower part of the table) have all grown significantly. The November 2018 version of the Common Crawl contains 2.5 billion pages originating from 32.8 million pay-level domains (PLDs). Examples of pay-level domains are, for instance, *amazon.de* or *ebay.co.uk*. Of these PLDs, 9.6 million use semantic annotations (29.3%).

Table 4.2: Adoption of product-related Schema.org properties. The percentages refer to all websites using Schema.org Product markup [Primpeli et al., 2019].

Property	#PLDs 2013	#PLDs 2017	%PLDs 2013	%PLDs 2017
s:Product/name	50,536	535,625	89.62%	92.11%
s:Offer/price	33,509	462,444	59.42%	79.53%
s:Product/offers	33,090	462,233	58.68%	79.49%
s:Offer/priceCurrency	14,704	430,556	26.08%	74.04%
s:Product/image	34,921	419,391	61.93%	72.11%
s:Product/description	38,037	377,639	67.46%	64.94%
s:Offer/availability	21,789	337,876	38.64%	58.11%
s:Product/url	11,937	263,720	21.17%	45.35%
s:Product/brand	5,880	73,934	10.43%	12.71%
s:Product/productID	7,392	35,211	10.90%	6.05%
s:Product/sku	1,323	126,696	1.95%	21.78%
s:Product/mpn	484	8,161	0.71%	1.40%
s:Product/gtin13	276	5,467	0.41%	0.94%
s:Product/identifier	160	538	0.24%	0.09%
s:Product/gtin8	0	257	0.00%	0.04%
s:Product/gtin12	0	577	0.00%	0.09%
s:Product/gtin14	0	722	0.00%	0.12%

Table 4.3 gives an overview of the most frequently offered types of data (in the form of Schema.org classes). The table distinguishes between the two most widely used annotation syntaxes. The Microdata syntax for annotating data in the *BODY* of HTML pages and the JSON-LD syntax which is used to embed data into the *HEAD* section of HTML pages. Around 850 thousand websites provide prod-

[6]https://www.webdatacommons.org/structureddata/
[7]https://commoncrawl.org/

Table 4.3: Number of websites (PLDs) offering specific types of data [Bizer et al., 2019].

Schema.org class	Microdata	JSON-LD
WebPage	1,124,583	121,393
Product	812,205	40,169
Offer	676,899	57,756
Article	612,361	57,082
Organization	510,069	1,349,775
PostalAddress	502,615	178,500
ImageObject	360,875	111,946
BreadcrumbList	344,538	205,971
ListItem	338,845	209,207
Blog	337,843	12,174
BlogPosting	327,828	43,243
Person	324,349	335,784
LocalBusiness	294,390	249,017
AggregateRating	258,078	23,105
WebSite	158,054	3,519,466
Review	124,022	6,622
Place	92,127	66,396
Event	88,130	63,605
Brand	65,835	11,439

uct data using the Schema.org vocabulary. The product properties that are most widely used are *name*, *description*, *brand*, and *image*. Interestingly and very crucial for using semantic annotations from different websites to train matching methods, 30.5% of the websites annotate product identifiers, such as MPNs, GTINs, or SKUs, which allow offers for the same products originating from different websites to be clustered.

4.4 Cleansing and Clustering Schema.org Product Data

Webmasters maintaining semantic annotations in the templates used to render HTML pages have different levels of knowledge and understanding of the Schema.org vocabulary. Schema.org terms are not used consistently and according to the specifications on all sites [Meusel et al., 2015]. As a result, semantic annotations must be cleansed before they can be used for training entity matching systems. The following paragraphs describe the pipeline of cleansing operations that are applied to create a large training dataset for products from Schema.org annotations. The Web Data Commons product corpus version November 2017[8] is the starting point for the creation of the training set. The corpus contains 809 million *schema:Product* and *schema:Offer* entities originating from 581,482 websites.

[8]https://www.webdatacommons.org/structureddata/2017-12/stats/schema_org_subsets.html

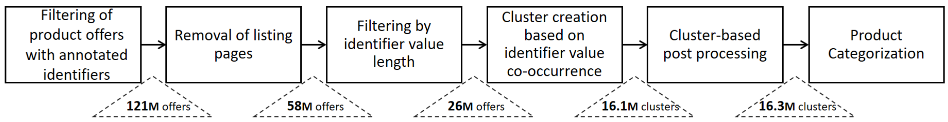


Figure 4.1: WDC Product Data Corpus creation pipeline [Peeters et al., 2020b].

In the first step, the subset of the product offers that provide some kind of product identifier, such as MPNs, GTINs, or SKUs, is selected. Afterwards, the selected offers are grouped into ID-clusters based on these identifiers. The resulting clusters are further cleansed of abnormalities. Figure 4.1 gives an overview of the data cleansing pipeline that is performed to group offers into clusters. The following paragraphs describe each of the cleansing steps. The cleansing and clustering pipeline is the work of Anna Primpeli and was presented in [Primpeli et al., 2019]. The process is presented in this thesis as it is the basis of the following experimental evaluation and the benchmarks created and presented in this thesis.

Filtering of Product Offers with Annotated Identifiers

All products and offers that include a globally unique identifier are gathered from the WDC product corpus, so they can later be clustered using these identifiers. When manually examining a subset of the annotations, it is noticeable that many websites annotate globally scoped identifiers, such as GTIN or MPN, using the vendor scoped term *sku* (stock keeping unit) or the generic terms *identifier* and *productID* as shown in Table 4.4. As a result, all offers that contain any identifier-related term (e.g. *gtin8, gtin12, gtin13, gtin14, mpn, sku, identifier* and *productID*) are considered and vendor-specific identifiers are filtered later in the cleansing process. Similarly to the observations of [Meusel et al., 2014] and [Kärle et al., 2016], 6% of the websites annotating product offers have syntax errors in the URIs identifying Schema.org terms or use deprecated or undefined terms. In an effort to make use of these offers as well, all entities that have at least one property with an identity revealing suffix[9] are included. Using this selection strategy, 116 million of the 809 million offers (14%) in the Web Data Commons product corpus are identified to contain some sort of product identifier.

Detection and Removal of Listing Pages and Advertisements

The goal is to include the comprehensive description of a product from its detail page in the final dataset, not the summary of this information often found on listing pages and in advertisements on detail pages of other products. However, identifiers in listing items and advertisements are also annotated, making it neces-

[9] Regex applied to each predicate URI for capturing identity revealing properties:
$.*/(gtin8|gtin12|gtin13|gtin14|sku|mpn|identifier|$
$productID)$

Table 4.4: Value overlap of different ID properties [Primpeli et al., 2019].

Value overlap	gtin8	gtin12	gtin13	gtin14
sku	2,065	18,682	103,736	16,897
productID	2,966	13,854	198,847	10,192
identifier	122	3,751	17,732	837
Overlap #	5,153	36,287	320,315	27,926
Overlap %	6.75%	15.81%	11.21%	11.35%

sary to detect those entities and remove them from the corpus. For the detection of listing pages and advertisements, a heuristic is applied that relies on the following features: amount of *schema.org/Offer* and *schema.org/Product* entities per web page, variation in the length of product descriptions, number of identifier values, and semantic connection to parent entities using the terms *schema:relatedTo* and *schema:similarTo*. This heuristic for identifying listing pages and advertisements achieves an F1 score of 94.8% on a manually annotated test set. This cleansing step removes 49% of the offer entities, leaving 58 million non-listing and non-advertisement offers in the dataset.

Filtering by Identifier Value Length

In the next step, the identifier values are normalized by removing non-alphanumeric characters and common prefixes such as initial zero digits and identifier-related strings like *ean, mpn, sku,* and *isbn*. Considering the length of global identifiers such as GTIN or ISBN numbers in comparison to vendor-specific identifiers that are often relatively short, all offers having identifiers that are shorter than 8 characters are filtered out. Additionally, offers whose id values completely consist of alphabetical characters are removed. Finally, it can be observed that a considerable number of websites use the same identifier value to annotate all their offers, likely due to an error in the script generating the pages. These websites and their offer are removed from the dataset. After applying both filtering steps, 26 million product offers remain in the dataset.

Cluster Creation

The remaining 26 million offers are grouped into 18 million clusters using their identifier values. It can happen that single offers contain multiple alternative identifiers referring to the same product, e.g., GTIN8 and GTIN12 or GTIN12 and MPN. This information is used to propagate connections across clusters and finally merge clusters referring to the same product, resulting in a reduction of the number of clusters to 16 million. 13 million of these clusters contain only a single offer. Some websites include identifiers that refer to product categories, such as UNSPSC numbers[10], in addition to identifiers referring to single products in the annotations. To detect such cases, the structure of the identifier co-occurrence graph within each cluster is examined. Vertices with a degree greater than 10 and a clustering coefficient of $C_i < 0.2$ tend to represent product categories rather than single products, and the clusters are split accordingly. This leads to the creation of 199,139 additional clusters. In the final step, an English language-focused subset of the clustering is created by including only offers from the top-level domains *com*, *net*, *co.uk*, and *org*.

Offer Categorization

The Schema.org vocabulary contains terms for annotating an offer's product category. However, less than 2% of the offer pages in the WDC 2017 corpus annotate category information. The ID clusters are instead categorized into a flat product categorization schema suggested by merging categories of a publicly available Amazon product dataset[11], as well as the Google[12] and UNSPSC[13] taxonomies. For this purpose, a one-vs-rest ensemble of logistic regression classifiers is trained to categorize product offers. For training and testing, 2,115 clusters are manually annotated while considering an annotation limit of at least 50 clusters per product category. The ensemble achieves 85% $F1_{micro}$ on the test set over all categories. The learned model is applied to the English subset and a product category is assigned to each ID cluster. Table 4.5 presents the distribution of clusters, offers, and cluster group sizes per category.

4.5 The WDC Training Dataset for Large-Scale Product Matching

Applying the cleansing procedure described above to the Web Data Commons product corpus of November 2017 results in a final dataset comprising 26 million offers originating from 79 thousand websites. The dataset is named *WDC Training Dataset for Large-Scale Product Matching*. Using the identifiers, the product offers are grouped into 16 million clusters referring to the same products. 13 million of these have a size of one, 1.9 have a size of two, and 1.1 million have a size larger than two. The English language subset consists of 16 million offers grouped into 10 million clusters. Of these clusters, 8.4 million have a size of one, 1 million have a size of two, and 625.7 thousand have a size larger than two. Only considering clusters in the English subset having a size larger than five and excluding clusters of sizes bigger than 80 offers, 20.7 million positive training examples (pairs of

[10] http://www.unspsc.org/
[11] http://jmcauley.ucsd.edu/data/amazon/
[12] https://www.google.com/basepages/producttype/taxonomy.en-US.txt
[13] https://wwwcfprd.doa.louisiana.gov/osp/lapac/vendor/commodityTree.cfm

Table 4.5: Distribution of clusters, offers, and cluster group sizes per product category in the English WDC Product Data Corpus [Peeters et al., 2020b].

Category	% Clusters	% Offers	# Clusters of Size			
			[3-4]	[5-10]	[11-20]	[>20]
Tools_and_Home	12.37	9.41	27,228	8,336	4,648	1,325
Home_and_Garden	10.59	8.89	33,812	16,873	5,364	1,997
Automotive	8.95	8.37	22,808	10,143	4,094	4,421
Other_Electronics	6.74	5.27	21,590	5,603	1,155	508
Clothing	5.93	6.67	40,767	32,915	3,177	1,840
Books	5.36	4.22	19,771	6,361	654	360
Jewelry	4.94	4.95	19,526	11,245	2,112	1,436
Shoes	4.42	4.15	27,948	9,080	1,468	381
Health_and_Beauty	4.23	3.93	22,852	7,429	1,302	537
Toys_and_Games	4.01	3.21	11,196	3,109	968	521
Office_Products	4.01	13.13	27,393	10,875	3,052	1,561
Sports	3.84	5.45	16,930	6,927	1,556	2,602
Grocery	3.51	3.01	20,428	3,416	416	243
Computers	3.03	4.04	12,917	6,694	3,257	4,184
Travel	2.63	2.68	21,871	7,914	704	385
CDs_and_Vinyl	2.49	1.80	4,251	1,514	486	71
Camera_and_Photo	2.38	1.97	11,583	3,851	663	134
Musical_Instruments	2.26	2.01	6,218	1,176	240	180
Cellphones	2.03	1.52	3,652	1,334	274	281
Others	6.28	5.33	27,781	8,561	1,972	1,378

matching product offers) and a maximum of 2.6 trillion negative training examples can be automatically derived from the dataset.

Table 4.5 shows the distribution of offers per product category, as well as the cluster size distribution in the English training set. Taking into account only clusters that have a size greater than two, more than 1.2 million positive pairs for the clusters in the categories *Office Product* and *Clothing*, and more than 300 thousand pairs each for the categories *Shoes, Camera and Photo, Cell Phone and Accessories, Computers and Accessories,* and *Jewelry* can be derived. The available training data that can be derived for each category is larger than the datasets that have been available to the public so far. Furthermore, the combined numbers of pairs are several orders of magnitude higher than the amount of training examples available in the benchmark datasets in Table 4.1 in Section 4.2.

Attribute Extraction

From the set of all clustered offers, the descriptive properties of the offers that are annotated with Schema.org terms are extracted. Table 4.6 shows the distribution of descriptive Schema.org properties in the full and English datasets, as well as the

Table 4.6: Amount of offers in the training set having specific properties [Bizer et al., 2019].

Property	Offers Full Set		Offers English Set	
	#	%	#	%
s:name	25,281,317	95.37%	15,653,878	95.15%
s:description	17,215,475	64.94%	11,352,319	69.00%
s:brand	9,313,258	35.13%	5,645,282	34.31%
s:image	5,785,250	21.82%	4,348,830	26.43%
s:price	3,335,306	12.58%	1,977,269	12.01%
s:priceCurrency	2,971,417	11.20%	1,873,611	11.38%
s:availability	1,180,268	4.45%	716,066	4.35%
s:manufacturer	2,024,537	7.63%	1,254,601	7.62%
s:sku	11,475,859	43.29%	7,239,039	44.00%
s:mpn	4,611,908	17.39%	3,167,895	19.25%
s:productID	9,386,433	35.41%	6,351,147	38.60%
s:gtin8	452,151	1.70%	167,411	1.01%
s:gtin13	3,529,958	13.31%	1,449,759	8.81%
s:gtin12	300,102	1.13%	261,431	1.58%
s:gtin14	420,712	1.58%	71,428	0.43%
s:identifier	179,225	0.67%	65,736	0.39%

distribution of identifier-related Schema.org properties in both sets. The density of descriptive properties beside *name* and *description* is low ($< 50\%$). This is in line with previous findings [Meusel et al., 2014] that only a small subset of the Schema.org vocabulary is widely used on the Web. More than 75% of the identifiers that were used for the clustering were annotated using the terms *sku* and *productID*, which justifies the decision not to ignore these, in theory, vendor-specific properties but also to consider their values in the cleansing process.

In addition to the annotated properties, the product specifications are extracted from HTML tables that are included in the detail pages in the form of key/value pairs. The method described in the works of Petrovski et al. [Petrovski et al., 2017] and Qiu et al. [Qiu et al., 2015] is applied for this purpose. The method detects specification tables for 24% of the offers contained in the full set and 17% of the offers in the English set.

Quality of the Clustering

In order to evaluate the quality of the clustering of product identifiers, 900 pairs of offers belonging to the same clusters are randomly sampled, and it is manually verified by two human domain experts if the offers do, in fact, refer to the same product by inspecting the *name* and *description* values of the offers. 93.4% of the sampled pairs are considered to be labeled correctly. 2.1% of the pairs in the sample (19 out of 900) are erroneous due to the web pages providing incorrect identifier values. 1.0% of the pairs in the sample (9 out of 900) are wrong due to errors introduced by the transitive grouping strategy, which merges two clusters if a single offer is found that is annotated with the identifier values of both clusters (e.g., a GTIN8 and an MPN number). For 3.4% of the sample (31 offer pairs), the two annotators were unable to decide whether the two offers referred to the same product because the names and descriptions were too short (e.g., just *"Samsung Galaxy"*) or too general (e.g. *"computer software"*). In 20 of these 31 cases, the name and description together contained less than four tokens. These pairs can be easily deleted from the dataset using a length filter if desired.

The experiments described in Section 4.7.3 show that the observed amount of errors is acceptable to use the identifiers to automatically generate training pairs and train effective product matching systems with them.

4.6 The WDC Product Data Corpus and Gold Standard for Large-Scale Product Matching

This Section gives an overview of the creation process and the profile of the test and development sets that make up the WDC LSPM benchmark. The previous sections presented the process for generating a large clustering of product offers using product identifiers annotated using Schema.org annotations from the Web. The LSPM benchmark presented in this section leverages this clustering, specifically the English WDC Training dataset, to automatically create development sets and semi-automatically create test sets for the benchmark.

4.6.1 Creation Process

Test Sets

The LSPM benchmark test sets consist of 1100 pairs of product offers for each of the following four product categories: *Computers & Accessories*, *Camera & Photo*, *Watches* and *Shoes*. Each test set covers 150 products (ID-clusters from the English WDC Training Dataset) from each category with positive and negative pairs. Overall, offers for more than these 150 products are contained in the test sets, serving only as negative correspondences to the 150 products without positives of their own. These are contained for the purpose of generating negative corner cases for the 150 products in an effort to create a challenging benchmark. For each of the 150 products the test set contains two matching pairs of offers (positives) and five or six non-matching pairs of offers (negatives). The aim was to include a diverse selection of products as well as a good mixture of difficult and easy-to-match pairs of offers into the test sets. To this end, similarity metrics are applied to different attributes to sort pairs and collect corner cases (positives and negatives) for the test set, while also including randomly selected offer pairs. The clustering of semantically annotated identifier values in the English WDC Training Dataset serves as distant supervision, separating matches (intra-cluster) from non-matches (inter-cluster). A domain expert manually reviewed all pairs of offers in the test set to ensure that the label was correct and replaced them if erroneous labels were discovered.

To build each test set, 150 ID-Clusters are randomly selected from the English WDC Training Dataset per category. For each of the 150 products, a good mixture of difficult and easy-to-match positive and negative pairs of offers in the test set is essential to represent a challenging real-world matching use-case and avoid bias in pair selection. The subsequently presented process represents a simulated blocking step as it may be performed in a real-world scenario. To determine which pairs to include, the following heuristic is applied.

For each product (ID-cluster), two positive pairs are selected from within a cluster. For this purpose, all possible pairs within a cluster are ordered using the Jaccard similarity of the titles, the Jaccard similarity of the descriptions, or the average of both (the selected metric is chosen randomly). One corner case pair is selected by selecting the most dissimilar pair and choosing the second positive randomly from the list.

For each cluster, one offer is selected to create negatives. Subsequently, all possible negative pairs with all offers of the same category are built. After applying one of the previously mentioned similarities at random, the resulting list of pairs is ordered, and two to three negative pairs with a high similarity and three randomly chosen negative pairs are selected from that list. A domain expert verifies that the selected pairs are correctly labeled as matches or non-matches by reading the titles and descriptions of the offers. Since negatives are not only selected among the 150 clusters but from the whole English WDC Training Dataset, technically, more than 150 products are represented in the test set as stated above.

An additional test set was created for the computers category as part of the MWPD2020 Semantic Web Challenge on Mining the Web of HTML-embedded Product Data [Zhang et al., 2020]. The test set was used for evaluating and ranking the matching systems of the participants of this challenge. A categorization scheme consisting of five specific types of matching challenges was designed to derive the pairs in the MWPD test set. The basis for its creation are the pairs in the computer test set that are characterized as *seen corner cases*, as their selection, as described above, is based on similarity metrics resulting in corner cases and different offers for the same product being contained in the development as well as test set resulting in a seen scenario. From these pairs, 200 positives are selected, with one hundred of those having typos introduced and the other hundred having words dropped manually by two domain experts. The resulting augmented pairs are added to the MWPD test set.

Finally, the similarity calculations used in the selection process for positives and negatives of the computers test set are leveraged to select 25 additional products (ID-clusters) that are similar to the existing ones in the computers test set. Similarly, 25 products are selected that have a low similarity. This selection is done manually by a domain expert from the respective sorted similarity lists. For each of these sets of 25 products, a single positive and three negatives are created using the same process as described before. Note that negative pairs are not created with the clusters already contained in the computers test set. The resulting test set contains pairs for five matching challenges: (1) corner cases in a seen scenario, (2) positive pairs with typos in a seen scenario, (3) positive pairs with dropped words in a seen scenario, (4) an unseen scenario with products similar to the seen products and, (5) an unseen scenario with products with low similarity to the seen products.

Development Sets

Development sets of four different sizes are created for the four product categories and subsequently split into training and validation sets. To build development sets, the ID-clusters of all products in the already created respective test set are used to automatically build positive pairs inside and negative pairs across clusters using a heuristic similar to the one used for building the test sets. These pairs are not manually checked for the correctness of the label. An evaluation of expected noise and its impact on training is presented in Section 4.7.3. The process of building development sets is presented in detail in the following.

For each cluster, all combinations of offers within the cluster are created and from this group x positive pairs are randomly sampled. x is used as a scaling factor to vary the number of positive pairs sampled per product. The maximum amount possible is sampled if the group contains fewer than x pairs. For each sampled pair, it is assured that it does not appear in the corresponding test set.

To build negative pairs, the similarity of the title of all the pairs of clusters from the test set is calculated by using the Jaccard similarity over the concatenated titles of all the offers within a cluster. The top ten most similar clusters based on this similarity are chosen for each cluster. Considering the resulting ten pairs of clusters, the smallest of the ten similar clusters is selected, and the other nine are randomly down-sampled to the same size. Then all possible pairs between each combination of the ten similar clusters with the current cluster are built. From the resulting group, y negative pairs are randomly sampled. y is a scaling factor for the negative pairs. The maximum amount possible is sampled if the group contains fewer than y pairs. If, during the selection of the ten most similar clusters, negative pairs between these clusters already exist due to prior sampling for the other cluster, this cluster combination does not contribute further pairs to the group. This procedure for sampling negative pairs ensures that the offers in a negative pair are either corner cases or at least similar, simulating a blocking process and resulting in pairs with interesting patterns for training entity matching systems.

Using varying scaling factors x and y, it is possible to easily create development sets of different sizes. A fixed positive to negative ratio of 1: 3 for x:y is chosen and four development sets of increasing size are built for the following values: 1:3(small), 3:9(medium), 15:45(large) and 50:150(xlarge).

Table 4.7: WDC LSPM test set profiling.

Category	#Products	# Pos. Pairs	# Neg. Pairs	% Density T	D	B	S
Computers	150(745)	300	800	100	82	42	22
Cameras	150(563)	300	800	100	73	25	7
Watches	150(617)	300	800	100	71	15	7
Shoes	150(563)	300	800	100	70	8	2
All	600(2485)	1,200	3,200	100	74	23	10

Table 4.8: Distribution of offer pairs per type of matching challenge in the MWPD test set.

Matching challenge	#Match	#Non-Match
(SN-DM) Similar non-matches, dissimilar matches	275	825
(NP-HS) New products - high similarity to known products	25	75
(NP-LS) New products - low similarity to known products	25	75
(KP-TY) Known products - introduced typos	100	0
(KP-DR) Known products - dropped tokens	100	0

4.6.2 Benchmark Profiling

Table 4.7 shows statistics for the WDC LSPM benchmark test sets, including the percentage of pairs per category, where both offers contain a value for the respective attribute. The test sets contain 300 positive and 800 negative pairs of offers from each category. The title attribute of all of these offers is filled. The other attributes describing the offers might contain NULL values and thus have a lower density. Altogether, the test sets consist of offers for 2485 products (ID-clusters) across all categories. These offers are distributed as follows over the product categories: 745 for Computers, 563 for Cameras, 617 for Watches, and 563 for Shoes. There are positive pairs for 150 of these products per category. The remaining products are contained as part of offers in the negative pairs, as previously described in Section 4.6. The MWPD test set contains in total 1,500 product pairs distributed over five matching challenges. The distribution of offer pairs by type of matching challenge in the MWPD test set is shown in Table 4.8.

Table 4.9 shows the statistics for the size and attribute density of the development sets. The sizes of the development sets range from ~2K offer pairs for the small development sets to ~20K pairs for the xlarge versions. The four training sizes allow for an in-depth evaluation of matching systems on the dimension of available labeled development data. Each development set contains offer pairs for the same products that are contained in the test sets. As a result, the WDC LSPM benchmark represents a *fully seen* scenario.

Table 4.9: WDC LSPM development set profiling.

Category	# Pos. Pairs	# Neg. Pairs	% Density T	D	B	S
Small						
Computers	722	2,112	100	51	34	21
Cameras	486	1,400	100	53	21	4
Watches	580	1,675	100	43	15	7
Shoes	530	1,533	100	49	8	0
All	2,318	6,720	100	49	21	10
Medium						
Computers	1,762	6,332	100	51	34	20
Cameras	1,108	4,147	100	57	22	4
Watches	1,418	4,995	100	44	14	6
Shoes	1,214	4,591	100	49	7	0
All	5,502	20,065	100	50	20	9
Large						
Computers	6,146	27,213	100	51	31	18
Cameras	3,843	16,193	100	60	25	3
Watches	5,163	21,864	100	45	13	6
Shoes	3,482	19,507	100	51	6	0
All	18,634	84,777	100	51	19	8
XLarge						
Computers	9,690	58,771	100	50	30	16
Cameras	7,178	35,099	100	66	29	3
Watches	9,264	52,305	100	50	11	5
Shoes	4,141	38,288	100	53	5	0
All	30,273	184,463	100	54	19	7

4.7 Training Dataset and Benchmark Evaluation

The first three parts of this section present a set of matching experiments conducted using the WDC LSPM benchmark to show the usefulness of Schema.org annotated product identifiers as distant supervision to automatically generate training data using the clustering process described before. These experiments include the training of product matchers, the maintenance of product matchers for unseen products, and an evaluation of the noise-robustness of the matchers. The last two parts present a fine-tuning experiment with the BERT [Devlin et al., 2019] PLM including a large-scale experiment that uses millions of training pairs from the WDC Training Set for Product Matching to intermediately train BERT before fine-tuning it on the WDC LSPM benchmark. Corresponding code and datasets are available on GitHub[14][15]

[14]https://github.com/wbsg-uni-mannheim/wdc-lspc-v2
[15]https://github.com/wbsg-uni-mannheim/productbert-intermediate

4.7.1 Learning Product Matchers

In order to demonstrate the utility of the training sets for learning product matchers, all deep learning models that are part of the *Deepmatcher*[16] framework [Mudgal et al., 2018] are trained using the training sets of the LSPM benchmark. The Deepmatcher models are binary classifiers that compare two product offers by first converting all word tokens into an embedding space attribute-wise using various pre-trained embedding options like *fastText* embeddings [Bojanowski et al., 2017]. The embedding sequences of each attribute of the two product offers are then summarized and aggregated before concatenation of the resulting per-attribute embeddings as input to a classification module. The models in the Deepmatcher framework differ in the complexity of the summarization step. The *SIF* model aggregates embedding sequences by simply averaging them using smooth inverse frequency [Arora et al., 2017]. The *RNN* uses a bi-GRU, while the *Attention* model employs *decomposable attention* [Parikh et al., 2016]. Finally, *Hybrid* is a combination of the latter two models.

Experimental Setup

In the pre-processing step, punctuation is removed, all word tokens are lowercased, and stopwords are removed using the *NLTK*[17] stopword list. Experiments are performed for all product categories from the WDC LSPM benchmark as well as their combination. For each category and their combination, all four variants of the Deepmatcher models are trained using all four development set sizes. Character-based fastText embeddings pre-trained on Wikipedia[18] are chosen to convert the inputs to vectorized format, as these have shown to perform well for the task of product matching with Deepmatcher [Mudgal et al., 2018]. To compare pre-trained embeddings to self-training on the WDC Training Dataset, the *brand*, *title* and *description* attributes of the WDC Product Data Corpus are used to derive self-trained fastText embeddings for comparison. These embeddings are trained using the default parameters of the fastText library. All development sets of the WDC LSPM benchmark are split into training and validation sets using a stratified 80:20 split. All Deepmatcher experiments are run three times for 15 epochs each, and the results are averaged. All models use default parameters, apart from the positive-negative ratio, which allows for penalizing errors on the minority class more severely in imbalanced datasets. The positive-negative ratio is set to the actual ratio found in each training set. Once training is complete, the model is evaluated using the corresponding test set.

These experiments are repeated while allowing the model to propagate gradients back to the embedding layer itself, resulting in true end-to-end training, which is not part of the default Deepmatcher implementation that uses fixed pre-trained

[16] https://github.com/anhaidgroup/deepmatcher
[17] https://www.nltk.org/
[18] https://fasttext.cc/docs/en/pretrained-vectors.html

embeddings. This *end-to-end training* allows the model to fine-tune the embeddings themselves for the product matching task in addition to the parameter of the model itself.

Finally, for both the standard Deepmatcher and the end-to-end variant, the training set of size *xlarge* for the *Computers* category is selected and the parameters of the best-performing model are optimized using random search to show possible performance gains of an optimized model. The adjusted parameters include learning rate, learning rate decay, batch size, and positive-negative ratio. Each selected parameter combination is trained three times and the results are averaged to find the best-performing combination.

Baselines

The *Magellan* [Konda et al., 2016] framework and a simple baseline using *word co-occurrence* are trained using the same training and validation sets, to allow a comparison to traditional non-neural methods. Magellan offers automatic feature creation using string-based similarity metrics like Levenshtein and Jaccard similarity. These features are then used as input to scikit-learn [Pedregosa et al., 2011] classifiers. The word co-occurrence method uses a binary feature vector of words occurring in the attributes of both products without regard for attribute borders. The vocabulary of words is derived using the words occurring in the training set. The resulting feature vector is then used as input to scikit-learn classifiers. For both Magellan and word co-occurrence, hyperparameters are tuned using automatic grid-search. All experiments are carried out using different combinations of the features *title*, *description*, *brand*, and *specification table content*.

Results

Table 4.10 shows the results of the experiments on the largest of the four training sets for each category of products. The Magellan system shows the lowest performance, with F1 values between 60 and 69%. The word co-occurrence baseline reaches up to 84% F1 with significantly higher precision than Magellan. All of the Deepmatcher approaches achieve at least 90% F1 while achieving nearly 96% F1 for the *Watches* category. Overall, the RNN-based model performs consistently well for the product matching task on this benchmark. These first results already show that automatic creation of training data using Schema.org annotations as distant supervision allows training product matchers that can achieve a high level of performance comparable to using manually created training sets of other benchmarks (see Section 5.2).

When comparing default Deepmatcher to the end-to-end variation, it is evident that fine-tuning the embedding layer can improve performance by up to 4% F1, which can be mostly credited to improved recall. Parameter optimization, represented by (opt.) in Table 4.10, leads to nearly 2% higher F1 for the *Computers* set

Table 4.10: Results on LSPM training set XLarge.

Category	Model	Features	P	R	F1	σ
Magellan						
Computers	XGBoost	T+D+B+S	72.32	65.33	68.65	-
Cameras	XGBoost	T+D+B+S	73.87	54.67	62.84	-
Watches	XGBoost	T+D+B+S	74.15	50.67	60.20	-
Shoes	RandomForest	T+D+B+S	51.33	83.67	63.62	-
All	RandomForest	T+D+B+S	48.68	78.42	60.07	-
Word Co-Occurrence						
Computers	LinearSVC	T+D+B+S	86.74	80.67	83.59	-
Cameras	LinearSVC	T+D+B+S	81.51	64.67	72.12	-
Watches	LinearSVC	T+D+B+S	88.14	69.33	77.61	-
Shoes	LogisticRegression	T	86.14	58.00	69.32	-
All	LogisticRegression	T+D+B+S	85.97	66.92	75.26	-
Deepmatcher - Pre-Trained fastText						
Computers	RNN	T+D+B	89.75	91.89	90.80	0.79
Computers(opt.)	RNN	T+D+B	93.07	92.11	92.56	0.76
Cameras	RNN	T+D+B+S	89.22	89.22	89.21	1.68
Watches	RNN	T+D+B+S	92.73	94.22	93.45	0.93
Shoes	RNN	T	93.16	92.11	92.61	1.05
All	RNN	T+D	92.04	88.36	90.16	0.43
Deepmatcher - Self-Trained fastText						
Computers	RNN	T+D+B	89.66	94.22	91.88	0.79
Cameras	RNN	T+D	89.74	85.44	87.53	1.34
Watches	RNN	T+D	93.66	91.78	92.70	0.95
Shoes	RNN	T	90.65	90.45	90.54	0.85
All	RNN	T+D	90.98	87.31	89.10	0.39
Deepmatcher(End-to-End) - Pre-Trained fastText						
Computers	RNN	T	94.00	97.00	95.46	0.70
Computers(opt.)	RNN	T+D+B	93.76	96.78	95.25	0.31
Cameras	RNN	T	91.44	93.00	92.18	0.70
Watches	RNN	T+D	94.58	96.89	**95.72**	0.50
Shoes	RNN	T	95.58	93.78	**94.67**	0.54
All	RNN	T+D+B	91.25	93.38	92.30	0.31
Deepmatcher(End-to-End) - Self-Trained fastText						
Computers	RNN	T+D+B	93.88	97.16	**95.50**	0.35
Cameras	RNN	T	93.16	92.11	**92.63**	0.27
Watches	RNN	T+D	94.84	95.17	95.00	1.45
Shoes	RNN	T	93.25	91.56	92.38	0.25
All	RNN	T+D+B	92.40	93.62	**92.98**	0.11

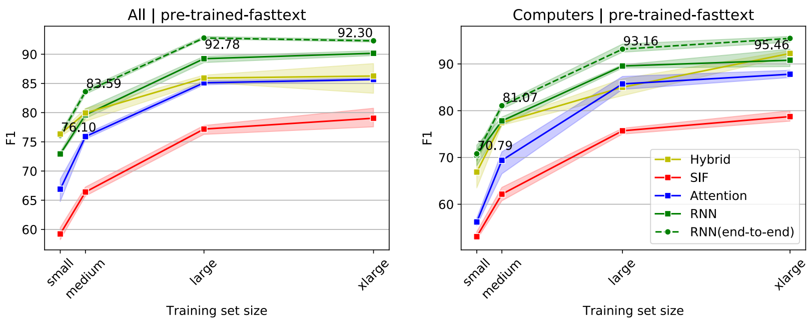
and standard RNN Deepmatcher, while improving primarily on precision. The end-to-end RNN parameter optimization did not lead to performance improvements.

The comparison of pre-trained and self-trained embeddings shows that self-training can marginally increase the performance for some product categories, but overall the fine-tuning of already pre-trained embeddings leads to higher results on average for the selected models and embeddings. Overall, fine-tuning pre-trained embeddings offers significant performance gains without the overhead of self-training on large corpora. An inspection of the training times of the *computers* RNN and its end-to-end equivalent shows an increase of 6% on average for the latter, which is acceptable considering the possible gains in performance.

The learning curves of a model represent an evaluation of the models performance along the dimension of available training data, in this case represented by the four training set sizes available in the LSPM benchmark. Figure 4.2 shows the learning curves for the four Deepmatcher models, as well as the end-to-end variant of the RNN on the *Computers* and combined test sets. This is representative of the learning curves of all product categories, with only minor differences between them. The curve shows that the RNN model significantly surpasses the other models for training sets of size *large*. The overall improvement from *large* to *xlarge* is only up to 3% F1 according to the experiments, showing that training sets of the size of *large* are already close to the maximum observed performance and may be preferable for practical applications due to the significantly lower amount of required training labels. The end-to-end variant of Deepmatcher surpasses the standard for medium-sized training sets and thus should be the preferred way of training if a medium-sized or larger training set is available.

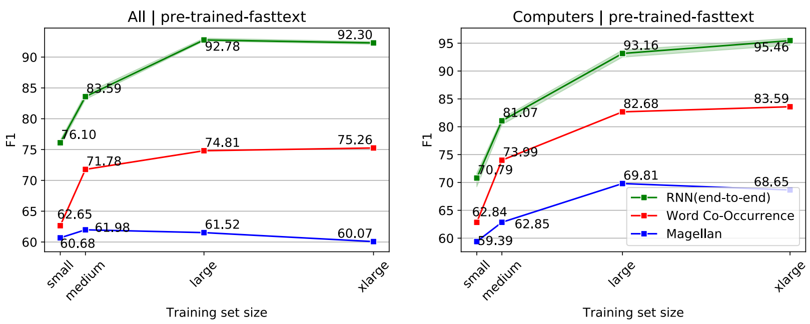
Figure 4.3 shows the learning curves of the best-performing Deepmatcher model (end-to-end RNN) compared to those of the traditional baselines on the *Computers* and combined test sets. Whereas Magellan does not improve with larger size training sets, both word co-occurrence and Deepmatcher show a strong increase with training sets of size *medium*. The *large* set marginally improves the performance of word co-occurrence by approximately 3% F1, while Deepmatcher gains nearly 10%. For all product categories, the performance of Deepmatcher on the smallest training set is greater than that of the traditional methods, making it the preferred method if resources are available for training and maintaining deep learning models. However, training sets of at least size *medium* are required to achieve a good level of performance.

Overall, the results demonstrate that using Schema.org annotations as distant supervision for building training sets allows to learn models that can achieve 90% and more F1 over several product categories considering the possible noise in labels of these automatically created training sets due to annotation errors or wrong annotation practices, as discussed in Section 4.5. Consequently, they are a viable alternative to manually labeled datasets, especially when considering the required size and the associated manual human workload this would incur.



(a) All **(b)** Computers

Figure 4.2: Comparison of Deepmatcher models.



(a) All **(b)** Computers

Figure 4.3: Comparison Deepmatcher to baselines.

4.7.2 Maintaining Matchers to Cover Unseen Products

An additional dimension to consider when training product matchers is the constant introduction of new products to the market, which are unseen during the training process of the matchers. For e-commerce applications, it is crucial to be able to cluster offers for unseen products and correctly distinguish them from offers for seen (known) products. This section investigates to what extent the matchers that were trained in Section 4.7.1 for a set of products can generalize to new, unseen products. This is followed by evaluating different approaches for fine-tuning these matchers using offers for previously unseen products gathered from the Web in the English WDC Training Dataset.

Experimental Setup

In order to evaluate how matchers generalize to unseen products, 30 of the 150 products having positive and negative pairs in the test set are randomly selected. These 30 products are treated as *unseen*. The test set of the category *Computers*, as well as the *Computers xlarge* development set, are split accordingly by removing all pairs that contain any of these 30 products from both sets into a new development set and test set. Table 4.11 provides statistics on the resulting development and test sets for *seen* and *unseen* products.

Table 4.11: Datasets for seen and unseen new products.

Category	# Pos. Pairs	# Neg. Pairs	% Density T	D	B
Test Unseen	59	154	100	89	57
Test Seen	241	646	100	90	56
Dev Unseen	264	3,562	100	66	49
Dev Seen	7,488	43,454	100	69	49
Dev Comb.	7,752	47,016	100	69	49

Generalization to Unseen Products

In the first experiment, the different matching models are trained using the training data for the seen products and subsequently evaluated using the test set for unseen products. Table 4.12 shows the results for the evaluation of the trained model on seen and unseen products for the Deepmatcher, as well as the Magellan and word co-occurrence baselines. The performance of Deepmatcher on the seen products is comparable to the results obtained in Table 4.10 for the *Computers xlarge* training set. Performance on unseen products drops considerably for Deepmatcher and word co-occurrence with losses of 16% and 40% F1, respectively. Magellan is the least affected, with an absolute loss of 4% in F1. However, the F1 performance for unseen products of all three models is below 90% and, as a result, likely insufficient for many e-commerce applications. The continuous maintenance of product matchers to cover new products must be considered necessary for such applications.

Table 4.12: Results for seen and unseen new products.

Model	Seen				Unseen				
	P	R	F1	σ	P	R	F1	σ	$\Delta F1$
Word Co-Occ.	84.9	81.7	83.3	-	72.0	30.5	42.9	-	40.4
Magellan	73.1	64.3	68.4	-	70.0	59.3	64.2	-	4.2
Deepmatcher	92.8	91.0	91.9	0.5	76.2	75.7	75.9	2.9	12.6

Fine-tuning Matchers for New Products

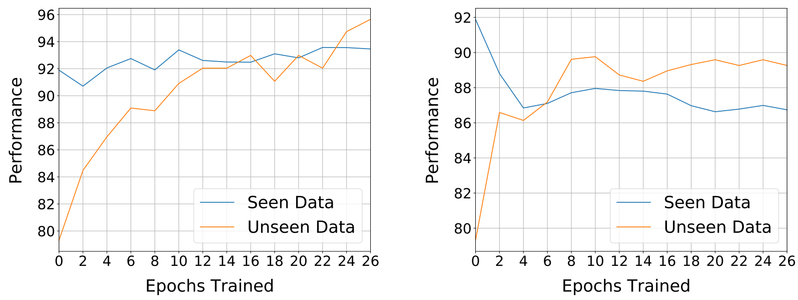
By regularly recrawling e-shops that are known to annotate product identifiers using Schema.org terms, it is possible to monitor the appearance of new products on the Web (products having identifiers that have not been contained in previous crawls) and to gather heterogeneous offers for these products. The crawled offers for new products can subsequently be used to maintain the existing product matchers.

In the following, three methods of adapting the product matcher to previously unseen products are explored. Firstly, the trained model is fine-tuned by continuing the training using only the small training set containing the unseen products from Table 4.11. Secondly, the model is fine-tuned using the training sets for previously seen as well as previously unseen products, i.e., all available training data. Both fine-tuning approaches are evaluated after every two trained epochs. The model optimizer is reset before fine-tuning. Finally, a model is trained from scratch using all of the training data for comparison with fine-tuning.

Figure 4.4 shows the results of the two continuously evaluated fine-tuning approaches for the products categorized as previously seen and unseen for up to 26 epochs. Fine-tuning the model using only training data for previously unseen products achieves around 90% F1 after 8 epochs on the previously unseen products, while performance on the previously seen products drops to around 89% F1 due to the network forgetting patterns learned during previous training due to noisy weight updates during this fine-tuning step.

This observation is supported, as when training using the combined training sets, the performance on the seen products does not experience this drop and can increase further by a small amount as the patterns of the seen products remain part of the training set. For previously unseen products, a good performance of 90% F1 is achieved after 10 epochs. This number further increases until it can match the performance on seen products after 16 epochs.

Fully re-training the model for 15 epochs on the combined training set achieves 91% F1 on the seen and 90% F1 on the unseen products, which is marginally



(a) Fine-tuning on full set. **(b)** Fine-tuning on new data only.

Figure 4.4: Fine-tuning product matchers

less performant than fine-tuning an existing model with the same training set over the same amount of epochs, making this the least desirable approach for matcher maintenance.

The two fine-tuning approaches differ in the size of the training set used (3,826 vs. 50,942 pairs) and subsequently in training time per epoch. In the experiments, one epoch of fine-tuning on the large set takes approximately 30 times longer than training one epoch on the small set, making the latter approach more relevant in time-constrained scenarios. One limitation of these experiments lies in the fact that it is not clear how much of the performance differences are attributable to the offers for new/existing products or size effects that may occur between two training sets of varying size.

4.7.3 Impact of Noisy Training Data

The usage of Schema.org data from a large number of websites as distant supervision implies a certain level of noise in the derived labels due to errors or wrong annotation practices, as discussed earlier. The results of Section 4.7.1 show that the 6% label-noise inherent to the WDC Training Dataset, as explained in Section 4.5, still allows the learning of high-performance product matchers.

In order to investigate the impact of further label noise on the performance of neural and non-neural methods, experiments with additional artificial label noise are conducted. For this purpose, the labels of 5, 10, 20, 30, and 40% of the labels in the *Computers xlarge* training set are randomly changed to the opposite label. Both positive and negative labels are affected equally. As the WDC Training Dataset contains approximately 6% inherent label noise, this serves as the minimum possible noise value for the experiments. The optimized standard Deepmatcher RNN model is trained on the *Computers xlarge* training set for this experiment. The model is trained for 15 epochs using the same train/validation split as in the experiments presented in Section 4.7.1.

Table 4.13 shows the results of this experiment. The Magellan and word co-

Table 4.13: Impact of Noisy Training Data.

Noise	Magellan			Word Co-Occ.			Deepmatcher		
%	P	R	F1	P	R	F1	P	R	F1
+0	74.13	64.00	68.69	84.91	80.67	82.74	93.12	93.11	93.11
+5	74.31	63.67	68.59	85.20	78.67	81.80	84.75	83.11	83.90
+10	68.66	61.33	64.79	79.42	73.33	76.26	79.82	77.56	78.66
+20	48.24	91.33	63.13	71.38	69.00	70.17	71.83	75.67	73.69
+30	36.40	92.33	52.21	48.12	85.33	61.54	45.49	71.67	55.63
+40	30.65	91.33	45.90	31.52	63.67	42.16	35.59	62.45	45.29

occurrence models can handle 5% in additional noise with only marginal losses in performance. Deepmatcher loses 10% F1 in this scenario. The addition of another 5% noise does impact all of the models significantly, with Magellan losing 4%, word co-occurrence 5%, and Deepmatcher 5% F1. With increasing label noise, all models continue to drop in performance until rapid deterioration occurs around the mark of 30% and 40% noise for all models. Interestingly, Deepmatcher still performs comparably better, but the difference in performance to the traditional methods is only marginal at this point.

Overall, the non-neural models can handle up to 10% label noise with marginal loss in performance, while Deepmatcher is severely affected by any increase in label noise. With further addition of label noise, all models rapidly decrease in performance. As a result, for designing the process of automatically deriving training labels using distant supervision, it is very important to make sure that label-noise is minimized as much as possible.

4.7.4 Fine-Tuning the BERT Transformer

Deep Transformer networks [Vaswani et al., 2017], pre-trained on large corpora via language modeling objectives [Devlin et al., 2019, Liu et al., 2019b, Clark et al., 2020] significantly pushed the state-of-the-art in a variety of downstream tasks [Wang et al., 2019, Hu et al., 2020], including a number of sentence-pair classification tasks, e.g. paraphrase identification [Dolan and Brockett, 2005]. Early studies with Transformers in entity matching [Brunner and Stockinger, 2020, Li et al., 2020] also demonstrate the effectiveness of models like BERT [Devlin et al., 2019] for the task of entity matching.

In this Section, BERT is fine-tuned for product matching using the WDC LSPM benchmarks and shown to be more training data efficient as well as more performant compared to training the Deepmatcher [Mudgal et al., 2018] models when using automatically generated training sets from Schema.org annotations. Fine-tuning BERT results in 15-20% higher F1 scores in settings with small- and medium-sized training sets. Even for large training sets, fine-tuning BERT can still yield a 2% improvement over Deepmatcher.

Fine-tuning Setup

The experiments use the training, validation, and test sets from the *Computers* category of the WDC LSPM benchmark. To test the efficiency of the classifiers with respect to training size, the experiments are conducted with all training set sizes: *small*, *medium*, *large*, and *xlarge*.

The input for BERT, consisting of two sequences, one for each product offer, is the concatenation of the attributes of both offers. To this end, all attributes of each product offer are concatenated into one string using the attributes *brand*, *title*, *description*, and *specification table content* in this order.

All experiments are conducted with PyTorch [Paszke et al., 2019] using BERT's implementation[19] from the HuggingFace Transformers library [Wolf et al., 2020]. All hyperparameters are set to their defaults if not stated otherwise. The chosen loss function is the binary cross-entropy loss using Adam [Kingma and Ba, 2015] as the optimization algorithm. BERT allows for input sequences of maximal length of 512 tokens. First, each attributes length is constrained to *5* (*brand*), *50* (*title*), *100* (*description*) and *200* (*specification table content*) words, respectively, dropping any words outside that range. Long product offers are further truncated by removing tokens from their end until BERT's constraints are satisfied. Fine-tuning happens on all layers for 50 epochs with a linearly decaying learning rate with warm-up over the first epoch. The validation set is used for model selection and early stopping: if the F1 score on the validation set does not improve over 10 consecutive epochs, the training stops. The batch size is fixed to 32 and the learning rate is optimized in the range [5e-6, 1e-5, 3e-5, 5e-5, 8e-5, 1e-4]. Three model instances are trained for each hyperparameter configuration, and the average performance is reported.

BERT-based product matching is compared with the same methods used in Section 4.7. The Magellan and word co-occurrence methods are combined with *XGBoost*, *Random Forest*, *Decision Tree*, *linear SVM*, and *Logistic Regression* as classification methods and randomized search is applied over the respective hyperparameter spaces. For Deepmatcher, fastText embeddings trained on the English Wikipedia[20] are used as input, and end-to-end training of word embeddings is enabled, as it has been shown to improve performance in Section 4.7.1. All Deepmatcher instances are trained for 50 epochs with default parameters apart from the learning rate, which is optimized in the same range as for BERT. For the preprocessing of Deepmatcher and BERT inputs, the method-specific tokenizers are used. For the other baselines, all attributes are lower-cased.

Fine-tuning Results

Table 4.14 compares the results of fine-tuning BERT to the baselines. BERT outperforms all three baselines in all settings. The gains from BERT-based product

[19] Used pre-trained BERT instance: *bert-base-uncased*.
[20] https://fasttext.cc/docs/en/pretrained-vectors.html

Table 4.14: BERT compared to baselines on WDC LSPM computers.

	Word Cooc.			Magellan			Deepmatcher			BERT			Ditto
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	F1
xlarge	86.59	79.67	82.99	71.44	56.89	63.33	89.63	94.78	92.12	95.99	93.00	**94.47**	95.45
large	79.52	77.67	78.58	67.67	63.67	65.60	85.70	91.22	88.38	91.64	95.00	**93.29**	91.70
medium	65.83	78.33	71.54	48.99	81.56	61.20	66.39	82.78	73.67	84.89	94.22	**89.31**	88.62
small	53.98	74.67	62.66	50.86	71.22	59.17	54.86	69.56	61.20	75.62	89.33	**81.89**	80.76

matching become larger the smaller the training dataset is: for the smallest training set, BERT outperforms Deepmatcher by 20% F1. For the largest training set, BERT gains 2.3% F1 over Deepmatcher. The results demonstrate that using product offers originating from Schema.org annotations is also a useful source of data for fine-tuning the BERT Transformer model. The next section shows that large-scale training of BERT with millions of product offer pairs from the WDC Training Dataset can further improve these results.

4.7.5 Large-Scale Intermediate Training of BERT

BERT has been pre-trained on a general-purpose natural language corpus, with the style of language and topics of most of its natural language training data, BookCorpus and Wikipedia, being very different from product descriptions. Intermediate training on large training sets for other tasks [Phang et al., 2018, Pruksachatkun et al., 2020] has been shown to improve downstream performance. In an effort to test the intuitive assumption that intermediate in-domain training can improve matching performance and to further substantiate the usefulness using Schema.org annotations as distant supervision, an intermediate training step is introduced before the model's final fine-tuning for specific products. In this intermediate step, BERT is trained on millions of product offer pairs from the English WDC Training Dataset.

In a second experiment, domain-specific (self-supervised) language modeling is added to the intermediate training as a second objective, proving the usefulness of the WDC Training Dataset also for product-specific language modeling.

Creation of Intermediate Training Sets

The English WDC Product Corpus for Large-Scale Product Matching and its product cluster structure are used to build wide coverage training sets consisting of millions of offer pairs. In order to have an unbiased evaluation, the clusters contained in the test set and fine-tuning training sets of the WDC LSPM benchmark are removed from the corpus prior to building the intermediate training sets.

Subsequently, the effects of intermediate training on two structurally different training sets are compared. The first intermediate training set contains only offer pairs for the category *Computers*: this leads to the introduction of more *Computer*-related information into BERT and steers the Transformer network to detect

Table 4.15: Intermediate training set statistics.

	# products w/ pos (overall)	# pos. pairs	# neg pairs	# comb. pairs
computers only	60,030 (286,356)	409,445	2,446,765	2,856,210
4 categories	201,380 (838,317)	858,308	2,665,056	3,523,364

relevant linguistic phenomena for recognizing matches between *Computer* offers. The second training set contains pairs from all four categories of the WDC LSPM benchmark with fewer training pairs per product. This offers a wider selection of products, resulting in more versatile information about what constitutes a product match for the model, but less in-depth information for each product/category.

The intermediate training sets, named *computers* and *4 categories*, are created as follows: for positive instances, only clusters containing more than one offer, from which at least one positive pair can be built, are selected. The selection of clusters is restricted to those of size $\leqslant$80. For each offer in each cluster, up to 15 (computers) or 5 (4 categories) positive pairs are created with the other offers from that cluster. Half of those are *positive corner cases*, created by (1) applying cosine similarity between the bag-of-words vectors for concatenation of title and the first five words of description and (2) sorting the offer pairs by cosine similarity and selecting the pairs with the lowest scores. The remaining 50% are selected by randomly pairing offers from the same cluster. Negative pairs are created using a similar heuristic: for each offer used for positive pairs, the same amount of negative pairs is created using offers from other clusters of the same category. *Negative corner cases* (50%) are pairs of offers from different clusters with the highest cosine similarity. The other half are randomly sampled pairs of offers from different clusters. Table 4.15 displays the statistics of the resulting intermediate training sets.

Intermediate Training Procedure

For the first set of experiments, intermediate training is performed with a single objective, the binary product matching task. The architecture is exactly the same as for the previous fine-tuning experiments. One model is trained for each of the training sets in Table 4.15. After intermediate training, the model is evaluated with and without final product-specific fine-tuning. The intermediate training is run for 40 epochs with a linearly decaying learning rate (starting from 5e-5) with 10,000 warm-up steps and a batch size of 256. Due to the long training times, the first 90% of the epochs are trained on sequences of length 128 and only the last 10% on the full sequences of 512 tokens to speed up training, similar to the original BERT training procedure [Devlin et al., 2019].

In the second set of experiments, the masked language modeling objective is added to the product matching objective and both are jointly optimized in the in-

Table 4.16: Intermediate training with PM objective.

	intermediate training							
	computers category				4 categories			
	P	R	F1	Δ only fine-tune	P	R	F1	Δ only fine-tune
xlarge	95.58	93.67	94.61	0.14	95.45	95.44	95.45	0.98
large	92.68	95.56	94.09	0.80	91.34	96.00	93.61	0.32
medium	94.01	95.78	94.88	5.57	91.59	95.67	93.59	4.28
small	94.38	93.11	93.73	11.84	90.39	90.89	90.64	8.75
none	94.41	90.00	92.15	-2.32 (xl)	88.24	95.00	91.49	-2.98 (xl)

termediate training step. For masking words, the original masking procedure is followed: randomly, 15% of tokens are selected for replacement. In 80% of the cases, the token is replaced with the [MASK] token, in 10% of the cases with a random vocabulary token, and in the remaining 10% the original token is kept. As in the original work, the Transformer network is trained by minimizing the cross-entropy loss over predictions of masked tokens. After intermediate training, two model variants are evaluated: with and without the final product-specific matching fine-tuning.

Intermediate Training Results

Table 4.16 shows the results of the intermediate training procedure. When comparing the intermediate training on the *computers* training set against the intermediate training on the training set comprising 4 product categories, it can be seen that without final fine-tuning (row *none* in Table 4.16), it is already possible to achieve a good matching performance of 92% F1. This result shows that, through the intermediate training, category-specific knowledge is injected into BERT's parameters, as it is able to make good matching predictions for products for which it has not seen any training examples yet. Once the intermediate model is subjected to further fine-tuning on offer pairs from the training sets, further improvements in all settings are observable, with improvements being most prominent for the smallest training set. Intermediate training followed by fine-tuning on the small training set reaches a performance of $\sim$94% F1, which, without intermediate pre-training, was previously obtained only on the largest training set. Training on category-specific data (*computers*) yields marginally better performance compared to training on the mix of pairs from 4 categories.

Table 4.17 shows the results of adding the MLM objective to the product matching objective in the intermediate training step using the *computers* intermediate training set. Compared to the corresponding settings in which the intermediate training did not include MLM, the performance with fine-tuning increases by up to 3% F1, producing a new top matching performance, >97% F1 for the largest

Table 4.17: Intermediate training with PM and MLM objective.

	intermediate training - PM + MLM				
	P	R	F1	Δ only fine-tune	Δ interm. only PM
xlarge	98.20	96.56	97.37	2.90	2.76
large	94.99	96.67	95.82	2.53	1.73
medium	96.05	97.11	96.58	7.27	1.70
small	95.64	97.44	96.53	14.64	2.80
none	94.31	94.00	94.16	-0.31 (xl)	2.01

training set and 96% F1 for all other training sizes. This confirms the findings from other application domains [Beltagy et al., 2019, Lee et al., 2020] that point to the benefits of domain-specific MLM pre-training. The original pre-training data likely contains only few instances of product-specific vocabulary, as it covers a wide range of topics. Applying intermediate MLM training on domain-specific data allows for better adaptation of vocabulary embeddings to the domain, resulting in better downstream performance.

In summary, subjecting BERT to an intermediate training step with large amounts of product data sourced from Schema.org annotations leads to a model that generalizes well to new unseen products from the same category and can be easily fine-tuned with small amounts of product-specific training data to further increase the performance for these products. Depending on the structure of the intermediate training set, more training data for a single category leads to a small increase in performance compared to a more heterogeneous training set encompassing a larger set of products from several categories. Adding the MLM objective to the intermediate training results in further improvements in matching performance, showing that domain-specific language modeling further adapts BERT's parameters to the product domain.

4.8 Conclusion

This chapter first presented a process for clustering product offers found in the Common Crawl using product identifiers like GTINs and MPNs that were extracted from Schema.org annotations from thousands of websites. Ideally, all offers in a cluster refer to the same product. To further ensure this, the process is accompanied by various cleansing steps such as detecting and removing offers annotated with categorical instead of product identifiers. The resulting corpus, the WDC Training Dataset for Large-Scale Product Matching, contains 26 million product offers originating from 79 thousand websites that are grouped in 16 million clusters. The profiling of the corpus showed that is possible to generate and automatically label 40 million matching pairs and over 5 trillion non-matching pairs from these clusters. The diversity in terms of sources and its size make this corpus the largest publicly available source of labeled data for product matching at the time of its creation, with a size that is several orders of magnitude larger than training sets of previous publicly available benchmarks.

The contributed WDC LSPM benchmark is based on this clustering. The benchmark offers training, validation, and testing splits for the four product categories *computers*, *cameras*, which contain more structured attribute information in their textual attributes, and *watches* and *shoes*, which have less structured textual descriptions and their combination. The training sets are available in four sizes ranging from two thousand to 65 thousand labeled pairs. The largest combined set containing all categories contains over 30 thousand matching and over 180 thousand non-matching pairs making it the largest and most diverse publicly available multi-source entity matching benchmark at the time of its creation. It further contains a second test for the computers category that contains pairs hand-annotated with specific matching challenges for more fine-grained evaluation of matching systems.

The LSPM benchmark and additional training pairs from the clustering were subsequently used to evaluate the usefulness of automatically generated training data using Schema.org annotations by training a set of neural and non-neural product matchers. The experiments showed that the Deepmatcher RNN model trained on these pairs can achieve a high performance of more than 90% F1, which can be further increased by allowing end-to-end training from embeddings to the final layer. The comparison to the non-neural baselines further confirmed the findings of Mudgal et al. [Mudgal et al., 2018] that deep neural networks perform better than symbolic methods on highly textual data. Although Deepmatcher loses 16% performance in an unseen scenario, the experiments showed that the matchers can be maintained to reach the same performance as on seen products by selecting relevant pairs from the clusters of the English WDC Training Dataset for continued fine-tuning of the Deepmatcher RNN model.

The experiments analyzing the noise resistance of the models showed that the average of 6% label noise inherent to the training pairs due to the automated clustering process does not prohibit training high-performance matchers, although further increases in label-noise rapidly deteriorate the matching performance of Deepmatcher.

Lastly, the million-scale intermediate training experiments with the BERT transformer further showed the usefulness of the size and diversity of the automatically generated training pairs, which further improved performance compared to solely fine-tuning the model, especially when adding the masked language modeling objective that was part of BERTs pre-training and training on millions of product offer pairs from the same category. This increased the performance of the BERT Transformer to over 97% F1 on the *computers* category of the LSPM benchmark with fine-tuning and 94% without additional fine-tuning, reaching 2-13% higher F1 for the respective fine-tuning set sizes compared to the Ditto method [Li et al., 2020].

Chapter 5

WDC Products: A Multi-Dimensional Entity Matching Benchmark

5.1 Introduction

Over the last decades, a wide range of benchmarks have been developed to compare the performance of entity matching systems [Primpeli and Bizer, 2020,Peeters et al., 2024b]. The datasets used by these benchmarks range from structured datasets to datasets exhibiting mostly textual entity descriptions. While the older benchmarks focus on matching records between two data sources, more recent benchmarks often rely on data from larger numbers of Web data sources that are more heterogeneous and more difficult to match as a result (see Section 5.4 for a comparison).

Although the field of benchmarking entity matching systems is mature, publicly available benchmarks do not explicitly cover the following aspects:

1. **Explicit Modeling of Combinations of Challenges:** The overall difficulty of an entity matching task depends on a combination of multiple dimensions, such as the amount of hard-to-solve matches and non-matches (corner cases) or the number of available training examples. Current benchmarks usually represent single points in the space along such dimensions, or they provide for the evaluation of matching tools along a single dimension, for instance, by offering development sets of different sizes. The benchmarks do not support the systematic evaluation of matching systems along a combination of different dimensions, for example, offering development sets of different sizes that exhibit different corner case fractions. Such multi-dimensionality would provide for a more fine-grained evaluation of the strengths and weaknesses of matching systems in a controlled environment.
2. **Generalization to Unseen Entities:** While state-of-the-art matching sys-

tems [Li et al., 2020, Yao et al., 2022] reach a very high performance for matching entities that are covered by records in the training data, experiments involving unseen entities which are not covered by the training data indicate that the methods lack robustness and perform worse for such entities [Wang et al., 2022d, Peeters and Bizer, 2021]. None of the publicly available English-language entity matching benchmarks explicitly measure the robustness of matchers with regard to their performance for unseen entities. Such unseen entities play an important role in many use cases, for example, when new products frequently appear in e-commerce scenarios [Wang et al., 2022d], while new books and new articles play a central role in bibliographic data management.

3. **Larger number of Descriptions per Entity:** Some of the widely used benchmarks only contain a small number of matching records per entity, e.g., on average around 1.2 records (see Table 5.1 and discussion in Section 5.4). This lack of training records hinders the systematic analysis along the dimension of available training data.

4. **Entity Matching as Multi-Class Classification:** There are matching use cases that do not require being able to match arbitrary entities but require the matcher to recognize a previously known set of entities. For instance, for price tracking or a market analysis, it may be necessary to find offers for the specific set of products that the company produces. For these use cases, treating entity matching as multi-class classification can be more appropriate, e.g. learning how to recognize relevant products in a large set of product offers. In this case, instead of the binary labeling of pairs of records into matches and non-matches, the classifier labels each record with a single label from the label space of all relevant entities.

This chapter contributes the WDC Products benchmark, which covers all of the points listed above, compared to publicly available benchmarks, to the field of benchmarking of entity matching methods. This benchmark, like the WDC LSPM benchmark, is based on Schema.org annotations for product offers, and the same process presented in the previous chapter is the basis for creating this benchmark. More specifically, *WDC Products*[1] is created from a similar clustering as WDC LSPM with the December 2020 Common Crawl as the data source.

The contribution of this chapter is the following:

- **WDC Products - A Multi-Dimensional Entity Matching Benchmark:** As an evolution of the LSPM benchmark, the highly textual WDC Products benchmark is designed along the evaluation of three dimensions that matching systems should handle. These are (1) the number of corner cases (hard positives and negatives), (2) the number of unseen entities in the test set,

[1] https://webdatacommons.org/largescaleproductcorpus/wdc-products/

and (3) the size of the development set. The WDC Products Benchmark contains 26 variants of training, validation, and test sets to evaluate each dimension separately by fixing the other two, which is impossible in publicly available benchmarks. The WDC Products benchmark, due to its structure, is also available in a multi-class formulation of the entity matching problem, allowing for an evaluation of this formulation of the task.

The WDC Products benchmark was published in [Peeters et al., 2024b] and is joint work with Reng Chiz Der, who contributed the Ditto and HierGAT baseline experiments and supported with annotation work.

Section 5.2 presents related work on existing entity matching benchmarks. Section 5.3 presents the creation process and profile of the multi-dimensional WDC Products benchmark. Section 5.4 compares both previously presented benchmarks (WDC LSPM and WDC Products) with existing benchmarks. Section 5.5 concludes the chapter with a discussion of the contributions.

5.2 Related Work

This section gives an overview of publicly available benchmarks for the entity matching task, including available data generators for creating synthetic benchmarks. The section ends with a short discussion of framing the entity matching task as a non-binary matching task.

Benchmarks for Entity Matching

Today, a large number of publicly accessible entity matching benchmarks exist, which were proposed over the past 50 years of entity matching research [Primpeli and Bizer, 2020]. Currently, only a subset is regularly used in recent research due to factors such as difficulty, size, and adoption of fixed evaluation splits. Almost all of these benchmarks provide datasets consisting of one or more sources together with a list of matching entities between these sources. However, these lists do not always provide a complete mapping that contains all matching pairs. Most of these benchmarks were created during a time when learning-based algorithms did not have the prominence they have today. As a result, fixed evaluation splits following the machine learning evaluation paradigm of training, validation, and test sets were not yet relevant. Consequently, researchers must devise their own evaluation setup, and if there is no agreement among the community, any results reported for entity matching systems are difficult to compare between research works [Primpeli and Bizer, 2020]. Furthermore, all existing publicly available benchmarks exhibit several limitations listed in Section 5.1. The following paragraphs list important repositories and publications for entity matching benchmarks sorted by time of cre-

Table 5.1: Comparison of WDC Products to existing original benchmarks that have been used in entity matching work or have been recently proposed. See [Primpeli and Bizer, 2020] for additional rarely used benchmarks. Splits marked with * have not been released with the respective benchmark but were adopted by the community. Dev and test size refer to the largest available split if applicable.

Benchmark	Domain	# Sources	# Entities	# Records	# Attr	Avg. Density	# Matches	# Non-Matches	Avg. Matches per Entity	Fixed Splits (# Train Sizes)	Dev Size (Matches)	Test Size (Matches)	Max F1 in related work
							Leipzig Database Group						
Abt-Buy	Product	2	1,012	1,081/1,092	3	0.63	1,095	-	1.08	✓* (1)	7,659 (822)	1,916 (206)	94.29 [Peeters and Bizer, 2022b]
Amazon-Google	Product	2	995	1,363/3,226	4	0.75	1,298	-	1.30	✓* (1)	9,167 (933)	2,293 (234)	79.28 [Peeters and Bizer, 2022b]
DBLP-ACM	Bibliogr.	2	2,220	2,614/2,294	4	1	2,223	-	1.00	✓* (1)	9,890 (1,776)	2,473 (444)	99.17 [Li et al., 2020]
DBLP-Scholar	Bibliogr.	2	2,351	2,616/64,263	4	0.81	5,346	-	2.27	✓* (1)	22,965 (4,277)	5,742 (1,070)	96.30 [Yao et al., 2022]
							DuDe						
Restaurants	Company	2	110	533/331	5	1	112	-	1.02	✓* (1)	757 (88)	189 (22)	100.00 [Li et al., 2020]
Cora	Bibliogr.	1	118	1,879	18	0.31	64,578	268,082	547.27		-	-	100.00 [Wang et al., 2011]
							Magellan						
Walmart-Amazon	Product	2	846	2,554/22,074	10	0.84	1,154	-	1.36	✓* (1)	8,193 (769)	2,049 (193)	88.20 [Yao et al., 2022]
Company	Company	2	28,200	28,200/28,200	1	1	28,200	84,432	1.00	✓* (1)	90,129 (22,560)	22,503 (5,640)	93.85 [Li et al., 2020]
Beer	Product	2	68	4,345/3,000	4	0.96	68	382	1.00	✓* (1)	359 (54)	91 (14)	94.37 [Li et al., 2020]
iTunes-Amazon	Product	2	120	6,906/55,932	7	0.99	132	407	1.10	✓* (1)	430 (105)	109 (27)	97.80 [Li et al., 2020]
							Alaska						
Camera	Product	24	103	3,865	56	0.13	157,157	-	1,525.80		-	-	99.40 [Yao et al., 2022]
Monitor	Product	26	242	2,283	87	0.17	13,556	-	56.02		-	-	99.60 [Yao et al., 2022]
							Chinese Academy of Sciences						
Ember	Product	1	350	6,245	5	1	5,053	206,296	14.44	✓(1)	8,000 (1,974)	50,000 (500)	78.45-96.89 [Wang et al., 2022d]
							WDC						
LSPM Computers	Product	269	745	3,665	4	0.51	7,478	59,571	10.04	✓(4)	68,461 (9,690)	1,100 (300)	98.33 [Peeters and Bizer, 2022b]
LSPM Cameras	Product	190	562	4,068	4	0.43	9,564	35,899	17.02	✓(4)	42,277 (7,178)	1,100 (300)	98.02 [Peeters and Bizer, 2021]
LSPM Watches	Product	235	615	4,676	4	0.5	9,991	53,105	16.25	✓(4)	61,569 (9,264)	1,100 (300)	97.09 [Peeters and Bizer, 2021]
LSPM Shoes	Product	120	562	2,808	4	0.41	4,440	39,088	7.90	✓(4)	42,429 (4,141)	1,100 (300)	97.88 [Peeters and Bizer, 2021]
WDC Products	**Product**	**3,259**	**2,162**	**11,715**	**5**	**0.79**	**28,299**	**124,899**	**13.09**	**✓(3)**	**24,335 (8,971)**	**4,500 (500)**	**64.50-89.04**

ation. Table 5.1 gives an overview of key features of entity matching benchmarks that are frequently used in publications[2].

The University of Leipzig Database Group[3] published four benchmark tasks for pair-wise entity matching in 2010 [Köpcke et al., 2010] which are still widely used today [Mudgal et al., 2018, Li et al., 2020]. These benchmarks were released as two-source tasks where two tables and the perfect mapping between both are available. The *Abt-Buy* and *Amazon-Google* benchmarks consist of longer textual attributes of the product domain, while the *DBLP-Scholar* and *DBLP-ACM datasets* contain more structured, less textual attributes of the publication domain. The Leipzig DB group also provides four benchmarks for entity clustering. To this end, two of the four datasets from the domains of *geography*, *music* and *person*, were enriched with artificial data, created using data generators [Christen and Vatsalan, 2013, Hildebrandt et al., 2020]. These benchmarks contain structured entity representations with amounts of attributes between three and five.

The Hasso-Plattner Institute maintains the DuDe repository[4] since 2011, providing three benchmark tasks [Draisbach and Naumann, 2017] originally published in other sources, which have been modified for the entity matching setting with the *Cora* benchmark considered more textual and the *Restaurants* and *CD* benchmarks representing very structured matching tasks. The benchmarks are provided together with the ground truth of the matches.

The Magellan repository[5] of the Data Management Research Group of the University of Wisconsin-Madison has been maintained since 2015 and includes a large collection of benchmark tasks from other repositories, as well as benchmarks created at the Research Group. The benchmarks span a wide range of topics and degrees of structuredness. The most prominent ones that have been used extensively in research are the *Walmart-Amazon* (product), *Company* (company), *Beer* (product), and *iTunes-Amazon* (product) benchmarks which apart from the highly textual *Company*, can all be considered structured benchmarks.

As all previously presented benchmarks were created during a time when learning-based systems were not as prominent as they are today, no fixed training, validation, and test splits were originally made available with them. In 2018, the release of the Deepmatcher [Mudgal et al., 2018] matching framework provided public downloads[6] of the fixed splits for many of the presented benchmarks, which have consequently been adopted by the research community, leading to better comparability of the benchmark results across systems in recent works.

The Alaska Benchmark[7] [Crescenzi et al., 2021] of the Roma Tre University released in 2019 is a benchmark for data integration tasks currently providing two

[2] https://paperswithcode.com/task/entity-resolution

[3] https://dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution

[4] https://hpi.de/naumann/projects/data-integration-data-quality-and-data-cleansing/dude.html

[5] https://sites.google.com/site/anhaidgroup/useful-stuff/the-magellan-data-repository

[6] https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

[7] http://alaska.inf.uniroma3.it/

e-commerce benchmarks for public download. The Alaska benchmarks support not only the entity matching task, but also allow for an evaluation of schema matching systems due to the data being crawled from web pages of more than 20 different hosts that use different schemata for their product representations. Consequently, these benchmarks are highly structured to support the schema matching task. Labels are provided in the form of ground truth for matches. No negatives or fixed splits are available for these benchmarks yet, but they are supposed to be provided in the future.

The Web Data Commons (WDC) project maintained by the Web-based systems research group of the University of Mannheim published the WDC Gold Standard for Product Matching and Product Feature Extraction[8] in 2016. The WDC LSPM benchmark presented in Section 4.6 has been released in 2019. The WDC Products benchmark presented in Section 5.3 was released in 2022. Both benchmarks are based on Schema.org product data that has been extracted from different versions of the Common Crawl.

Megagon Labs has published the *Machamp* entity matching benchmark [Wang et al., 2021a] in 2021 which combines structured and unstructured tasks from a selection of the benchmark tasks described above into seven new tasks that exhibit differing degrees of structuredness. The *Machamp* benchmark datasets provide fixed training, validation, and test splits together with the datasets to facilitate reproducibility and comparability.

In 2022, the Chinese-language product matching benchmark Ember [Wang et al., 2022d] has been released. Similarly to WDC Products, Ember also supports evaluating the generalization of matchers to unseen entities. Ember provides various test sets to investigate the impact of a higher negative-to-positive ratio among matches and non-matches and the utility of multi-modal features for product matching.

Data Generators for Entity Matching

In addition to the previously presented benchmarks, which rely on real-world datasets, a selection of data generators for entity matching exists, which allow for the creation of artificial datasets. This paragraph discusses a selection of data generators proposed over the last ten years. All of the presented generators follow the pattern of using a source dataset or repository of source entities, which are then duplicated and corrupted using various transformation functions with varying degrees of corruption on the attributes of the data to generate heterogeneous matches.

The GECO [Christen and Vatsalan, 2013] data generator focuses on generating artificial datasets for the personal data domain using a set of heuristics and produces duplicate records by corrupting the generated data using various string corruption techniques. This generator has produced the *Voters* benchmark in the Leipzig repository.

[8] http://webdatacommons.org/productcorpus/

The EMBench [Ioannou et al., 2013] generator uses real-world datasets as a basis to generate datasets by applying varying transformations to real-world examples to generate a higher/more diverse heterogeneity artificially.

The SWING [Ferrara et al., 2011] data generator focuses on building matching data for knowledge graphs by applying transformations to surface forms or structures of samples of existing knowledge graphs.

The LANCE data generator [Saveta et al., 2015] also focuses on generating additional matching data for knowledge graphs and, in addition to value and structure transformations, also takes into account expressive OWL constructs and can consequently produce test sets of varying levels of difficulty.

DaPo [Hildebrandt et al., 2020] is a more recent data generator based on Apache Spark to create datasets of very large size. It requires input from a source dataset, which can then be duplicated using various corruption techniques. DaPo has been used to generate various sizes of the *Musicbrainz* benchmark in the Leipzig repository.

The ALMSERgen [Primpeli and Bizer, 2022] data generator was created to produce multi-source entity matching tasks along various matching dimensions such as entity overlap, value heterogeneity, and pattern overlap. To this end, it uses an existing benchmark dataset for duplication and value transformations for the multi-source matching task along the given dimensions.

Benchmarks for Blocking

Many of the entity matching benchmarks presented are also used to evaluate blocking methods [Wang et al., 2023, Brinkmann et al., 2024, Papadakis et al., 2020, Papadakis et al., 2016]. Recent work on blocking introduces larger benchmark datasets in order to challenge the scalability of blocking systems [Papadakis et al., 2016, Papadakis et al., 2022]. Due to its size the PDC2020 product corpus presented in Section 5.3.1 is well-suited as starting point for building blocking benchmarks. An example of a blocking benchmark that is derived from the PDC2020 corpus is the recent SC-Block benchmark [Brinkmann et al., 2024][9].

Entity Matching as Non-Binary Classification

While the WDC Products benchmark presented in Section 5.3 provides two versions for pair-wise and multi-class entity matching, related work has nearly exclusively addressed entity matching as a binary pair-wise task. Multi-class classification methods are more prominent in the area of product categorization [Silla and Freitas, 2011, Gao, 2020] which constitutes a higher-level task that aims at groups of entities rather than single entities. Recent work introduced FlexER [Genossar et al., 2023] which discusses the notion of what constitutes a match along a hierarchy of increasing strictness. Multi-class entity matching, as presented in the

[9]https://webdatacommons.org/largescaleproductcorpus/wdc-block/

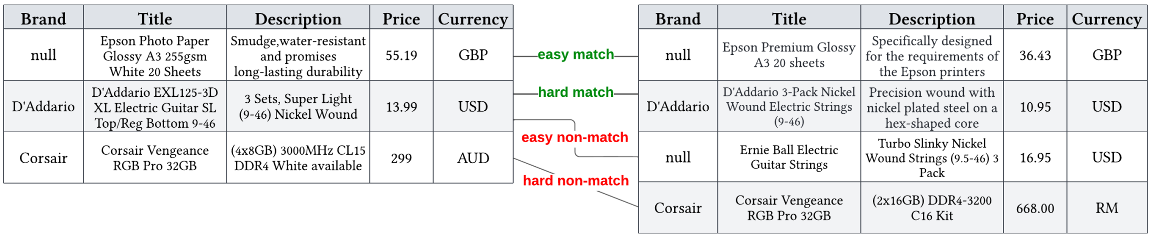


Figure 5.1: Example of hard and easier matching and non-matching offer pairs from WDC Products. Hard matches (non-matches) exhibit a strong textual dissimilarity (similarity) making them harder to classify correctly.

WDC Products benchmark, is a relevant formulation of the task for practitioners, as experiments with proprietary data at eBay [Shah et al., 2018] show.

5.3 WDC Products: A Multi-Dimensional Entity Matching Benchmark

The WDC Products benchmark, as an evolution of WDC LSPM, is the first benchmark that employs non-generated real-world data to assess the performance of matching systems along three dimensions: (1) amount of corner cases, (2) fraction of unseen entities in the test set, and (3) development set size. For this, the benchmark provides nine training, nine validation, and nine test sets, which can be combined into 27 variants of the benchmark. The range of variants includes simple variants that contain only a low amount of corner cases and no unseen entities while enough training data is available. It also includes highly challenging variants that require the system to match corner cases involving unseen entities, while for the seen entities, only a small number of training examples are available. While existing publicly available benchmarks can partly support these dimensions, this usually entails severe limitations, especially for modeling the corner case and robustness dimensions, as discussed in Section 5.4, which can limit the meaningfulness of the obtained results.

WDC Products is based on product data that has been extracted in 2020 from 3,259 e-shops that mark up product offers within their HTML pages using the Schema.org vocabulary. Compared to the WDC LSPM benchmark, WDC Products uses the same basic process to generate a clustering of product offers using a more recent version of the Common Crawl. WDC Products contains 11,715 product offers describing in total 2,162 product entities belonging to various product categories. Figure 5.1 shows examples of hard and easy matches and non-matches sourced from the benchmark.

In order to prevent any potential information leakage between training and testing, the WDC Products benchmark strictly separates offers that appear in the pairs contained in the training, validation, and test sets so that each offer can only be

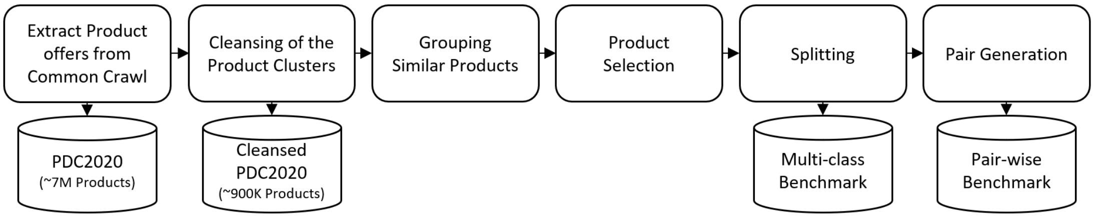


Figure 5.2: Creation process of WDC Products from the extraction of offers from the Common Crawl to the final benchmark.

contained in one of the splits. This splitting also allows the formulation of the benchmark in pair-wise format and as a multi-class matching benchmark. WDC Products is the first benchmark to offer fixed datasets for pair-wise and multi-class entity matching tasks and offers multiple dimensions for a systematic evaluation of entity matching systems. The WDC Products benchmark and the code used for its generation are available for public download[10].

5.3.1 Creation Process

The creation pipeline of the WDC Products benchmark consists of the following six steps: (1) extraction of product offers from the Common Crawl, (2) cleansing of the product clusters, (3) grouping similar products, (4) product selection, (5) splitting, and (6) pair-generation. This section gives an overview of the six steps, which are visualized in Figure 5.2. Detailed explanations of every step are given in the following sections.

The first step of the pipeline is the extraction of large amounts of product offers from the Common Crawl[11] using Schema.org annotations using the same extraction, cleansing, and clustering process as the WDC Training Dataset in Chapter 4. More specifically, the first step is based on a clustering created from the December 2020 Common Crawl[12]. In the second step, several cleansing heuristics are applied to prepare the clustered data for the benchmark. In the third step, similar product clusters are grouped together in order to facilitate the later discovery of corner cases. In the fourth step, several sets of 500 product clusters are selected from the previous grouping to mix and merge for materializing the two dimensions *amount of corner cases* and *unseen products*. For this purpose, the groups are iterated over, and one seed product cluster is randomly selected for each group. To ensure the inclusion of corner cases, similarity searches are performed using different similarity metrics among the product clusters from the previous grouping and the respective selected seed product. For each corner case selection, random drawing is performed from a set of similarity metrics to avoid selection bias. As

[10]http://webdatacommons.org/largescaleproductcorpus/wdc-products/
[11]https://commoncrawl.org/
[12]https://webdatacommons.org/largescaleproductcorpus/v2020/index.html

the basis for the later generation of the unseen dimension, a second distinct set of 500 products is selected for each corner case ratio.

In the fifth step, the offers in the selected clusters are split into training, validation, and test sets. In order to prevent information leakage in the training process, each offer is only assigned to a single set. Depending on the corner case ratio, this splitting is done either randomly or using the previously applied similarity metrics to ensure positive corner cases. At this point, the unseen dimension is also materialized by replacing, for example, 50% of the products in the test split with the products from the second set of 500 *unseen* products. Finally, the development set size dimension is realized by further selecting subsets from the training split of each product to build a medium and small training set. In the sixth and final step, the generated splits are used to generate pairs of offers for the pair-wise formulation of the benchmark. Again, random choice is applied to the same set of similarity metrics already used in Step 4 to ensure an appropriate number of corner case pairs.

Extraction of Product Offers from the Common Crawl

The WDC Products benchmark uses product offers from the WDC Product Data Corpus V2020 (PDC2020)[13]. The corpus was created by extracting Schema.org product data from the September 2020 version of the Common Crawl. The extracted data goes through a pipeline of cleansing steps similar to the one presented in Chapter 4. The resulting PDC2020 corpus consists of ∼98 million product offers originating from 603,000 websites. All of these offers contain some form of product identifier that allows them to be grouped into ∼7.1 million clusters that have a size of at least two offers. An evaluation of the cleanliness of the clusters estimated a noise ratio of 6.9% for clusters smaller than or equal to 80 and around 1.8% for clusters larger than 80. A cluster is considered clean if all contained offers are for the same product. Each product offer in the corpus is described by the textual attributes *title*, *description*, and *brand*, as well as the numerical attribute *price* and a corresponding *priceCurrency* attribute.

Cleansing of the Product Clusters

Some further cleansing steps are applied to the PDC2020 corpus to increase the quality of the product clusters and find a suitable subset for building the benchmark. These steps are detailed in the following paragraphs.

PDC2020 is a multi-lingual corpus with product offers originating from all across the Web, as such it also includes a lot of non-English product offers. As WDC Products is designed to be an English benchmark, some steps are applied to filter out non-English offers. In its original form, PDC2020 contains more than 98M offers and over 7M product clusters with a size of at least two. In the first step,

[13] http://webdatacommons.org/largescaleproductcorpus/v2020/

the fastText language identification model[14] is applied to each row in PDC2020, more specifically to the concatenation of the attributes *title* and *description*. A product offer with a longer textual description should lead to reliable identification as English or non-English for the algorithm. For any offers without description, the language identification has to rely on non-English words in the product titles alone. After applying the fastText model, all rows are kept where the classifier confidence is highest for English. In the second step, a regular expression is applied to identify non-Latin characters in the remaining product offers, and only those that contain less than four non-Latin characters are kept. These are kept as sometimes product descriptions contain non-Latin characters as part of a model name or branding of the manufacturer. Any product offers with more than four non-Latin characters are assumed to be non-English and removed.

In the second cleansing step after language identification, the attributes *title*, *description*, and *brand* are concatenated, and any duplicate rows on this combined attribute are deleted, keeping only the first occurrence. Finally, all product offers are removed where the title attribute contains less than five tokens, as it is expected that offers with such short titles are unsuited for a benchmark due to being very sparsely described, which results in unsolvable ambiguity when trying to match similarly sparse but different products.

The third and last step aims to remove any remaining intra-cluster noise in the form of wrongly assigned product offers. For this purpose, a heuristic based on simple word occurrence is employed among offers inside a cluster. More specifically, each clusters offers are scanned and the average length of *titles* is recorded while at the same time a dictionary of word counts across offers' *titles* is created. Any offer containing very unique words compared to all others in the cluster are assumed to be noisy non-matching product offers and subsequently removed from the cluster.

After these cleansing steps, PDC2020 contains 22M offers and 900K product clusters of size greater than one. This cleansed corpus is used as the base for creating WDC Products and is also available for download on the WDC Products website with the creation code. In combination, these two artifacts allow for the creation of more fine-grained steps for each of the benchmark dimensions and the generation of entirely new benchmarks similar to WDC Products.

Grouping Similar Products

To group similar products for easier discovery of corner case products and reduce computational complexity in later stages, the scikit-learn implementation of DBSCAN clustering[15] is used on the data corpus with an epsilon of 0.35 and min_samples of 1 to determine coarse groups of similar products. Simple binary word occurrences after lower-casing and removing tags and punctuation are used as a feature vector for each product. The values for epsilon and min_samples were

[14]https://fasttext.cc/docs/en/language-identification.html
[15]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

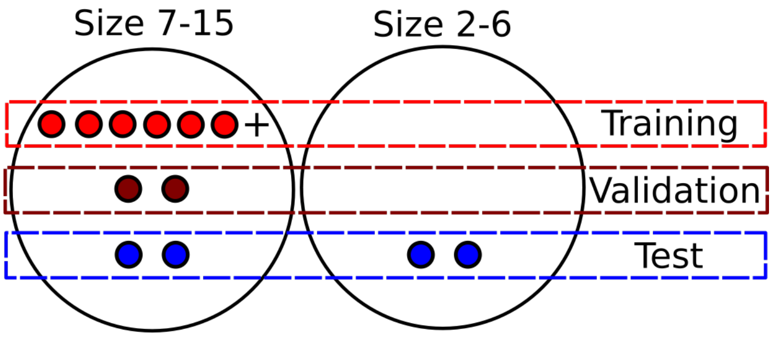


Figure 5.3: Depiction of the cluster sizes and distribution of offers into splits for seen product clusters with size 7-15 (left) and unseen product clusters with size 2-6 (right).

chosen to generate the largest number of groups containing products with at least 7 offers. The minimum number of 7 is chosen because this is the minimum amount needed to cleanly split offers into train, validation, and test at a later stage. A random manual check of the generated groups showed that they contain either only highly similar products, e.g., hard drives from the same vendor, or very similar products, e.g., graphics cards from the same but also other vendors.

In the next step, the corpus with annotated DBSCAN groups is split into two parts, the first one, from which the *seen* portion of the benchmark is drawn and represented by 629 DBSCAN groups, containing all products represented by at least 7 offers and the second one, from which the *unseen* examples are drawn and represented by 2,845 DBSCAN groups, with products having between 2 and 6 offers. In the final step, two domain experts manually check all 629 groups in the first part and annotate them with *useful* or *avoid* depending on the cleanliness of the group. The domain experts then look at the same number of groups for the second part and annotate them in an analogous manner. The results of this process are two sets of grouped and manually curated products from which sets of products are selected in the next step.

Product Selection

For creating WDC Products, multiple sets of 500 products are selected from the cleansed and grouped PDC2020 corpus along three dimensions that characterize the final benchmark. The three dimensions are: (1) amount of corner cases, (2) amount of unseen products, and (3) development set size. The amount of corner cases determines the percentage of the 500 products that have at least 4 textually highly similar products in the set of 500. The higher the amount of corner cases in a dataset, the harder it will be for matching systems to disambiguate between matching and non-matching offers as textual descriptions will be very similar for such products. This dimension is varied in 3 steps: 80%, 50%, and 20%. The amount of unseen products is only relevant for test splits and determines the percentage of products that are represented in the training and validation sets. This dimension is varied from 0% unseen over 50% to 100% unseen. The third dimension varies the amount of available development data in small, medium, and large sizes. These sets of products need to fulfill two characteristics: each product must be represented by at least 7 (seen) or 2 (unseen) unique product offers for the purpose of cleanly splitting them among training, validation, and test sets, and at least four very similar but different products need to be available in the selected set of products for the purpose of generating negative corner cases.

The next paragraph describes the selection of 500 products for the final benchmark datasets while keeping the corner case dimension fixed to 80% corner cases. The process is applied analogously for the other corner case percentages.

400 products are selected in such a way that this number includes at least four negative corner cases for each selected product (80% corner cases). Subsequently, 100 random products (the remaining 20%) are selected from the corpus. This selection process is applied once for the *seen* and once for the *unseen* groups of the corpus, to make it possible to build sets for each unseen percentage later. The selection process starts by iterating through the annotated DBSCAN groups. Inside each DBSCAN group, one product cluster is randomly selected and subsequently the four most similar product clusters inside that group are identified by randomly alternating between the most similar examples on the product *title* according to a variety of similarity metrics: *Cosine*, *DICE* and *Generalized Jaccard* similarities from the py_stringmatching[16] package and a fastText embedding model trained on the titles of the product matching benchmarks *Amazon-Google*, *Abt-Buy*, and *Walmart-Amazon*.

Although it is possible to simply apply a stricter version of the DBSCAN grouping to select corner cases, this would limit the selection to the representation and similarity metric used in this clustering, leading to a benchmark that can be easily solved using the DBSCAN algorithm. By using a varied selection of similarity metrics, biasing the benchmark towards a single method of corner case selection can be avoided while ensuring that it cannot be solved by a matcher based on a single metric. After collecting 400 products in this way for the *seen* and for the *unseen* split, respectively, another 100 products are randomly selected for seen and unseen from the remaining product clusters, bringing both product sets to 500 products with an 80% negative corner case ratio.

Splitting

In this step, the offers of each of the 500 product clusters of the *seen* part are split into training, validation, and test offers. For the *seen* products, the maximum number of selected offers from a product cluster is limited to 15. Exactly two offers are sampled from the respective cluster for the *unseen* products. Figure 5.3 shows how the subsequent splitting of the selected offers of both sets into training

[16]https://github.com/anhaidgroup/py_stringmatching

validation and test offers is performed. The validation and test splits are assigned two offers each, while the remaining offers are assigned to the training split. The selection of offers for each split is done along the corner case dimension, which means that for 80% of the products the same similarity metrics as before are used for all combinations of offers in a product cluster and then sorted by increasing similarity to find positive corner cases. The resulting list is split at the first fifth, and two pairs are randomly selected from the corner case part (low similarity) for inclusion in test and validation splits. The rest are assigned to the training split.

To build the unseen dimension, products in the fully seen test set are systematically replaced with products from the unseen split. To keep the amount of corner cases the same when building the 50% unseen test set, 250 products are replaced with unseen products in a ratio of 80% corner cases and 20% random. Finally, to build the 100% unseen test set, the original test set is fully replaced with the corresponding selection from the unseen split. In summary, three test sets (0%, 50%, and 100% unseen) are generated for the corner case ratio of 80%.

In the last step, datasets for the last dimension, the development set size, are generated. For this purpose, the subsets of each product that were assigned to the training split are further divided as follows: All product offers are assigned to the large training set, three among those are assigned to the medium training set, and two of the three are assigned to the small training set. For 80% of the products, the same positive corner case procedure as before is applied to ensure that the pairs in the small and medium training sets are corner cases.

The same process is repeated again for the corner case ratios 50% and 20%. This procedure results in three test sets and three development sets for each corner case ratio: nine test sets, nine validation sets, and nine training sets overall. At this stage, the multi-class version of the WDC Products benchmark is materialized. For the pair-wise formulation, pairs are generated from the selected product offers in the final step.

Pair Generation

The training, validation, and test splits prepared previously are used as a base to generate pair-wise datasets from the selected products and splits. For each offer in each set, all positive pairs, as well as varying amounts of negative corner cases and random pairs, are created, depending on the development set size, to simulate a blocking process. In the following, this process is explained for the development size *large*.

A product cluster in the large training set can contain between 3 and 11 unique offers. All possible positive pairs are generated using the available offers and included in the final large pair-wise training set. Afterward, for each offer in the product cluster, the three most similar product offers (using alternating similarities as before) among the remaining 499 product clusters in the dataset are selected. If a pair is already contained in the final training set, e.g. because of a mirrored pair, the next most similar pair is added instead. Finally, a random negative pair is added for a total of four negative pairs for each offer in the current product cluster. This process is continued for all 499 remaining product clusters, resulting in the final pair-wise large training set.

The generation of all pair-wise test sets and the large validation sets follows the same process as for the large training sets. For the medium and small training and validation sets, the amount of selected negative corner cases per product offer is reduced to 2 (medium sets) and 1 (small sets) to simulate a reduced labeling effort in addition to the lower amount of available product offers per product for the different training sets (3 for medium and 2 for small).

5.3.2 Benchmark Profiling

Table 5.2: Statistics of the training, validation and test sets of WDC Products along the dimensions amount of corner-cases and development set size for the pair-wise and multi-class matching tasks.

Type	Corner-Cases	Pair-Wise									Multi-Class		
		Small			Medium			Large			Small	Medium	Large
		All	Pos	Neg	All	Pos	Neg	All	Pos	Neg			
Training	80%	2,500	500	2,000	6,000	1,500	4,500	19,835	8,471	11,364	1,000	1,500	2,841
Validation		2,500	500	2,000	3,500	500	3,000	4,500	500	4,000	1,000	1,000	1,000
Test		4,500	500	4,000	4,500	500	4,000	4,500	500	4,000	1,000	1,000	1,000
Training	50%	2,500	500	2,000	6,000	1,500	4,500	19,607	8,339	11,268	1,000	1,500	2,817
Validation		2,500	500	2,000	3,500	500	3,000	4,500	500	4,000	1,000	1,000	1,000
Test		4,500	500	4,000	4,500	500	4,000	4,500	500	4,000	1,000	1,000	1,000
Training	20%	2,500	500	2,000	6,000	1,500	4,500	19,015	7,963	11,052	1,000	1,500	2,763
Validation		2,500	500	2,000	3,500	500	3,000	4,500	500	4,000	1,000	1,000	1,000
Test		4,500	500	4,000	4,500	500	4,000	4,500	500	4,000	1,000	1,000	1,000

The WDC Products benchmark consists of 11,715 unique product offers, which describe 2,162 different products. It further consists of nine training sets, nine validation sets, and nine test sets, each for pair-wise matching and multi-class matching. Table 5.2 shows statistics on the size of the datasets for each of these splits. Each split contains offers for exactly 500 products. All test sets contain exactly 4,500 pairs of offers (pair-wise) or 1,000 offers (multi-class). The sizes of the training and validation sets, on the other hand, vary across the development set size dimension from 5,000 (pairwise) and 2,000 (multi-class), representing the small dataset, to 8,500/2,500 in the medium and ∼25,000/∼4,000 in the large set. This allows for the evaluation of matching systems along the dimension of development set size while ensuring comparability between pair-wise and multi-class tasks, as both datasets always contain the same set of offers in training, validation, and test, as well as no overlapping offers between the training and evaluation splits. In addition to the development set size dimension, each dataset exists in three versions of increasing difficulty, represented by the number of corner cases among the contained products. Finally, for each of the three difficulties, three test sets of the same size are available with an increasing amount of unseen products from 0% over 50% to 100%, representing the third and final dimension of the benchmark.

As explained in Section 5.3.1, care is taken to preserve the corresponding corner case ratio when replacing seen products with unseen products.

Attribute Density and Vocabulary:

Each offer in WDC Products has five attributes, *title*, *description*, *price*, *priceCurrency*, and *brand*. Table 5.3 contains statistics on density and value length for each attribute in all datasets. Most of the attributes have a density of more than 90%, with *description* (75%) only filled in three-quarters of offers and *brand* (35%) being the least dense attribute. Manual inspection revealed that brand information is often found in the *title* attribute if the *brand* attribute itself is missing. The median value lengths of the attributes reveal that *title* can be considered a shorter textual attribute while *description* contains longer strings. *Brand* and *priceCurrency* consist of a single word which in the case of *priceCurrency* is often a three-character identifier for the currency, such as *USD* or *EUR*. The *price* represents the single numerical attribute of the benchmark. An example of each attribute can be seen in Figure 5.1. Table 5.3 further shows statistics for the unique vocabulary used across all datasets, which highlights the heterogeneity of the benchmark, with each dataset containing between 17K and 20K unique words on average, as well as making use of around 12K (24%) tokens contained in RoBERTa's vocabulary of size ∼50K.

Table 5.3: Attribute density and length statistics across the merged sets by development set size and amount of corner cases.

Development Set Size	Corner-Cases	# Entities	Density (%)/ Median Length (# Words)					Vocabulary (unique)	
			title	description	price	priceCurrency	brand	Words	Tokens
Average	-	-	100/8	75/32	93/1	90/1	35/1	18,832	11,876
Small		500	100/8	75/37	94/1	91/1	34/1	16,662	11,420
Medium	80%	500	100/8	75/37	94/1	91/1	34/1	17,612	11,718
Large		500	100/8	75/36	94/1	91/1	34/1	19,862	12,246
Small		500	100/8	76/32	93/1	90/1	35/1	17,232	11,548
Medium	50%	500	100/8	76/32	93/1	90/1	35/1	18,313	11,912
Large		500	100/8	75/32	93/1	90/1	35/1	20,361	12,372
Small		500	100/8	74/29	92/1	90/1	35/1	16,179	11,201
Medium	20%	500	100/8	74/29	92/1	90/1	35/1	17,220	11,563
Large		500	100/8	74/29	92/1	90/1	34/1	19,235	12,028

Label Quality:

As the benchmark labels originate from automatic clustering of offers using annotated product identifiers from the Web, a manual evaluation of their correctness is performed on the test splits of WDC Products. Two domain experts check the match and non-match labels from a sample of labeled pairs which are sampled from all nine available test splits. From each test split, an equal amount of positives and negatives is sampled while the overall amount depends on the corner case percentage of the respective test set. As the test sets with higher corner case ratios contain harder-to-match pairs, this ensures the sampling does not favor easily disambiguated pairs. Specifically, 100/60/40 pairs are sampled for corner case percentages of 80/50/20, resulting in 600 (300 positives, 300 negatives) labeled pairs in total. Overall, the noise level in the sample is estimated as 4.00% by the first annotator and 4.17% by the second annotator with a Cohen's Kappa of 0.91.

5.4 Comparison to Existing Benchmarks

Table 5.1 compares the WDC LSPM and WDC Products benchmarks to publicly available benchmark tasks along various profiling dimensions. Among the benchmarks presented, the WDC LSPM and WDC Products benchmarks are the only ones that originate from hundreds and thousands of sources, in this case, e-shops, underlining the diversity of both benchmarks compared to the existing benchmarks that originate primarily from two sources.

WDC Products is the only multi-dimensional benchmark that provides 27 variants along the three dimensions, amount of corner cases, unseen entities in the test set, and development set size, which allow for the systematic comparison of matching systems. Most other benchmarks consist of a single variant apart from the WDC LSPM benchmark, which allows for the evaluation along the two dimensions of development set size and product category. Varying degrees of difficulty in the form of corner cases and separate test sets for unseen entities are original features of WDC Products not found in other benchmarks. WDC Products is, furthermore, the only benchmark in this comparison that offers a multi-class formulation of the entity matching task. This is possible due to the constant amount of available matches per entity across the full set of products included in the benchmark, which provides enough offers for each product to cleanly split them into training, validation, and test sets. Although the WDC LSPM benchmark also has a high amount of average matches per entity, they are not equally distributed across all contained products due to differences in the creation process and fewer available product offers at the time of its creation. Comparing the maximum F1 reported for each benchmark in related work shows that WDC Products is challenging for matching systems. The WDC LSPM and WDC Products benchmarks expand the space of publicly available benchmarks by allowing for an extensive evaluation of multiple dimensions for the product domain of entity matching that was not possible before, as discussed in the next section.

Adaptation of Existing Benchmarks

This section discusses whether existing benchmarks already cover the dimensions of the WDC Products benchmark. It also explores the feasibility of extending existing benchmarks to implement these dimensions and points to problems that hinder such an adaption.

Development Set Size: Most of the benchmarks listed in Table 5.1 provide a single fixed development set size (see column *Fixed Splits* in Table 5.1). The only benchmark besides WDC Products that defines development sets of different sizes is the WDC LSPM benchmark. It is possible to downsample the training and validation sets of all benchmarks in order to generate development sets of different sizes. A problem that arises from the downsampling is that most benchmarks except Alaska and the WDC benchmarks only contain a single match for most entities (see column *Avg. Matches per Entity* in Table 5.1). Downsampling these benchmarks will increase the amount of entities that are not covered in the training set (unseen dimension), making the benchmarks more difficult and blurring the distinction between the two dimensions.

Amount of Corner Cases: None of the publicly available benchmarks implements the dimension amount of corner cases nor offers test sets with different difficulty levels. The corner case dimension is realized in WDC products by careful selection of sets of 500 products from the PDC2020 corpus using the set of similarity metrics described in Section 5.3.1. The large amount of products contained in PDC2020 ensures the availability of very similar but distinct products. Although a similar dimension can be realized using the English WDC Training Dataset for the WDC LSPM benchmark, this was not considered at the time of creation but remains a possible addition in future work.Most publicly available benchmarks are not accompanied by a product corpus that can be used to select additional similar products. To include a corner case dimension in existing benchmarks, filtering out some entities to create easier versions of the benchmark would be a possibility. However, reducing corner cases in this way would entail reducing the overall size of the benchmarks, which would blur the distinction between the development set size and the corner case dimensions.

Unseen Entities: The dimension unseen entities is inseparably linked to the splitting of product offers into development and test sets. For most existing benchmarks, namely the Magellan and Leipzig benchmarks, the community has adopted a set of splits that were created using random splitting[17] of product pairs after blocking, which does not explicitly consider the unseen dimension. Given the low amount of average matches per entity in Table 5.1 for these benchmarks, it is possible that by chance all offers for a specific product end up only in the test set resulting in an unseen entity. The explicit modeling of this dimension requires knowledge of all matching records in a dataset, or, if not available, the reduction of the dataset to the set of known matches, which would severely reduce the size of a benchmark. Followed by a careful splitting procedure, it would be possible to explicitly model this dimension in the existing benchmarks, albeit on a much smaller scale than WDC Products. The WDC LSPM benchmark is designed to not include any unseen entities as part of the default available test sets. A subset of the MWPD challenge test set included in WDC LSPM as presented in Section 4.6.2 allows the evaluation on 50 unseen products which is on a much smaller scale than

[17]https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

the 500 available in WDC Products.

Multi-Class Formulation: The formulation of entity matching as a multi-class task is not explicitly available in any publicly available benchmarks apart from WDC Products. Evaluation of a multi-class task requires the availability of a relevant amount of examples for each class, the classes being, for example, product identifiers in the case of product benchmarks, that further need to be split across development and test sets. The low amount of available matches per entity in existing benchmarks prohibits formulating these in a multi-class format.

Multi-Dimensionality: In summary, while all dimensions can be realized to a certain extent in some of the publicly available benchmarks in isolation, it is essential to note that the dimensions interact with each other. For example, classifying unseen entities in an environment of low corner cases can be significantly easier than classifying unseen entities when many corner cases are present, as the results in Section 8.4.3 show. Realizing a single one of the chosen dimensions is often already prohibitive in publicly available benchmarks, as discussed above. Realizing all of them in order to explore their interaction is practically infeasible for most benchmarks. WDC Products offers the possibility of exploring such nuances along its 27 variants and is further supported by the large PDC2020 corpus that can be used as a starting point for extending the benchmark or exploring further dimensions.

5.5 Conclusion

This chapter presented the WDC Products benchmark based on Schema.org annotations found on the public Web. The process presented in Chapter 4 and an updated clustering of product offers using annotated product identifiers such as MPNs and GTINs from the December 2020 Common Crawl served as the starting point for the creation of the benchmark.

WDC Products complements the field of benchmarking entity matching systems with a new multi-dimensional benchmark consisting of real-world product offers originating from thousands of e-shops annotated in the Common Crawl of December 2020. The benchmark provides 27 variants for the comparison of systems along the dimensions (1) amount of corner cases (2) generalization to unseen entities, and (3) development set size. In comparison to other benchmarks, WDC Products contains a larger amount of examples per entity which enabled the modeling of the entity matching task as a binary pair-wise classification task on the one hand and on the other hand as a multi-class matching task. This makes WDC Products the first entity matching benchmark which provides for the fine-grained evaluation of matching systems along three dimensions, as well as the first benchmark to offer evaluation setups for both pair-wise and multi-class matching. WDC Products is further accompanied by a large product corpus that can be used to extend the benchmark or explore additional dimensions.

Part III

Deep Neural Networks for Entity Matching

Chapter 6

Cross-Language Learning for Entity Matching

6.1 Introduction

Transformer-based entity matching methods [Barlaug and Gulla, 2021] have significantly improved the state-of-the-art for textual matching tasks such as matching product offers in e-commerce. To excel at these tasks, Transformer-based matching methods require a decent amount of training data, as shown in Section 4.7.4. Collecting enough training data can be challenging and costly. Section 4.4 showed that semantic annotations can be leveraged for this purpose for the product domain by using annotated product identifiers to cluster product offers for the same products and subsequently automatically build training pairs. If a matcher for non-English product descriptions should be learned, it can be difficult to find enough offers in the respective target language on the Web for less widely used languages and less commonly sold products.

This chapter contributes an evaluation of the utility of English language offers for training product matchers for less widely spoken target languages, such as German, to the field of supervised entity matching. To this end, experiments are carried out with training sets that combine a large number of English language record pairs with a smaller number of training pairs in the target language. Additionally, the experiments encompass matchers that internally rely on different mono- as well as multi-lingual pre-trained Transformer models, including BERT [Devlin et al., 2019], a German version of BERT[1], multilingual BERT[2], XLM-R [Conneau et al., 2020], as well as on an SVM classifier.

The experiments show that extending the German training set with English pairs is always beneficial. The impact of adding English pairs is especially high in low-resource settings, where only a small set of German pairs is available. When this contribution was made, no other work had investigated cross-lingual product

[1]https://github.com/dbmdz/berts

[2]https://github.com/google-research/bert/blob/master/multilingual.md

matching with Transformer models. Since then, more exploration has been done on this topic in languages other than German, which is presented in Section 6.2.

The contribution of this chapter is:

- **Evaluation of Cross-Lingual Fine-Tuning for Product Matching:** An evaluation of the cross-lingual product matching performance of mono- and multilingual PLMs in English and German languages. The evaluation shows that by combining English language training pairs and a small number of training pairs in the German target language, it is possible to learn product matchers that achieve F1 scores above 90%, outperforming matchers that were trained using only a small amount of product offers in the target language.

Section 6.2 presents related work on cross-lingual entity matching. Section 6.3 introduces the datasets created for the experiments. Section 6.4 gives an overview of the different matchers and the training process, while Section 6.5 presents and discusses the results of the experiments. Finally, Section 6.6 concludes this chapter with a review of the results and findings.

The work presented in this chapter has previously been published in [Peeters and Bizer, 2022a]

6.2 Related Work

This section gives an introduction to related work for cross-language learning and applications for the related field of entity linking as well as entity matching.

Cross-language Learning

Language understanding between different peoples has been a cornerstone of trade and social exchange since the dawn of civilization. This has become even more relevant in today's highly globalized society. The natural language processing research field, from which many of the methods in state-of-the-art entity matching are inspired, contains many subfields that research cross-lingual tasks, for example, machine translation, which has become ubiquitous due to the developments in smartphone technology. Early machine translation tasks were based on statistical models [Lopez, 2008] like Markov chains, while the resurgence of neural networks in the 2010s has moved the field nearly completely towards these models with the Transformer architecture being the foundation of state-of-the-art models in this area [Dabre et al., 2020]. In addition to machine translation, cross-language learning is relevant in fields such as text processing [Pikuliak et al., 2021] and text summarization [Wang et al., 2022a]. Multi-lingual Transformers that revolutionized the field of machine translation are usually pre-trained on (aligned) multi-lingual text and are fine-tuned either with or without using explicit cross-language alignments from multi-lingual dictionaries [Ruder et al., 2019, Conneau et al., 2020, Wang et al., 2018].

Cross-language Learning for Entity Linking

Cross-lingual matching is a task that is relevant in many fields, for example, cross-lingual entity linking between knowledge graphs which is related to entity matching, is an active research field [McNamee et al., 2011, Sil et al., 2018, Sevgili et al., 2022]. Chen et al. [Chen et al., 2017] claim to be the first to experiment with cross-lingual alignment of knowledge graphs. In their follow-up work [Chen et al., 2018], they experimented with learning cross-lingual embedding models for entity linking learned on multilingual data from Wikipedia. Wang et al. [Wang et al., 2018] use convolutional neural networks to train an entity linking model that leverages prealigned entities during training to learn an embedding model that is subsequently used for aligning entities by distance in the embedding space. Pei et al. [Pei et al., 2020] train a cross-lingual alignment model by combining adverserial noise detection with a graph neural network-based encoder. More recent methods in the field have almost exclusively moved to using the Transformer architecture [Sevgili et al., 2022]

Cross-language Learning for Entity Matching

In the entity matching field, Rinser et al. [Rinser et al., 2013] leverage inter-language links in Wikipedia to build a linked graph of entities in different languages combined with a top-down approach of stricter connectivity measures to remove erroneous links. Subsequently, they match entities using similarity measures for each attribute together with a matching threshold achieving F1 scores above 90% F1. Their work builds on previous methods for entity matching between languages in Wikipedia based on connectedness measures in graphs [Adar et al., 2009, Hassan and Mihalcea, 2009, Hecht and Gergle, 2010]. Narducci et al. [Narducci et al., 2013] use the bag-of-words language model combined with cosine similarity to match e-government services to services from other administrations in different languages that are already linked to the Linked Open Data cloud.

To the best of the authors' knowledge, the contribution of this chapter was the first to explore cross-language learning for product matching with Transformers. The following related work was released after the contribution of this thesis:

Alves et al. [Alves et al., 2024] explore cross-lingual learning for product matching by using English product offers to train classification models for transfer to the Portuguese target language using various mono- and multi-lingual Transformers as well as various cross-lingual transfer strategies such as joint and cascade learning. They use the computers category of the WDC LSPM benchmark together with a Portuguese dataset they create themselves by crawling Portuguese product websites and using GTIN and EAN numbers as distant supervision. The authors experiment with an experimental setup similar to the one presented in the following sections by training on a selection of source and target language data which they call *joint learning*. Another learning strategy, *cascade learning*, first fine-tunes on a source language while subsequently starting a second fine-tuning on target language data. They show results similar to those presented in this thesis, stating that it is possible to achieve high scores of 90% F1 by training in a source language and only a few examples of the target language, especially for the multi-lingual XLM-RoBERTa and multi-lingual BERT models. They state that having target language data available for training is important as their zero-shot experiments resulted in comparably low results with a maximum of approximately 80% F1 for multi-lingual BERT.

Możdżonek et al. [Możdżonek et al., 2022] prepare a product matching dataset for the Polish language similar to the WDC LSPM benchmark datasets for two product categories. The authors show that they can achieve a performance of 85% F1 with small training set sizes (1.5K to 2K pairs) and up to 93% F1 with large training set sizes (10K-14K pairs) with multilingual BERT as well as XLM-RoBERTa when training with polish training pairs. They further apply both multi-lingual models to the WDC LSPM benchmark and compare them to results of the Ditto [Li et al., 2020] and Deepmatcher [Mudgal et al., 2018] systems, showing that the multi-lingual models achieve comparable performance to mono-lingual Transformers in the English language, with improved performance (2-6% F1 depending on product category) for the small training sizes of the WDC LSPM benchmark. Their work, while employing multi-lingual transformers on product matching tasks in two different languages, does not directly study cross-lingual fine-tuning in the sense of transferring matching knowledge gained from a different language to a target language.

6.3 Datasets

The datasets used as a basis for the experiments in this chapter were created as part of a student project[3]. The experiments in this chapter make use of a selection of these datasets for the product category mobile phones containing English- and German-language offers. The offers have been crawled from 66 different e-shops, auction platforms, and electronic marketplaces. Each offer contains a title, a description, and a product identifier such as a GTIN or MPN number. The data was collected using 150 mobile phones as seeds for the crawling process. These seeds contain widely-sold head products but also less-sold long-tail phones. In addition to offers for the seed phones, the dataset also includes offers for additional phones that have been discovered during the crawling process. The offers are grouped into pairs by using shared GTIN, EAN, and MPN numbers as distant supervision, similar to the clustering process presented in Section 4.4.

[3]https://data.dws.informatik.uni-mannheim.de/Web-based-Systems-Group/StudentProjects/2020/Cross-lingual-Product-Matching-using-Transformers/

title_left_offer	title_right_offer	lang
Samsung Galaxy S10 Factory Unlock (Prism Black 128GB)	Samsung Galaxy S10 sim-free in black 128 GB storage	en
Samsung Galaxy S10 prisma-schwarz 128GB ohne Vertrag	Samsung Galaxy S10 128GB schwarz ohne Simlock	de

Figure 6.1: Example of a matching product offer pair in English (top) and a pair for the same product in German (bottom).

The identifiers are not used during the experiments to prevent the matching from being trivial and to represent a real-world matching use-case for which identifiers are usually only available for a subset of offers or not at all. Non-matching pairs are created by combining an offer for a seed product with an offer for a similar seed product or a similar offer from a phone that has been discovered during crawling. The pairs are arranged into language-specific training sets of different sizes. The training sets range from 450 to 7200 pairs and contain 50% matches and 50% non-matches. Figure 6.1 shows an example of a pair of English product offers and a pair of the same product in German. For the experiments, the German-language test set containing 1200 pairs (25% matches and 75% non-matches) is used. None of the pairs in the test set is included in the training sets. Half of the pairs in all sets were chosen randomly, while the other half contains corner case pairs, measured and created by the students using cosine similarity. Figure 6.2 shows the distribution of positive and negative pairs in the German test set for the 150 seed mobile phones. The datasets represent a seen matching scenario as for each product contained in the test set, offers for that product also exist in the training set.

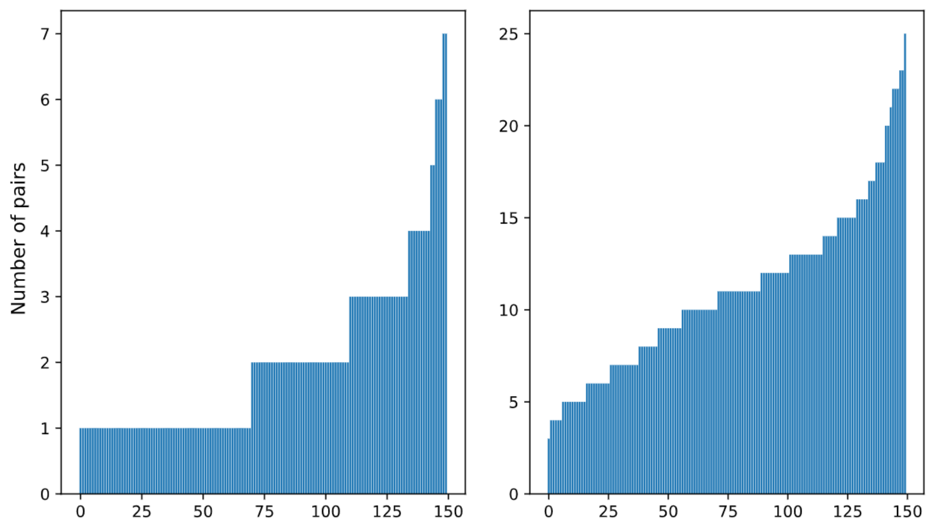


Figure 6.2: Distribution of positive (left) and negative (right) pairs in the German test set for each of the 150 seed mobile phones.

6.4 Models and Baselines

The experiments encompass three pre-trained Transformer-based models from the HuggingFace[4] library. The BERT base model (*bert-base-uncased*) represents a monolingual English option. This model was pre-trained on a dump of the English Wikipedia and BookCorpus. The second model is a German BERT model (*bert-base-german-dbmdz-uncased*) pre-trained on diverse German texts, including a Wikipedia dump, parts of the Common Crawl, and the EU Bookshop corpus. As multilingual models, multi-lingual BERT (*bert-base-multilingual-uncased*), which is trained on the top 100 largest Wikipedias, as well as XLM-RoBERTa (*xlm-roberta-base*), which is trained on a CommonCrawl corpus consisting of 100 different languages, are chosen. This selection of models allows the examination of the performance differences that result from pre-training using multi-lingual texts compared to pure mono-lingual pre-training. The simple word co-occurrence baseline using an SVM classifier that has been used in previous experiments in Section 4.7.1 is used as a non-neural baseline.

Input sequences for the Transformer-based models are created by concatenating the title and description attributes of a product offer into one string and applying the respective tokenizer to both product offers in a pair to represent them in the standard input representation for Sequence Classification, i.e., "[CLS] Product 1 [SEP] Product 2 [SEP]" for BERT-based models. As input for the SVM baseline, a bag-of-words word vector representation indicating co-occurring words in a product pair is generated, which serves as input for the classifier.

For every experimental run, the learning rate is optimized in the range between 5e-6 and 1e-4 with early stopping. The run is stopped if a given model does not improve for three consecutive epochs during hyperparameter tuning on the validation set. Models are fine-tuned for 25 epochs. The batch size is fixed to 16 and the weight decay in the optimizer to 0.01. All other hyperparameters are left at their default values. The scores reported are averages over three runs that were individually trained using the same hyper-parameter setup.

6.5 Results and Discussion

The first set of experiments compares the performance of the different mono- and multi-lingual models on the German test set while training on the one hand with only 1800 German training pairs and on the other hand with the same 1800 German pairs and an additional 7200 English pairs. Table 6.1 shows the results of this experiment. When fine-tuned with only 1800 German pairs, the English BERT model scores the lowest overall, falling 6% F1 behind the SVM baseline at 65% F1, showing that the language mismatch between English pre-training and German fine-tuning has a severe negative impact. The German version of BERT can instead improve on the SVM by 2.5% F1, showing the importance of German lan-

[4]https://huggingface.co/transformers/

guage pre-training for German tasks. The multilingual XLM-R achieves a result comparable to that of German BERT. Multilingual BERT achieves the best result with 87% F1, outperforming German BERT by 14% F1. After extending the German training set with the additional 7200 English training pairs (Row *F1 with EN* in Table 6.1), all models apart from the SVM improve significantly. German BERT sees the highest improvement with a gain of 16% F1, putting it 1.5% points behind multilingual BERT, which achieves the highest score of 91.4% F1.

Table 6.1: Results on the German test set with and without additional English training data. Training sizes: EN - 7200, DE - 1800.

	SVM	BERT	gBERT	mBERT	XLM-R
F1 without EN	71.00	65.27	73.43	87.69	73.40
F1 with EN	71.05	74.29	89.83	**91.44**	86.98
Difference	0.05	9.02	16.40	3.75	13.58

These results show that adding training data in a high-resource language like English, which can be automatically collected, for example, by the process presented in Chapter 4, to a smaller amount of training data in the target language leads to improvements for all Transformer models and, as a result, is a promising course of action for matching tasks in low-resource languages.

Table 6.2: Results on the German test set when varying the amount of training data for both languages.

EN / DE	0	450	900	1800	3600	7200	Δ 0-7200
450	67.11	72.79	75.44	80.83	86.82	87.97	20.86
900	75.76	75.10	74.00	87.67	88.92	88.19	12.43
1800	87.69	88.43	88.38	90.17	90.72	91.44	3.75
3600	93.63	92.98	92.46	93.97	93.25	94.46	0.83

In a second set of experiments, multi-lingual BERT is trained with different combinations of training data sizes for both languages to understand how much training data in the target language is required and how much English training data needs to be added to reach a high performance level. Table 6.2 shows the results of the experiments. For German training sets of size 900 and less, any combination of training datasets resulting in an overall amount of less than 2000 pairs leads to an F1 around 75% F1 or less regardless of the composition among languages. If the training set consists of more than 2000 pairs, an F1 greater than 80% is achievable in all scenarios. If training data in the target language is scarce (450 pairs), adding English training pairs has a significant effect, leading to a large improvement in every step up to 3600 additional English pairs (86.82% F1). After this point, doubling the amount of English training data only yields an improvement of 1% F1.

The beneficial effect of additional English training data is visible across all settings, although it diminishes with increasing size of the German training set. Once the German training set reaches a size of 1800 pairs, the effect of adding English training data is no longer as strong as before but still results in an overall improved model, reaching a maximum of 94.5% F1 when trained using 3600 German and 7200 English pairs.

6.6 Conclusion

This chapter contributed an evaluation of cross-lingual fine-tuning using mono- and multi-lingual Transformer for the task of product matching to the field of supervised entity matching. The presented work has shown that the performance of Transformer-based product matchers for potential low-resource languages, in this case German, can be significantly improved by adding English-language offer pairs to the training set. The impact of adding the English pairs is especially high for low-resource settings in which only a small number of non-English pairs is available. Furthermore, for successful cross-language learning during fine-tuning, a Transformer model like multilingual BERT, that has also been pre-trained on large amounts of text in different languages, achieves a higher performance compared to mono-lingual models in English or the target Language German. Given that training pairs in high-resource languages such as English can automatically be extracted from the Web by exploiting Schema.org annotations (see Section 4.4), cross-language learning can contribute to reducing labeling costs in many low-resource language matching scenarios.

Compared to previous work, Rinser et al. [Rinser et al., 2013] used interlanguage links in Wikipedia to align entities before the matching step, which is based on similarity calculations per attribute and a matching threshold. For the presented product matching case from many web sources, such an alignment graph via interlanguage links is not available. A similar graph could theoretically be created using product identifiers such as GTINs and MPNs similar to the clustering performed in Section 4.4, but in real-world use cases, these identifiers are often only available for a small subset of product offers to be matched. The matching step with attribute-wise similarity metrics would require an information extraction step as the product offers are highly textual, and as previously shown in Section 4.7, similarity-based features do not achieve the same matching performance level as neural networks in general and Transformers in particular. A similar conclusion follows for other previous works that are based on graphs [Adar et al., 2009, Hassan and Mihalcea, 2009, Hecht and Gergle, 2010] or similarity-based features [Narducci et al., 2013].

Chapter 7

JointBERT: Dual-Objective Fine-tuning for Entity Matching

7.1 Introduction

Practical data integration settings often require the integration of datasets from multiple sources, while these datasets themselves are not deduplicated. As a result, there may be multiple records in each dataset that describe the same entity. These groups of records contain diverse descriptions that, in unison, give a holistic view of the entity. In some use-cases shared identifiers are available for a subset of the records to be integrated, which allow for the identification of these entity groups. For instance, in the product domain some e-shops annotate product offers with GTIN numbers while others do not. The same applies to financial information providers, which might identify companies using DUNS numbers or libraries that provide ISBN, LCCN, GND, or ORCID identifiers. As not all records are accompanied by an identifier, which would make the matching step trivial, the challenge in these settings is to learn a matcher for records not containing an identifier using the records having an identifier as training data.

The matching tasks in these use-cases often involve a set of popular head entities for which many records, including identifiers, are available from different sources. However, the tasks also involve long-tail entities for which hardly any data is available. The users of applications that build on the integrated data expect data describing head entities to exhibit hardly any integration errors, e.g., in the context of a price portal or electronic marketplace, users would likely expect data describing a widely-sold phone to be correct, while mistakes for offers for a tail product would not be relevant to most customers. This means that matching methods should excel on both head and tail entities, while taking full advantage of the large amounts of training data that are available for head entities.

This chapter contributes the JointBERT method to the field of supervised entity matching, which combines a binary matching and a multi-class matching objective into a dual-objective training approach for the BERT model that can better exploit the training data available in multi-source matching scenarios as described above, leading to improved overall matching performance. In addition to the binary matching objective, the model is tasked with predicting the entity group of each of the two records in a pair during training. The idea behind this dual-objective training is to force the model to perceive the matching task not only as a comparison of two isolated records but also to incorporate information from other records of the respective entities seen during training into the matching decision. When training data is scarce for each entity in a pair, i.e., for long-tail entities, the model can still rely on learning patterns using the binary matching objective. The entity matching methods available at the time of this contribution have not directly incorporated information about the entity groups as part of the method.

The JointBERT method is experimentally compared with BERT [Devlin et al., 2019], RoBERTa [Liu et al., 2019b], Ditto [Li et al., 2020], Deepmatcher [Mudgal et al., 2018], Magellan [Konda et al., 2016], and the word co-occurrence baseline used in the previous chapters on five entity matching benchmark datasets. The experiments show that the dual-objective trained JointBERT can improve on single-objective classifiers by 1% to 5% F1 for seen entities, given that enough training data for both objectives is available. The results further show that JointBERT performs only slightly worse than BERT in cases where multi-class training data is scarce, i.e., entity groups are small.

To gain a deeper understanding of the strengths and weaknesses of the dual-objective training approach, the learned models are evaluated on the matching challenges in the MWPD test set of the WDC LSPM benchmark, such as dropped tokens and typos in the entity descriptions as well as records pairs involving new entities unseen during training. The code for replicating these experiments is available on GitHub[1].

The contributions of this chapter are:

- **JointBERT - A Dual-Objective Fine-Tuning Method for Entity Matching:** This chapter introduces JointBERT, a dual-objective supervised training method for entity matching with BERT that combines binary pair-wise entity matching with multi-class classification by leveraging information about entity groups in the training data. At training time, the method predicts the entity group each record in a pair belongs to in addition to the pair-wise matching decision, whereas existing entity matching methods do not directly use information about the entity groups.
- **Evaluation of JointBERT Compared to Single-Objective Methods:** The JointBERT method is experimentally compared with various other entity matching methods showing that dual-objective training performs best for seen entities, given that enough training data for both objectives is available. The strengths and weaknesses of dual-objective training are evaluated

[1] https://github.com/wbsg-uni-mannheim/jointbert

and compared to other methods using the matching challenges found in the MWPD test set of the WDC LSPM benchmark.

Section 7.2 presents related work on multi-objective entity matching methods. Section 7.3 presents the JointBERT method and its architecture. Section 7.4 presents the experiments and a discussion of the results compared to existing methods. Finally, Section 7.5 concludes this chapter with a review of the results and contributions.

The work presented in this chapter has previously been published in [Peeters and Bizer, 2021].

7.2 Related Work

This section gives an introduction to related work for multi-task learning, its application to related matching tasks, and entity matching.

Multi-task Learning

Multi-task learning (MTL) [Caruana, 1997, Ruder, 2017, Zhang and Yang, 2021], which is closely related to the dual-objective fine-tuning approach presented in this chapter, has a long history in natural language processing [Ruder, 2017] and serves the purpose of complementing a main task using auxiliary training tasks or directly training for multiple main tasks in order to obtain better representations compared to single task training. With the ubiquity of neural models, MTL has seen a resurgence, with BERT [Devlin et al., 2019] and other Transformers [Lan et al., 2020] using a form of MTL for pre-training for various tasks like aligning natural language text with structured table formats [Yin et al., 2020] or question answering via knowledge bases [Wang et al., 2021b]. Liu et al. [Liu et al., 2019a] presented a method to jointly train the lower layers of a BERT model using multiple objectives such as sentence classification, sentence similarity, and sentence ranking, resulting in improved overall performance compared to single-task training.

Multi-task Learning for Matching Tasks

Various approaches exist in the related field of entity linking that apply multi-task learning. For example, Deng et al. [Deng et al., 2020] pre-train a tabular transformer for various tabular tasks, including entity linking using two tasks, masked language modeling, and masked entity recovery. Mrini et al. [Mrini et al., 2022] apply multi-task learning specifically for the case of entity linking by training with two additional auxiliary tasks that learn mention detection and prediction reranking in combination with the main task.

Suhara et al. [Suhara et al., 2022] learn a joint model for column type prediction and column relation extraction by constantly switching between both tasks at training time. Shraga et al. [Shraga et al., 2020] apply multi-task learning to learn a schema matching model based on similarity matrices consisting of two interacting neural networks, one adjustor of the matrix and an evaluator, that are jointly trained.

Multi-task Learning for Entity Matching

The JointBERT method presented in this chapter is inspired by some of the earlier presented work, especially in the area of natural language processing [Liu et al., 2019a]. At the time of its contribution it was the first method that applied multi-objective training for entity matching with Transformers. Relevant neural entity matching methods at the time of contribution, such as the Deepmatcher framework [Mudgal et al., 2018] and the ground-laying works for Transformer-based entity matching by Brunner and Stockinger [Brunner and Stockinger, 2020] and the Ditto framework [Li et al., 2020] used single-objective training (binary matching decision) and did not make use of entity group information during training. The following paragraphs present some work which applies multi-task learning to entity matching, and was presented after the contribution of JointBERT.

Genossar et al. [Genossar et al., 2023] discuss the notion of what constitutes a match along a hierarchy of increasing strictness. They state that depending on the application domain, a pair of records may also be labeled as a match even if not referring to the same entity but closely related entities, such as for example all sneakers from the brand *Adidas*. The multiple intents entity resolution (MIER) method they propose combines multiple of these intents using a multi-label approach which uses Transformers and a superimposed graph neural network for message passing among all of the intents in an effort to improve results for each of them. The model is jointly trained on all intents, resulting in improved performance on the chosen benchmarks.

Zhang et al. [Zhang et al., 2024b] presented EMBA in 2024, which applies multi-task learning using an attention-over-attention mechanism on individual token representations for improving entity matching. This method is highly similar to the method presented in this chapter as the authors primarily compare to the JointBERT method in their paper. Compared to the method presented in this chapter, EMBA does not use the [CLS] token as a representation of a record pair for both objectives, but instead directly leverages the embeddings of each individual token which are used for predicting the individual entity group each record belongs to separately. The final matching decision for a pair is the result of the *attention-over-attention* module, which combines all token embeddings of both records using an additional attention operation which is subsequently passed to a linear layer for the binary decision. The authors can improve the results of JointBERT using this adaptation of the model architecture (also see Section 7.5).

Fan et al. [Fan et al., 2024a] presented *Unicorn* in 2024 which is a multi-task matching model for entity matching. It consists of a trained encoder that converts any pair of records into an embedding representation and a matcher, which is a binary classifier for the matching decision for the pairs of records. Their method additionally uses the mixture-of-experts ensembling technique, which combines various submodels, each with different task specialization, using a gating network to improve matching results. The tasks they use for training are entity matching, column type annotation, schema matching, entity linking, entity alignment, string matching, and ontology matching. They project pairs for each task into the same embedding space using the DeBERTa encoder [He et al., 2020], learn a mixture-of-experts to align the pair representations in a shared matching space, and train a classifier for the decision end-to-end. Their results show improvements for various matching tasks, although the entity matching task specifically sees only marginal improvements compared to the Ditto framework [Li et al., 2020]. Additional experiments with zero-shot inference suggest a strong generalization performance of the model. However, this is only studied on a single dataset and thus cannot be generalized to hold for all datasets without more analysis. Compared to JointBERT, which focuses solely on entity matching, *Unicorn* does not make use of entity group information but uses different related matching tasks to improve the results for all tasks in unison.

7.3 Dual-Objective Training of BERT

The JointBERT method assumes that the following two requirements concerning the training set are fulfilled:

1. The training examples consist of pairs of entity descriptions which are accompanied by pairs of entity identifiers that reference the entity group each description belongs to.
2. The training set contains multiple entity descriptions for many of the described entities, i.e. entities are described by multiple diverse descriptions referencing the same entity.

These two requirements are typically fulfilled in multi-source entity matching settings where some data sources provide entity identifiers, such as GTIN, ISBN, DUNS, or ORCID numbers, together with the entity descriptions.

JointBERT Model Architecture:

To input an entity description pair into BERT in entity matching, the string representations of two entity descriptions are usually concatenated using BERTs *[CLS] Sequence 1 [SEP] Sequence 2 [SEP]* input scheme for sequence classification tasks [Li et al., 2020]. Sequences 1 and 2 consist of the concatenated attribute values of the respective entity description, optionally including the attribute name as well. The pooled output representation of the [CLS] token from the last encoder layer is then used as input for a linear classification layer followed by sigmoid or softmax to obtain the final class probabilities, in the case of entity matching, for the two classes *match* and *non-match*. The [CLS] token is used as a representation of the two sequences, trained for a specific task [Devlin et al., 2019].

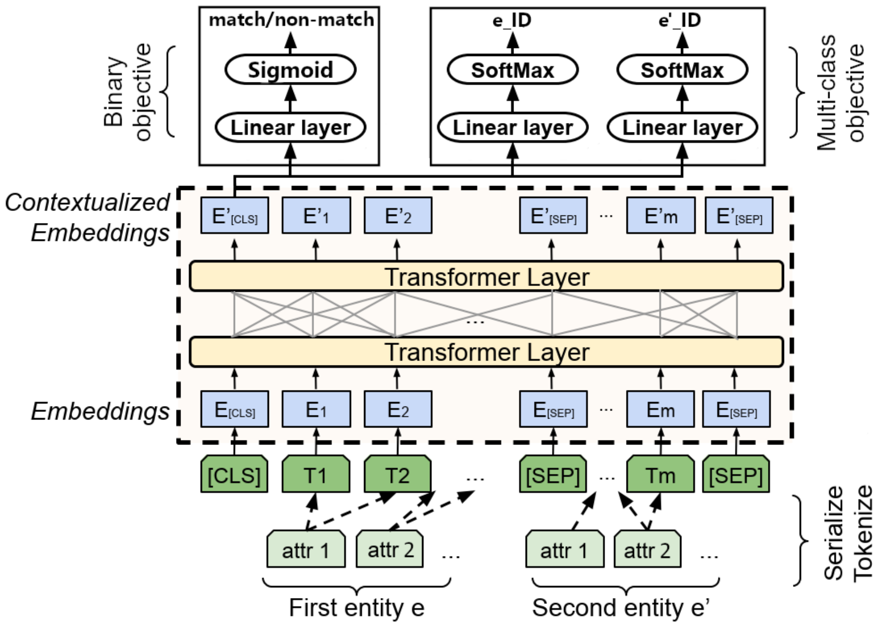


Figure 7.1: The JointBERT architecture. Picture adapted from [Li et al., 2020].

JointBERT adds a multi-class training objective to the binary matching objective during the fine-tuning phase. For this objective, the model is tasked with predicting the entity group identifier of each of the two entity descriptions in a pair in addition to the binary entity matching decision.

Figure 7.1 illustrates the architecture and training objectives of JointBERT. The input to the JointBERT model is designed, similar to related work, by first concatenating all attributes of an entity description into a single string representation and then combining two entity descriptions into a pair using [CLS] and [SEP] tokens as follows: *[CLS] Entity 1 [SEP] Entity 2 [SEP]*. The output representation of the [CLS] token of the last encoder layer is then fed into three separate linear layers. The first corresponds to the classifier for the binary decision, i.e. do the entity descriptions refer to the same entity. The activation function for this layer is the sigmoid function, resulting in the probability of the positive class (match). The second and third linear layers are trained to predict the entity identifier of the left and right entity descriptions, respectively. The activation function for both is a softmax layer, resulting in the probabilities for each entity identifier in the training set. Binary cross-entropy loss is used for the binary objective, and cross-entropy loss is used for both multi-class objectives. With $BCEL$ and CEL as binary cross-entropy and cross-entropy loss, respectively, and $y_{b_i}, y_{l_i}, y_{r_i}$ as binary and multi-class labels, the instance loss of JointBERT is defined as:

$$L_i = BCEL\left(y_{b_i}, \hat{y}_{b_i}\right) + \left(CEL\left(y_{l_i}, \hat{y}_{l_i}\right) + CEL\left(y_{r_i}, \hat{y}_{r_i}\right)\right)$$

JointBERT is initialized with the pre-trained BERT_{base} parameters. During training, both objectives are jointly optimized. Matching decisions during inference are solely based on the output of the binary classification layer, making the method applicable on any pair of entity descriptions without the requirement of group identifiers as these are usually not available for all entity descriptions as described in the previous section.

7.4 Method Evaluation

In this section, the performance of JointBERT is compared to the performance of BERT, RoBERTa, Ditto, Deepmatcher, Magellan, and the word co-occurrence baseline using five entity matching benchmark datasets. Two of these benchmarks, *WDC LSPM* and *DI2KG monitors* [Crescenzi et al., 2021], model multi-source entity matching settings and fulfill the requirement from the problem statement in Section 7.3 that multiple entity descriptions should be available for many of the described entities. The other three benchmarks, *Abt-Buy*, *DBLP-Scholar* and *Company* [Mudgal et al., 2018], do not fulfill this requirement and are included in order to evaluate the performance of JointBERT in a two-source setting, i.e., the entity groups in these benchmarks have a maximum size of two unless duplicates within the datasets exist.

7.4.1 Datasets

Tables 7.1 and 7.2 compare the development and test sets used for the experiments with additional information on the size distribution of the entity groups found in the development set.

WDC LSPM

The WDC LSPM benchmark has been introduced in detail in Section 4.6. WDC LSPM models a multi-source entity matching scenario and fulfills the requirements from Section 7.3 that entity identifiers as well as multiple entity descriptions should be available for many entities (see last two columns of Table 7.1). The experiments make use of training, validation and test sets for the four categories *computers*, *cameras*, *shoes* and *watches*. All entities contained in the test sets are also represented with different entity descriptions in the training set, representing a fully seen scenario. In addition to the WDC LSPM default test sets, JointBERT is evaluated on the MWPD test set containing a set of 5 matching challenges as described in Section 4.6.

The attributes *brand*, *title*, *description*, and *specTableContent* are used for all experiments. The three latter attributes are highly textual and contain longer sequences of words. As the attribute values originate from the Web and may contain

Table 7.1: Statistics of the training sets.

Training Set	Size	# Pos. Pairs	# Neg. Pairs	# entities	% entities with min # of descriptions 5	10
WDC computers	xlarge	9,690	58,771	745	57	10
	large	6,146	27,213	745	57	10
	medium	1,762	6,332	745	55	8
	small	722	2,112	745	28	0
WDC cameras	xlarge	7,178	35,099	562	50	14
	large	3,843	16,193	562	50	14
	medium	1,108	4,147	562	47	11
	small	486	1,400	562	22	1
WDC watches	xlarge	9,264	52,305	615	62	15
	large	5,163	21,864	615	62	15
	medium	1,418	4,995	615	59	12
	small	580	1,675	615	30	0
WDC shoes	xlarge	4,141	38,288	562	42	2
	large	3,482	19,507	562	42	0
	medium	1,214	4,591	562	39	0
	small	530	1,533	562	13	0
DI2KG monitor	default	611	68,538	100	59	1
abt-buy	default	822	6,837	992	0	0
dblp-scholar	default	4,277	18,688	2,312	12	1
company	default	22,560	67,569	22,560	0	0

Table 7.2: Statistics of the test sets.

Test Set	# entities w/ pos (overall)	# Pos. Pairs	# Neg. Pairs	# Comb. Pairs
WDC computers	150 (745)	300	800	1,100
WDC cameras	150 (562)	300	800	1,100
WDC watches	150 (615)	300	800	1,100
WDC shoes	150 (562)	300	800	1,100
DI2KG monitor-seen	84 (88)	1,369	122,882	124,251
DI2KG monitor-unseen	123 (141)	7,651	852,365	860,016
DI2KG monitor-combined	207 (229)	9,020	975,247	984,267
abt-buy	205 (819)	206	1,710	1,916
dblp-scholar	848 (1,635)	1,070	4,672	5,742
company	5,640 (5,640)	5,640	16,863	22,503

noise due to extraction errors, the maximum number of words in each attribute is limited. This is done with regard to the length limits of the BERT Transformer model, which is 512 tokens. The attribute value length is limited to twice their median length to ensure that this step only affects unusually long strings.

DI2KG Monitors

All models are further evaluated using the *DI2KG monitors* dataset [Crescenzi et al., 2021] that was used for the DI2KG challenge at the DI2KG workshop[2] at VLDB2020. The dataset contains product offers and corresponding ground truth (entity identifiers) from a wide range of online shops and represents a multi-source matching problem. This dataset fits the requirements of Section 7.3, as entity identifiers are available and multiple entity descriptions exist for most entities (see the last two columns of Table 7.1). The pairwise training set offered for the challenge is used for training. This set was created using a subset of the entity descriptions of a subset of all entities in the ground truth [Crescenzi et al., 2021]. To allow for unbiased evaluation, all entity descriptions contained in the training set are removed from the collection of entity descriptions in the ground truth before building test sets. This means that the same entity description will not appear in the training and test sets. Using the remaining offers, two test sets are built by creating all possible pairs and assigning the corresponding pair as well as entity identifiers to the two sets (1) entities appearing in the training set (seen) and (2) entities not appearing in the training set (unseen). This allows for comparing the performances of classifiers specifically for these two cases. Each offer in this dataset contains a title attribute and a set of specifications that are not aligned across offers originating from different e-shops. The attributes used are the title and the concatenation of all specifications as key-value pairs into a second attribute *specs*. The length of both attributes is restricted in the same way as the length of the attributes in the WDC LSPM benchmark.

Abt-Buy, DBLP-Scholar, Company

All models are evaluated using the *Abt-Buy*, *DBLP-Scholar*, and *Company* entity matching benchmarks. For these experiments, the preprocessed versions and splits[3] that were also used for the Deepmatcher paper [Mudgal et al., 2018] are used. The datasets all model the use-case of two mostly deduplicated datasets to be matched for different domains (see Table 5.1), namely products (Abt-Buy), scientific texts (DBLP-Scholar), and companies. The last two columns of Table 7.1 show that Abt-Buy and Company do not fulfill the second requirement of Section 7.3 that multiple entity descriptions should be available for a larger fraction of the entities. Results on these datasets are included to show how JointBERT performs in such cases. Due to duplicates within the two sources, DBLP-Scholar contains

[2]http://di2kg.inf.uniroma3.it/2020/

[3]https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

at least five entity descriptions for 12% of the entities. The experiments in Section 7.4.3 show that this fraction is too small to meet the second requirement. As the datasets only contain pairwise ground truth labels (match/non-match), the transitive relations resulting from positive pairs are used to assign entity labels to each entity description in all pairs. A summary class *other* is introduced and assigned to all entity descriptions that appear only in negative pairs. All available attributes are used during the experiments for each of the three datasets.

7.4.2 Models and Baselines

As part of the experiments, the performance of *JointBERT* is compared to that of BERT and RoBERTa. Furthermore, Ditto [Li et al., 2020], the state-of-the-art Transformer-based entity matching framework at the time of JointBERTs creation, the previous state-of-the-art framework Deepmatcher [Mudgal et al., 2018], as well as Magellan and the word occurrence-based baseline are evaluated. The following paragraphs present each model and the specific settings used for training them.

Transformer-based models:

All Transformer-based models are implemented using PyTorch [Paszke et al., 2019] and the HuggingFace Transformers library [Wolf et al., 2020]. The uncased base versions of the pre-trained models for BERT and RoBERTa are used for the experiments. For each of them, a linear layer is added on top of the pooled output of the [CLS] token and combined with a sigmoid activation function, resulting in a single output probability for the positive class (match), for the current pair of product offers. All models are trained using binary cross-entropy loss.

All Transformer-based models are trained on a single NVIDIA RTX 2080Ti GPU with 12GB VRAM. The attributes of each entity description are concatenated into a single string. Any further pre-processing is omitted and left to the tokenizer of the respective models. All models are allowed the full input length of 512 tokens. The batch size is fixed at 32, and the Adam [Kingma and Ba, 2015] optimizer is used to train the models for 50 epochs using a linearly decaying learning rate with one epoch warm-up. The learning rate is optimized over the range [1e-5, 3e-5, 5e-5, 8e-5. 1e-4]. Model selection is performed using the maximum F1 value on the validation set. Training is stopped early if the performance of a model on the validation set does not increase over ten consecutive epochs. All models are trained three times and the average performance is reported.

Ditto

For the experiments with the Ditto [Li et al., 2020] framework, the domain knowledge injection module is activated using the spans for the product or general domain, depending on the dataset. The training data augmentation module with the operator *span_del*, which deletes randomly sampled spans to augment the data, is used. For the Abt-Buy, DBLP-Scholar, and Company datasets, the respective operator that was found to work best in the original paper is applied. Instead of using a RoBERTa model at the core of Ditto as in the original paper [Li et al., 2020], the same uncased base BERT version is used to make the results directly comparable to the results of the JointBERT experiments to avoid effects that may occur due to different pre-training of the Transformers. The batch size is set to 8 due to memory constraints of the GPU and the model trained for 50 epochs using a linearly decaying learning rate of 3e-5 with warmup. All Ditto models are trained three times, and the performance average is reported.

Deepmatcher:

For Deepmatcher [Mudgal et al., 2018], the RNN summarization method is selected which has been proven to perform best for the WDC LSPM datasets [Peeters et al., 2020b] (see Section 4.7). The batch size is fixed at 16. The positive-negative ratio, which controls the class weighting, is set to the actual distribution found in each training set. All other hyper-parameters are left at default values. FastText embeddings pre-trained on the English Wikipedia[4] are used as input for Deepmatcher. Each model configuration is trained three times for 50 epochs, and the average is reported. For the datasets Abt-Buy, DBLP-Scholar, and Company, the best result from the original paper is reported [Mudgal et al., 2018].

Magellan and Word Co-occurrence:

The two non-neural baselines are the word co-occurrence model that has been introduced before (see Section 4.7) and the Magellan framework. Both methods are optimized using random search over parameter grids for the following classifiers: Logistic Regression, LinearSVC, DecisionTree, Random Forest, and XGBoost. After finding the best hyperparameter settings and the best classification algorithm on the validation set, the best model configuration is trained three times and the average is reported. For the datasets Abt-Buy, DBLP-Scholar, and Company, the results of the Magellan experiments from the Deepmatcher paper are reported [Mudgal et al., 2018].

7.4.3 Results and Discussion

Table 7.3 shows the F1 results of the experiments across all models and datasets. On the WDC LSPM datasets and the *large* and *xlarge* training sizes, the JointBERT model performs best and can improve on single-objective BERT by 1% to 10% F1 depending on the dataset. The improvement range over RoBERTa and Ditto is smaller with a range of 1% to 5% over Ditto apart from the *watches xlarge* training set, where both perform equally well. Using the multi-class objective in addition to binary classification results in a high matching performance of >95%

[4] https://fasttext.cc/docs/en/pretrained-vectors.html

F1 in all categories for the *large* and *xlarge* WDC training sets. For the *medium* and *small* WDC training sets, JointBERT is outperformed by BERT, RoBERTa and Ditto. The performance loss to single-objective BERT is up to 3% F1 and up to 6% for Ditto for these training sizes. This is due to the overall smaller amount of training data combined with the smaller amount of unique entity descriptions per product, resulting in no longer fulfilling requirement two from Section 7.3, which turns the multi-class training objective to affect the performance negatively instead. All BERT-based models consistently beat the baseline methods and Deepmatcher on all training set sizes and categories for the WDC datasets. The differences are most pronounced for smaller training set sizes, making BERT-based models highly training data efficient compared to Deepmatcher, Magellan, and the word co-occurrence baseline.

Table 7.3: F1 results on the test sets for each dataset.

Testset	Training Set	Word Co-oc	Magellan	Deepmatcher	BERT	RoBERTa	Ditto	JointBERT
LSPM computers	xlarge	82.39	63.16	88.95	94.57	94.73	96.53	**97.49**
	large	81.23	64.56	84.32	92.11	94.68	93.81	**96.90**
	medium	70.94	61.59	69.85	89.31	**91.90**	88.97	88.82
	small	62.69	57.60	61.22	80.46	**86.37**	81.52	77.55
LSPM cameras	xlarge	73.33	51.70	84.88	91.42	94.39	94.74	**98.02**
	large	76.24	54.49	82.16	91.02	93.91	94.41	**96.51**
	medium	69.89	54.99	69.34	87.02	**90.20**	87.97	87.91
	small	64.86	52.78	59.65	77.47	**85.74**	78.67	78.30
LSPM watches	xlarge	79.78	56.04	88.34	95.76	94.87	97.05	**97.09**
	large	79.64	60.59	86.03	95.23	93.93	97.17	**98.46**
	medium	69.54	66.62	67.92	89.00	**92.28**	89.16	87.46
	small	63.49	59.73	54.97	78.73	**87.16**	81.32	75.83
LSPM shoes	xlarge	70.38	61.45	86.74	87.44	88.88	93.28	**97.88**
	large	71.18	60.48	83.17	87.37	86.60	90.07	**95.16**
	medium	72.43	59.80	74.40	79.82	81.12	**83.20**	82.61
	small	63.65	58.57	64.71	74.49	**80.29**	75.13	73.13
DI2KG monitor-seen	default	83.59	21.79	77.41	96.22	96.65	86.51	**97.82**
DI2KG monitor-unseen	default	21.24	16.02	33.93	91.49	**93.26**	73.52	82.86
DI2KG monitor-combined	default	33.82	16.90	37.37	92.19	**93.76**	75.28	84.77
Abt-Buy	default	36.30	43.60	62.80	84.64	**91.05**	82.11	83.44
DBLP-Scholar	default	85.38	92.30	94.70	95.27	**95.29**	94.47	93.99
Company	default	71.81	79.80	**92.70**	91.70	91.81	90.68	91.40

For the DI2KG monitor dataset, RoBERTa performs slightly better than BERT. JointBERT achieves the best overall performance for the *seen* test set is achieved by JointBERT which outperforms RoBERTa by nearly 1.5% F1. On the *unseen* test set, BERT and RoBERTa lose around 3-5% performance while JointBERT drops by 15% F1. As the DI2KG monitor dataset provides enough training data for the multi-class objective (see last two columns of Table 7.1), the impact of this objective on the final prediction is high. This circumstance leads to very good results when predicting for seen entities and has a negative effect when predicting for unseen entities.

For two of the three two-source datasets *Abt-Buy*, *DBLP-Scholar* and *Company*, RoBERTa performs best, while Ditto and BERT perform slightly worse. On Abt-Buy, RoBERTa outperforms BERT by 6% F1. JointBERT results are up to 1%

F1 worse than BERT, due to the limited number of diverse entity descriptions per entity in these datasets.

In conclusion, if training data is limited and JointBERT's second requirement of multiple entity descriptions for many entities is not fulfilled, then using single-objective training and a robust model like RoBERTa, which has been pre-trained on large amounts of textual data, including web pages, leads to higher performance, especially for unseen entities. If more training data for both objectives is available and the task mainly contains seen entities, then JointBERT consistently outperforms RoBERTa. The results presented for JointBERT achieved state-of-the-art results on the WDC LSPM benchmark at the time of this contribution.

7.4.4 Challenge-specific Analysis

As the JointBERT method performs best with many available training examples, in addition to previous experiments, the strong performance of JointBERT is further analyzed when trained using large amounts of training data by evaluating all models, trained with the *computers xlarge* training set on the MWPD test set of the WDC LSPM benchmark.

Table 7.4 provides statistics on the amounts of matches and non-matches for each challenge and reports the performance of all matchers, trained using the *computers-xlarge* training set, for the specific matching challenges of the MWPD test set. For the two challenges that involve unseen products, BERT and RoBERTa perform the best (within 1.5% F1 of each other), with around 84% F1 for unseen products similar to seen products and around 77% F1 for those that are less similar to seen products. JointBERT's performance for unseen products is the worst among all BERT-based models, as the multi-class objective steers the model towards recognizing a specific set of entities and reduces its ability to generalize to unseen entities. This is the same effect that was already observed on the monitor-unseen test set in the previous section.

Table 7.4: F1 Results for the specific matching challenges of the MWPD test set.

Matching Challenge	# Match	# Non-Match	Word Co-oc	Magellan	Deepmatcher	BERT	RoBERTa	Ditto	JointBERT
Unseen - high similarity	25	75	48.00	27.89	58.64	**84.53**	83.18	69.97	59.92
Unseen - low similarity	25	75	11.43	59.04	58.78	76.92	**77.72**	65.78	58.91
Seen - introduced typos	100	0	54.01	19.79	60.76	71.21	85.49	83.49	**89.08**
Seen - dropped tokens	100	0	78.79	50.35	71.75	87.62	89.88	90.28	**93.23**
Seen - very hard cases	25	75	61.22	11.00	74.53	89.04	88.51	94.12	**95.55**
Mix of corner cases	250	750	77.93	58.48	79.90	84.08	**86.32**	85.30	84.24
Full MWPD test set	525	975	69.65	48.23	71.53	82.58	**86.20**	83.96	83.35

When looking at the challenges involving seen products, which have typographical errors in important words or in which words are dropped entirely, JointBERT outperforms all other models by at least 3% F1. The robustness to such challenges thus stems from the additional multi-class objective that supports the binary decision, as the direct similarity comparison of the two strings in a purely binary pair-wise setting becomes harder due to typos or dropped words.

The third challenge for seen products consists of especially hard offer pairs. This challenge set consists of highly similar negatives and dissimilar positives. JointBERT performs best on this challenge and can outperform BERT by 6.5% F1, highlighting the utility of the dual-objective training approach for seen products.

Finally, the last set contains a mix of hard offer pairs for seen products and some unseen products and constitutes the bulk of the MWPD test set. All BERT-based models perform comparably here with around 85% F1 except RoBERTa, which has a slightly higher performance of 86.5% F1. Overall, RoBERTa proves to be the most robust of all models, even though it does not excel over other models at any particular challenge and as a result performs the best overall on the full set.

7.5 Conclusion

This chapter contributed the JointBERT method for dual-objective fine-tuning of Transformers to the field of supervised entity matching. The method is trained for the classic binary matching objective using record pairs (similar to Ditto [Li et al., 2020]) and a second matching objective that leverages available information about entity groups found in the training set, for example, product identifiers for the product matching domain.

The experiments demonstrated that jointly training BERT for binary and multi-class entity matching outperforms state-of-the-art matching models like Ditto that were trained using only the binary objective by 1% to 5% F1 for seen entities given that sufficient amounts of training data for both objectives are available. This joint training method is applicable to any entity matching scenario in which entity identifiers are available for a subset of entity descriptions to be matched. An example of such a scenario are price portals that maintain a product catalog and have to match incoming product offers to it.

The analysis of specific matching challenges in Section 7.4.4 illustrated that JointBERT outperforms the other models on challenges involving seen products by at least 1.5% F1, given that enough training data is available. As a downside, JointBERT underperforms on unseen products in comparison to Ditto, BERT, and RoBERTa, as the multi-class objective biases the model towards recognizing seen entities. JointBERT should be considered for use cases involving seen entities and enough labels for both objectives, while single-objective Transformers are more suited for unseen entities and use cases involving small amounts of training data.

Compared to the work released after the contribution of JointBERT (see Section 7.2), EMBA [Zhang et al., 2024b] is a direct refinement of the dual-objective training approach of JointBERT that, instead of using just the [CLS] token, uses all token representations and an additional attention module to reach a matching decision. The authors were able to further improve the results of the dual-objective method as they directly compare with the original JointBERT in their experiments.

FlexER [Genossar et al., 2023] added looser matching intents to the strict entity matching definition to jointly train for all objectives, resulting in improvements over the Ditto baseline. As the authors restructured the WDC LSPM benchmark and used a different splitting, the JointBERT requirements may no longer be fulfilled in their version of the LSPM benchmark, making the comparison difficult. Furthermore, the FlexER model is not directly applicable to the existing entity matching benchmarks as they do not offer the required multi-intent information.

The *Unicorn* model [Fan et al., 2024a], instead of having multiple objectives, uses multiple matching tasks in their training set to learn a model that projects matching and non-matching pairs of the respective task into a combined embedding space, where pair representations are aligned by a mixture-of-experts and finally passed to a binary classifier. Although their results showed good improvements for many of the tasks considered, the improvements in entity matching are marginal compared to Ditto, mostly < 1% F1. The comparison to JointBERT proves difficult as all their chosen benchmarks do not fulfill JointBERTs requirements to have groups of entity descriptions in the training data. Both methods, FlexER and Unicorn, could be combined with the dual-objective method of JointBERT in future work which may lead to further improvements.

Chapter 8

R-SupCon: Supervised Contrastive Learning for Entity Matching

8.1 Introduction

Contrastive learning [Chopra et al., 2005] is a form of metric learning [Kaya and Bilge, 2019] with the goal of separating dissimilar instances while grouping similar instances in the embedding space. The contrastive learning approach has seen success in the area of information retrieval [Gao et al., 2021] and computer vision [Chen et al., 2020, Khosla et al., 2020], where recent approaches [Khosla et al., 2020] can outperform methods based solely on cross-entropy learning.

This chapter contributes R-SupCon, a method that leverages supervised contrastive learning, to the field of supervised entity matching. The JointBERT method presented in Chapter 7 uses information on entity groups in the training data by predicting entity group identifiers together with the binary match or non-match decision during training. The R-SupCon method presented in this chapter follows the same idea of incorporating entity group information during training, but instead of using the entity group information jointly together with the pair-wise matching during fine-tuning as a dual-objective method, R-SupCon first performs a contrastive pre-training step adapting all layers of the Transformer encoder with a supervised contrastive loss that makes use of entity group information. After this pre-training step, the encoder parameters are frozen, and a single tunable classification layer is added that is fine-tuned with cross-entropy loss for the binary matching objective.

R-SupCon adopts a recent approach for supervised contrastive learning from computer vision called SupCon [Khosla et al., 2020] for entity matching tasks in which the development set contains identifiable groups of entity descriptions, for example, using product identifiers such as GTINs. Similarly to JointBERT, the approach relies on these groups of entity descriptions during training but does not require information about these groups during inference, making it applicable also to any entity description pair without such identifiers.

This chapter further contributes a source-aware sampling strategy for creating training batches for supervised contrastive pre-training, which can be applied to datasets that do not contain explicit identifiers for entity groups but for which pairs of labeled entity descriptions are available. This strategy samples training batches in a way that avoids label-noise inside the training batch, which would harm the learning process.

By combining the RoBERTa transformer encoder, supervised contrastive learning, and the source-aware sampling strategy, R-SupCon achieves new state-of-the-art results on the WDC LSPM benchmark, as well as on Abt-Buy, Amazon-Google, and the *seen* variants of the WDC Products benchmark. The code for replicating the experiments is available on GitHub[1].

The contributions of this chapter are:

- **R-SupCon - Method for Supervised Contrastive Learning for Entity Matching:** R-Supcon, a method for entity matching leveraging supervised contrastive learning based on the SupCon loss proposed in computer vision. It consists of a contrastive pre-training stage followed by cross-entropy fine-tuning of only the final layer. The method can achieve state-of-the-art results on various benchmarks compared to previous state-of-the-art methods based purely on cross-entropy fine-tuning.
- **Source-Aware Sampling Strategy:** A source-aware sampling strategy for training batches that eliminates inter-source label-noise and enables contrastive pre-training also to be successfully applied to use cases without explicitly available identifiers by inferring them from available labeled pairs in the development set.
- **Evaluation of R-SupCon Compared to Previous Methods:** R-SupCon is experimentally compared with recent entity matching methods on a variety of datasets, including the multi-dimensional WDC Products benchmark, showing that the method excels for seen entities and is especially useful in low-resource scenarios compared to existing methods.

Section 8.2 presents related work on contrastive learning and its application to entity matching. Section 8.3 presents the R-SupCon method and its architecture. Section 8.4 presents the experiments on various benchmarks and discusses the results compared to the existing methods. Finally, Section 8.5 summarizes the results and findings of this chapter.

The work presented in this chapter has previously been published in [Peeters and Bizer, 2022b].

[1] https://github.com/wbsg-uni-mannheim/contrastive-product-matching

8.2 Related Work

This section gives an introduction to related work for contrastive learning and its application to data integration tasks, specifically entity matching.

Contrastive Learning

Contrastive learning is a form of metric learning with the goal of clustering similar examples in a vector space while pushing dissimilar examples apart [Jaiswal et al., 2020]. Contrastive learning has a strong foundation in self-supervised learning approaches as it is often used in conjunction with data augmentation to generate positive examples, e.g., in the field of computer vision [Jaiswal et al., 2020]. A well known example of an early loss function for contrastive learning is the triplet loss [Schroff et al., 2015], consisting of triplets of an anchor example together with a positive and a negative example for that anchor. During training, the embedding space over all anchors is learned in a way that the distance between anchor and positive is minimized while the distance between anchor and negative is maximized.

Subsequent work on contrastive losses extended this approach by including more positives or negatives per anchor or both [Chen et al., 2020, Gao et al., 2021, Khosla et al., 2020], often by using matrix multiplication inside training batches to automatically build large amounts of negatives, reducing the need for explicit preparation of anchor/negative/positive pairs as in earlier work [Schroff et al., 2015]. The Sentence-BERT [Reimers and Gurevych, 2019] variants of the BERT transformer are examples of an encoder architecture trained with contrastive losses. Recent work in entity linking also applied self-supervised contrastive losses for learning embedding spaces that cluster similar entities [Wang et al., 2022b].

One of the drawbacks of self-supervised contrastive training is the reliance on augmentation methods to produce interesting positives and the possibility of treating actual positive examples as negatives during batch-based pair induction processes (see results in Section 8.4.3). To avoid the latter, explicit pair-building or special sampling strategies can be applied. If labeled positive examples are available, the former can be replaced by using actual positive examples from the respective domain, which are likely to be more representative of the distribution than augmented examples. Labeled and augmented examples can also be combined for further improved results as shown in Section 8.4.3.

A contrastive loss that incorporates available label information is the supervised contrastive loss SupCon [Khosla et al., 2020] proposed in computer vision. The authors show that this loss in combination with the standard data augmentation techniques from the computer vision domain leads to higher performance in image classification tasks compared to self-supervised and supervised cross-entropy losses. The method presented in this chapter adapts the SupCon loss for entity matching. Another version of a supervised contrastive loss, SCL [Gunel et al., 2020], was proposed in the area of natural language processing.

Contrastive Learning for Entity Matching

In the area of data integration, Brinkmann et al. [Brinkmann et al., 2024] recently proposed SC-Block, a blocking method for entity matching that is based on the adapted SupCon loss presented in this chapter. The authors combine a contrastive training stage using labeled data to learn an embedding space that clusters similar records together with a nearest neighbor search to create candidate pairs to be input to a subsequent entity matching method. They show that the entity matching pipelines run between 1.5 to 4 times faster with supervised contrastive blocking compared to state-of-the-art methods [Brinkmann et al., 2024].

State-of-the-art entity matching models like Ditto [Li et al., 2020] or HierGAT [Yao et al., 2022] are trained with cross-entropy losses, while contrastive learning has only recently become a focus of research interest [Peeters and Bizer, 2022b, Bai et al., 2023, Wang et al., 2023] for deep entity matching methods. To the best of the authors' knowledge, the method presented in this chapter is the first to apply contrastive learning to the entity matching problem. The following paragraphs list relevant work that was published after the method presented in this chapter.

Bai et al. [Bai et al., 2023] apply contrastive learning to learn a general embedding space using multiple entity matching datasets at the same time. Their work is focused on aligning differences in matching patterns across different product matching datasets to one single embedding space that performs well on all datasets and can be further fine-tuned for a specific dataset. Their contrastive loss is a supervised contrastive loss using the labels of positive pairs to generate positives for an anchor in a batch and using the negative batch sampling strategy for negative examples [Chen et al., 2020]. This supervised contrastive loss is highly similar to the adapted SupCon method presented in this chapter. Their results show that their method can improve on Ditto on both in-domain and out-of-domain record pairs after fine-tuning. Being in domain in their terminology means that record pairs from the relevant benchmarks training sets were part of the contrastive training stage.

Wang et al. [Wang et al., 2023] present Sudowoodo which is a method using self-supervised contrastive learning for multiple data integration tasks. They used the SimCLR [Chen et al., 2020] loss in combination with data augmentation to perform contrastive pre-training and subsequently fine-tune using cross-entropy loss using the learned representations for single records during pre-training. The authors specifically investigate a low-resource setting with a budget of 500 available labeled pairs for fine-tuning which is significantly smaller than the available labeled examples in the benchmarks they investigate. The authors further propose a pseudo-labeling strategy to add additional, potentially noisy, labeled pairs for the fine-tuning stage. For this, the authors use cosine similarity in conjunction with the learned embedding representations from contrastive pre-training. Their results show that in this low-resource setting, their method can achieve comparable or better results than the Ditto and Deepmatcher baselines.

8.3 Supervised Contrastive Learning for Entity Matching

The SupCon contrastive loss presented in [Khosla et al., 2020] uses all examples in a batch to maximize the distances between an example and all of its negatives in that batch, as well as minimize the distances between an example and all of its positives. To achieve this, the method exploits label information about the training examples in combination with augmenting single examples to derive additional positive examples for learning. All examples in a batch that do not carry the same label are treated as negatives. The SupCon loss can also be applied in a self-supervised fashion if label information is not available by only generating positives for each example using data augmentation (this corresponds to the SimCLR loss [Chen et al., 2020]). The original implementation of SupCon for computer vision uses a set of *N* randomly sampled example/label pairs (e.g., a picture of a dog and the label "dog") where each example is augmented twice using a random function from a set of augmentation functions, creating final input batches of *2N* examples. Using the assigned labels, the contrastive loss among all examples of a batch can be calculated and iteratively optimized using gradient descent optimization. In comparison to SimCLR, the SupCon loss can also handle different examples from the same class that did not originate from augmentation of the same example. The experiments in Section 8.4.3 show that this property is essential for the application of contrastive learning to the entity matching problem.

The SupCon loss is defined by [Khosla et al., 2020] as:

$$L = \sum_{i \in I} L_i = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)} \tag{8.1}$$

where $i \in I \equiv 1...2N$ is the index of the anchor. $\tau \in \mathbb{R}^+$ is the scalar temperature parameter and $A(i) \equiv I \backslash i$. Further, $P(i) \equiv \{p \in A(i) : \tilde{y}_p = \tilde{y}_i\}$ is the set of indices of all positives in the batch distinct from i.

The R-SupCon method for applying contrastive learning to entity matching consists of two steps: (1) a contrastive pre-training step on batches of individual entity descriptions using SupCon loss, followed by (2) a fine-tuning step using matching and non-matching pairs of entity descriptions. The RoBERTa-base model is used as the encoder architecture, which has been shown to achieve strong results across different entity matching benchmark datasets and different training set sizes in previous chapters.

8.3.1 Stage 1: Contrastive Pre-Training

Labels for Contrastive Pre-Training

Supervised contrastive training assumes that all examples that refer to the same entity share the same label, e.g. all pictures of dogs are labeled as "dog". The development sets of some entity matching tasks include identifiers for entity groups, such as GTIN or MPN numbers for products, which can directly be used as labels for contrastive training. The training sets of other tasks do not provide explicit group identifiers but only label a certain amount of entity description pairs from different sources as matching or non-matching. For contrastive pre-training, explicit labels on the entity level need to be obtained, so that matching entity descriptions share the same label. To obtain such labels, R-SupCon uses the matching pairs from the development set and builds a correspondence graph over all entity descriptions, where the graph's edges connect matching entity descriptions. A unique label can then be assigned to each connected graph component so that matching entity descriptions share the same label.

Source-aware Sampling Strategy

As often only a subset of matches between sources is known and labeled in the development set, it is likely that actually matching entity descriptions will be assigned different labels during the process presented above. During the contrastive pre-training stage, this will result in treating these as non-matching entity descriptions if they appear in the same batch due to the different labels. This circumstance severely deteriorates the quality of the learned representations as discussed in Section 8.4.3. To alleviate this problem, a source-aware sampling strategy is applied that allows the elimination of such inter-source label noise. Instead of generating one combined dataset containing all entity descriptions and their labels from each source, one dataset is generated per source, containing all entity descriptions from that source as well as only those entity descriptions from other sources which share a label with an entity descriptions from the current source, i.e. entity descriptions that were originally labeled as a match. Figure 8.1 illustrates this procedure for a matching task involving three sources.

Once a sampling dataset is built for every source using this procedure, entity descriptions are sampled from only one of the sampling datasets into each batch. For each batch, the dataset from which to sample is chosen at random. This procedure eliminates inter-source label noise during contrastive training, given that the data sources themselves do not contain duplicates. If it is known beforehand that inter-source label noise does not exist, e.g., when the selected entity descriptions for contrastive training are already annotated with group identifiers, a single sampling set containing all entities is sufficient. Note that this sampling strategy assumes that the sources themselves are either deduplicated or duplicates are known. Otherwise, a certain amount of label-noise may still exist inside each sampling dataset.

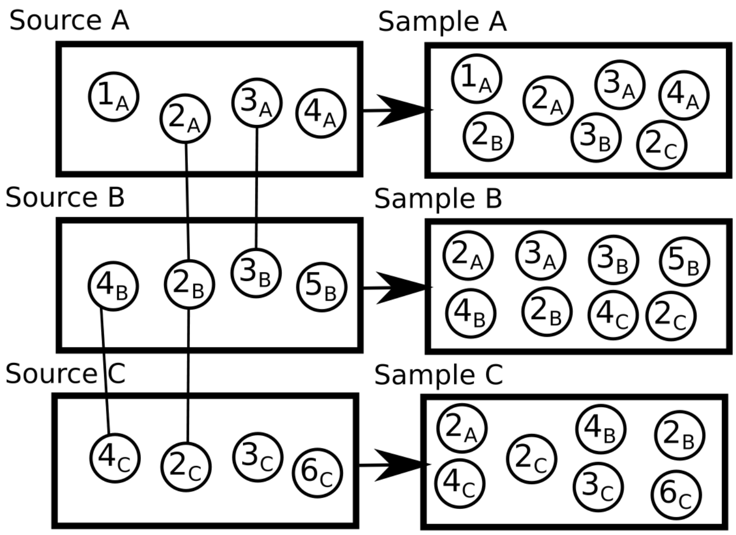


Figure 8.1: Sampling strategy for datasets with limited product identifier information. Produces one sampling dataset per source containing all product offers of that source as well as all known (connected) matching offers from other sources.

Batch Building Process

The method of the original SupCon paper [Khosla et al., 2020] for assembling each batch is applied with modifications. First, *N* entity descriptions from a random one of the sampling datasets are selected and the available group identifier information is used to randomly select for each of these entity descriptions a matching entity description from the dataset. The selection of the same entity description is explicitly allowed, even if other entity descriptions with the same label exist. The final batch then consists of *2N* offers where for each of the *N* offers at least one offer having the same label is contained in the batch. In contrast to the original SupCon paper, an additional projection head is not used on top of the encoder, but the mean-pooled representation of the last encoder layer is directly taken and normalized using L2 normalization to attain a vector representation for a single entity description for contrastive training.

All entity descriptions in a batch are then propagated through the encoder network to produce their vector representations, which are subsequently used to calculate the SupCon loss and tune the encoder parameters minimizing/maximizing distances between all matching/non-matching entity descriptions in the batch. As batches are sampled differently across epochs, many distance comparisons across all entity descriptions are performed over all training epochs, leading to good representations in the learned vector space.

Attribute Value Serialization

Entity descriptions are serialized by concatenating all their attributes while maintaining attribute separation by inserting additional tokens. More specifically, a single attribute is serialized as "[COL] column_name [VAL] actual_value". These strings are then concatenated for all attributes of an entity description to build its serialized input representation.

Data Augmentation

Applying data augmentations, such as deleting words, to entity descriptions can easily distort entity descriptions to the extent that the assigned label is no longer correct. For example, dropping the "4s" from the string "Apple iPhone 4s" would make it impossible to assign the correct label with certainty. Even without such explicit data augmentation in the symbolic space, the usage of dropout noise during training inherent to Transformer encoders can be regarded as soft data augmentation in the embedding space since two embeddings of the same product offer will nearly never look exactly the same during training. As a result of this, even the selection of the same entity description as a positive to itself can still have a positive effect on training the Transformer.

In addition to using the default dropout noise, the following evaluation experiments with applying explicit data augmentations to entity descriptions during the pre-training stage. For this purpose, the nlpaug[2] python package is used and 6 types of augmentations are selected: (1) simulating typos, (2) swapping words, (3) deleting words, (4) deleting spans of words, (5) substituting words with synonyms, and (6) randomly splitting words. For every selected entity description in a batch, a random choice is made among all augmentations and the option of not augmenting at all. If an entity description is selected for augmentation, every word in that entity description has a 10% chance to be augmented with the currently selected augmentation method.

8.3.2 Stage 2: Cross-Entropy Fine-Tuning

For the fine-tuning step, a single dropout and linear layer are added on top of the model, which returns a binary label, match, or non-match, for a pair of entity descriptions. Both entity descriptions in a pair are propagated through the encoder, and their mean-pooled representations are combined as input to the final classification layer as follows: Given the two embedding representations u and v, they are combined as: $(u, v, |u - v|, u * v)$. The model is trained using binary cross-entropy loss. The parameters of the encoder layers can either be frozen or further tuned during the fine-tuning step, while the parameters of the classification layer are always tuned.

[2]https://github.com/makcedward/nlpaug

8.4 Method Evaluation

In the first set of experiments, R-SupCon is evaluated on three benchmark datasets from the product matching domain: Abt-Buy, Amazon-Google and the WDC LSPM computers category. Abt-Buy and Amazon-Google represent the use case of matching product offers from two deduplicated sources. The offers in both datasets do not contain product identifiers. The WDC LSPM dataset, on the other hand, represents a multi-source matching task. The WDC LSPM training and validation sets feature product identifiers for all offers.

In the second set of experiments, R-SupCon is applied to the various variants of the WDC Products benchmark to analyze the models performance on the three dimension of the benchmark. R-SupCon is compared with the following baselines on WDC Products: the word (co-)occurrence baseline, Magellan [Konda et al., 2016], Ditto [Li et al., 2020], RoBERTa-base [Liu et al., 2019b] and HierGAT [Yao et al., 2022]. In addition to the pair-wise variant of the benchmark, R-SupCon is also applied to the multi-class version and compared to the relevant baselines.

The R-SupCon model is implemented using the Huggingface *transformers* library[3]. For contrastive pre-training, the batch size is set to 1024. The temperature parameter is left at its default value of 0.07 and the model is trained using the Adam optimizer for 200 epochs with a linearly decaying learning rate of 5e-05 with a 0.05 warm-up ratio. For the fine-tuning step, the batch size is set to 64 and the model trained for up to 50 epochs using early stopping if validation loss does not improve for 10 consecutive epochs. Each model is trained three times and the average results are reported.

8.4.1 Datasets

Abt-Buy/Amazon-Google

All available attributes are used for training and evaluation for the two-source datasets Abt-Buy and Amazon-Google. Both benchmarks do not contain product identifiers for individual product offers. Only labeled positive and negative offer pairs are available. The method described in Section 8.3.1 is applied together with the source-aware sampling strategy to generate identifiers for contrastive pre-training. The selection of pairs to use for building the correspondence graph is done only on the training and validation splits of the pairwise datasets. In an effort to introduce a regularization effect, only 80% of matching pairs from the training and validation splits are used to perform this calculation. Due to the low number of training offers per product in the two-source matching case, the model is more prone to overfit to the few known matching offers, reducing performance for products where no matching offers have been seen during contrastive pre-training. By withholding known matching information for 20% of offer pairs, the model can later use these during the fine-tuning stage to better adapt to such cases.

[3] https://github.com/huggingface/transformers

WDC LSPM

The training, validation, and test sets from the *computers* category of WDC LSPM, as introduced in Section 4.6, are used in these experiments. In addition to the pair labels, the product offers are further annotated with product identifiers, which identify offers for the same products from different sources. These were previously used to build the benchmark from Schema.org annotations in the Common Crawl as described in Chapter 4. The source-aware sampling strategy is not necessary in this case, as the product identifiers can be directly used as labels for contrastive pre-training and the pair-wise labels for the fine-tuning step. All product offers that are part of the training and validation sets are used for contrastive pre-training. The attributes used are *title*, *description*, *brand*, and *specTableContent*.

WDC Products

The WDC Products benchmark is used in its entirety, allowing for the evaluation along its dimensions as introduced in Section 5.3. This benchmark is annotated with product identifiers which also originate from Schema.org annotation in the Common Crawl. Source-aware sampling is not necessary for this benchmark. All product offers that are part of the training and validation sets are used for contrastive pre-training. The full range of available attributes, *title*, *description*, *brand*, *price* and *priceCurrency*, is selected for the experiments.

8.4.2 Models and Baselines

Word (Co-)Occurrence

The Word co-occurrence model represents the same simple symbolic baseline for pair-wise matching used in the previous chapters of this thesis. It uses the binary word co-occurrence between two entity descriptions in a pair as feature input to a binary LinearSVM sklearn [Pedregosa et al., 2011] classifier. For the multi-class matching case in WDC Products, the method is slightly adapted, as the feature input is no longer a representation of a pair of offers but a single product offer to classify. A binary word occurrence vector instead of word co-occurrence is used as a result. During training, a grid search is performed on a set of combinations of parameters.

Magellan

Magellan [Konda et al., 2016] is the second symbolic baseline. Magellan selects appropriate similarity metrics for each attribute depending on the attribute datatype. The resulting attribute-specific similarity scores are inputted to a sklearn binary Random Forest classifier, which determines the matching decision. Similarly to the previous baseline, the model parameters are optimized using a grid search.

Deepmatcher

For the Deepmatcher models the RNN variant of the models is used, which has been proven to work well for the task as discussed in Section 4.7. The parameter settings correspond to those presented in that section.

RoBERTa

The RoBERTa-base [Liu et al., 2019b] model is a sub-symbolic baseline representing a Transformer-based language model which has been shown to reach high performance on the entity matching task (see Section 7.4 and [Li et al., 2020, Peeters and Bizer, 2021]). The RoBERTa model is fine-tuned for pair-wise and multi-class matching for 50 epochs, with early stopping after 10 epochs if the validation score does not improve. The batch size is set to 64, and the learning rate linearly decreases with warm-up, with a maximum value of 5e-5.

Ditto

For Ditto [Li et al., 2020] the experiments use a RoBERTa-base language model with the activated data augmentation module, specifically the *delete* operator. The Ditto model is fine-tuned for 50 epochs with a batch size of 64 and a learning rate of 5e-5 on a linearly decreasing schedule with warm-up.

JointBERT

The JointBERT model as introduced in Chapter 7. The parameter settings for each dataset correspond to those presented in that chapter.

HierGAT

The HierGAT [Yao et al., 2022] model combines the attention mechanism of the Transformer language model and a hierarchical graph attention network. The HierGAT model is trained for 50 epochs with a batch size of 16 and a linearly decreasing learning rate of 5e-6 with warmup.

8.4.3 Results and Discussion

Abt-Buy, Amazon-Google, WDC LSPM Results

The first set of experiments compares R-SupCon to the neural entity matching systems Ditto [Li et al., 2020], JointBERT [Peeters and Bizer, 2021], Deepmatcher [Mudgal et al., 2018] and RoBERTa [Liu et al., 2019b]. Two versions of contrastively pre-trained RoBERTa models are evaluated: (1) R-SupCon using supervised SupCon loss, and (2) R-SimCLR using self-supervision only, corresponding to SimCLR loss [Chen et al., 2020]. For R-SimCLR, each product offer is assigned a unique identifier, and a match for each offer is only sampled by augmenting the

Table 8.1: F1-score results on the test sets for each dataset and training size. (F) and (UF) signify frozen and unfrozen encoder parameters during fine-tuning. For Abt-Buy and Amazon-Google results in brackets signify not reducing label-noise by separately sampling from both data sources. Results with * are taken from [Li et al., 2020].

	Abt-Buy	Amazon-Google	LSPM Computers			
# Training Pairs	~7.5K	~9K	~3K (small)	~8K (medium)	~23K (large)	~68K (xlarge)
Deepmatcher	62.80*	70.70*	61.22	69.85	84.32	88.95
RoBERTa	91.05	74.10*	86.37	91.90	94.68	94.73
Ditto	89.33*	75.58*	80.76*	88.62*	91.70*	95.45*
JointBERT	-	-	77.55	88.82	96.90	97.49
R-SupCon(F)	93.70(38.24)	**79.28**(42.44)	93.18	97.66	98.16	**98.33**
R-SupCon(F)+aug	**94.29**	76.14	**95.21**	**98.50**	**98.50**	98.33
R-SupCon(UF)	79.99(71.47)	71.81(61.06)	79.52	87.32	94.59	96.16
R-SupCon(UF)+aug	77.84	68.37	80.69	89.12	94.56	96.13
R-SimCLR(F)	56.63	56.16	53.98	55.25	58.97	60.66
R-SimCLR(F)+aug	53.67	54.29	53.36	54.97	58.34	62.19
R-SimCLR(UF)	79.99	64.87	65.75	82.72	92.20	95.25
R-SimCLR(UF)+aug	79.28	63.71	66.73	82.24	91.89	95.75
Δ to best baseline	+3.24	+3.7	+8.84	+6.6	+1.6	+0.84

same offer either via implicit dropout noise or explicit data augmentation. Table 8.1 shows the results of applying contrastive pre-training compared to the four baseline systems.

For the two-source datasets Abt-Buy and Amazon-Google, applying contrastive pre-training results in an improvement of 3.2-3.7% F1 compared to the respective strongest baseline model. Results are reported for two sampling strategies for contrastive learning, one containing all offers from all sources in the sampling set leading to label-noise for these dataset (see Section 8.4.1), and the other using the source-aware sampling strategy with separated sampling datasets to eliminate label-noise. The experiments show that it is important to apply the source-aware sampling for supervised contrastive learning for such cases: For Abt-Buy and Amazon-Google, performance drops by 55% and 37% F1, respectively, without the source-aware sampling strategy. All contrastively pre-trained RoBERTa models outperform the baselines by 0.8-8.8% F1 for the WDC LSPM dataset. Adding contrastive pre-training can improve the best baseline results for small and medium training sizes by 8.8% and 6.6% F1, respectively. Improvements on large and xlarge are visible but comparably small in the range of 0.8% to 1.8% F1. Freezing the encoder parameters after the contrastive pre-training step leads to higher performance on all datasets compared to further updating them during the fine-tuning step.

Applying augmentation during the contrastive pre-training phase delivers mixed results across datasets. For the smaller WDC training sets, there is an improvement of 1-2% F1 but only minimal improvements on the larger training sets and on Abt-Buy, while Amazon-Google sees a 4% drop in performance when augmentation is applied.

Using only self-supervision during contrastive pre-training leads to worse results compared to not pre-training at all. The label-noise inherent to self-supervised SimCLR due to only regarding an augmented version of the same offer as matching and treating all others as non-matching, even if they do actually match, is likely the cause of the large drop in performance.

In summary, supervised contrastive pre-training can further improve the performance of matching systems compared to direct cross-entropy fine-tuning, with the caveat that label-noise must be minimized during supervised contrastive pre-training as otherwise the performance of the matchers is negatively impacted as entity descriptions belonging to the same real-world entity will be pushed apart. Using SupCon loss and an appropriate sampling strategy for the two-source datasets to reduce label-noise can improve on all benchmark datasets by 1-9% F1, setting a new state-of-the-art for all of them at the time of contribution.

Compared to JointBERT, which had the requirement of multiple available entity descriptions for each entity group to perform well, the R-SupCon method is also feasible on smaller training set sizes and when only a small amount of entity descriptions are available in each entity group. The next sections further compare the strengths and weaknesses of R-SupCon to baseline methods along the dimensions of the WDC Products benchmark.

WDC Products Pairwise Benchmark Results

Table 8.2 summarizes the F1-score results of the experiments for the pair-wise matching tasks along all three dimensions of the WDC Products benchmark: amount of corner cases, amount of unseen entities in the test set, and development set size. See Section 5.3 for a detailed description of the dimensions and the benchmark. Table 8.3 shows the corresponding precision and recall values of the four neural matching systems. Figures 8.2, 8.3, and 8.4 visualize the results along the three dimensions.

Dimension Corner Cases: Figure 8.2 fixes the development set size to *medium* and unseen products to 0% while varying the amount of corner cases. Increasing the amount of corner cases causes all methods to lose performance while not changing the ranking of the methods. The overall drop in performance is similar for all models, suggesting that no model is much better suited for handling corner cases. The absolute F1 values for the variants of the benchmark containing 80% corner cases for the top-performing systems R-SupCon, Ditto, and RoBERTa all lie between 72.18 and 79.99. Varying the amount of corner cases negatively impacts the precision of matchers more than it does recall. The matching systems mistake very similar negatives as positives which is in line with the increased amount of these cases along this dimension.

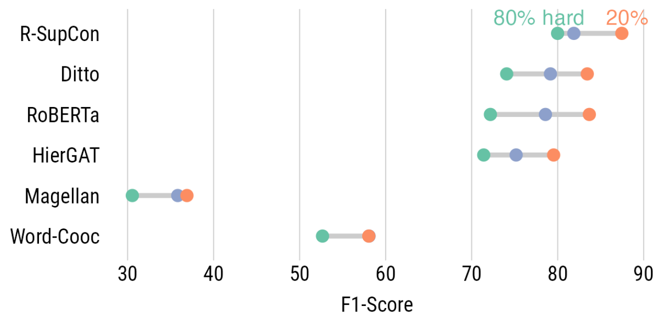


Figure 8.2: Varying corner-case ratio combined with no unseen entities in the test set and development set size medium.

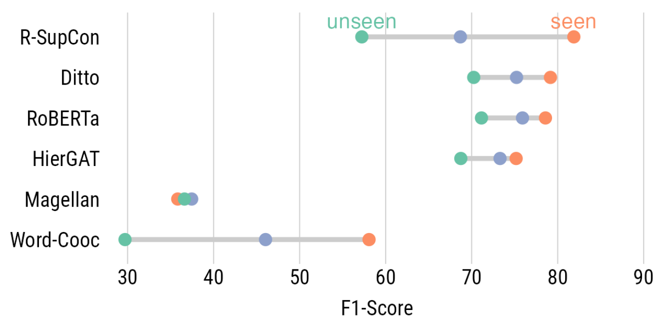


Figure 8.3: Varying fraction of unseen products in the test set combined with 50% corner-cases and development set size medium.

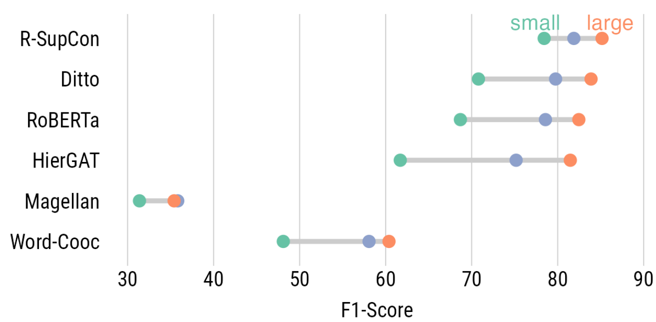


Figure 8.4: Varying development set sizes combined with 50% corner-cases and no unseen entities in the test set.

Table 8.2: Results of the pair-wise experiments over all three dimensions, amount of corner-cases, amount of unseen products in the test set, and development set size. Results are F1 scores for the class match. Bold results indicate the best F1 score for that benchmark variant, underlined indicates second best.

Development Set Size	Corner Cases	Word-Cooc			Magellan			RoBERTa			Ditto			HierGAT			R-SupCon		
		Seen	Half-Seen	Unseen	Seen	Half-Seen	Unseen	Seen	Half-Seen	Unseen	Seen	Half-Seen	Unseen	Seen	Half-Seen	Unseen	Seen	Half-Seen	Unseen
Small	80%	43.73	40.07	27.46	31.15	33.75	33.34	65.45	**66.68**	**64.50**	58.33	58.97	57.16	59.65	61.54	60.63	**77.48**	64.25	51.91
Medium		52.66	44.06	30.57	30.55	35.00	33.47	72.18	72.05	**70.13**	74.07	**72.78**	69.49	71.40	67.64	67.45	**79.99**	67.21	53.10
Large		56.67	50.24	30.26	31.96	36.42	34.95	78.15	**75.52**	**69.75**	79.46	68.81	67.94	75.42	73.20	68.53	**82.15**	67.27	53.31
Small	50%	48.10	40.23	29.44	31.38	32.44	33.34	68.69	**69.18**	**65.79**	70.19	65.40	61.84	61.70	60.74	59.21	**78.43**	68.24	57.44
Medium		58.07	46.04	29.70	35.83	37.45	36.61	78.58	**75.91**	**71.14**	79.16	75.22	70.24	75.17	73.30	68.74	**81.88**	68.69	57.23
Large		60.39	51.15	31.64	35.41	37.39	38.51	82.46	78.89	**71.52**	83.88	**79.36**	69.36	81.47	76.98	71.34	**85.16**	71.15	57.68
Small	20%	46.55	45.30	33.30	34.17	37.50	35.18	75.24	**75.87**	72.44	73.96	75.36	**72.62**	64.34	64.62	68.25	**85.06**	73.09	64.56
Medium		58.04	51.33	34.38	36.90	40.68	37.10	83.68	**80.60**	**78.35**	83.43	78.40	76.33	79.53	77.60	74.84	**87.46**	73.17	63.52
Large		61.81	54.26	35.83	37.58	41.57	37.23	87.80	82.17	**78.64**	87.52	**82.81**	77.92	84.15	79.54	75.53	**89.04**	74.59	62.45

Table 8.3: Precision and recall values for the neural matching systems.

Development Set Size	Corner Cases	RoBERTa						Ditto						HierGAT						R-SupCon					
		Seen		Half-Seen		Unseen		Seen		Half-Seen		Unseen		Seen		Half-Seen		Unseen		Seen		Half-Seen		Unseen	
		P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
Small	80%	55.84	79.07	57.06	80.27	54.20	79.80	46.28	79.33	46.28	83.13	43.93	83.60	49.20	76.27	51.06	77.53	48.92	80.13	71.19	85.00	55.87	75.60	40.14	73.47
Medium		66.61	79.07	65.60	80.47	61.92	81.40	70.97	77.53	67.10	79.53	60.76	81.47	68.68	74.40	61.08	75.93	59.82	77.47	76.18	84.20	60.08	76.27	41.73	73.00
Large		73.82	83.07	69.73	82.53	61.93	80.27	77.09	81.33	63.81	74.80	59.67	78.93	71.42	80.00	67.27	80.47	60.96	78.60	79.09	85.47	59.89	76.73	42.32	72.00
Small	50%	60.35	79.93	58.77	84.33	56.00	80.00	62.43	80.20	54.06	83.20	50.15	82.20	51.83	76.67	47.39	84.60	45.83	83.73	73.56	84.00	57.00	85.00	45.83	76.93
Medium		76.02	81.33	67.92	86.07	62.61	82.40	75.99	82.87	65.28	88.87	60.74	83.33	74.72	75.80	65.32	83.73	59.10	82.40	79.04	84.93	57.58	85.13	45.70	76.53
Large		83.20	81.80	71.89	87.53	63.26	82.40	84.29	83.53	72.21	88.13	59.25	83.67	80.49	87.47	68.60	87.73	62.01	84.00	84.08	86.27	59.81	87.80	47.07	74.47
Small	20%	71.72	79.47	69.60	83.53	63.66	84.13	67.13	82.87	68.02	84.80	63.55	85.13	56.10	76.60	54.95	79.33	58.86	81.33	83.09	87.13	63.60	85.93	53.59	81.20
Medium		83.43	83.93	75.12	87.00	70.63	88.00	83.10	83.80	71.70	86.53	68.54	86.13	77.26	81.93	71.66	84.67	66.95	84.87	87.05	87.87	63.57	86.20	52.96	79.33
Large		87.81	87.80	75.53	90.13	70.51	88.93	89.93	85.27	78.59	87.53	72.34	84.47	81.17	87.40	71.35	89.93	66.49	87.47	88.69	89.40	64.80	87.87	51.13	80.20

Dimension Seen/Unseen Figure 8.3 shows the performance of all methods in test sets that contain only seen, a mix of unseen and seen, as well as only unseen products while fixing the corner case dimension to 50% and development set size to *medium*. All methods show a significant drop in performance compared to completely seen data, but specifically R-SupCon, which performs best on seen data, experiences a large drop of 25%. However, all deep learning-based approaches outperform symbolic baselines even for unseen products. These same findings are visible on the remaining corner case and development size sets. The unseen dimension negatively impacts the precision of all matchers significantly while recall increases, only R-SupCon experiences drops in performance on both measures, explaining the observed large drop in F1. The contrastive pre-training thus severely limits the generalization ability of the method as the created clusters in the vector space are not helpful for correctly disambiguating unseen products.

Dimension Development Set Size Figure 8.4 shows the results for the development set size dimension while fixing the corner cases to 50% and the unseen products to 0%. The results show that most methods struggle with the *small* development set apart from R-SupCon which can achieve an F1 of over 78%. This gap closes significantly with the *medium* development set for deep learning methods. Deep learning-based methods outperform symbolic baselines for pair-wise matching, even with the smallest development sets. Regarding the precision and recall measures, an increased size of the development set translates into improvements for both measures on all systems, although the increases in precision are more pronounced, suggesting that the additional development data helps the models to predict negative corner cases better.

WDC Products Multi-class Benchmark Results

Table 8.4 shows the results of the multi-class matching experiment. R-SupCon performs significantly better than the other two baselines for multi-class matching. The simple symbolic word occurrence baseline is able to beat the fine-tuned RoBERTa model for small- and medium-size datasets for multi-class matching, an effect that was not observed for the pair-wise variant. The comparison of the multi-class RoBERTa model to its pairwise counterpart shows that both methods achieve similar results given the large development size, but the multi-class model is not able to achieve the same results for the smaller sizes, showing that for a multi-class formulation, fine-tuning a transformer model requires a minimum amount of three to four offers per class to achieve good results. Finally, R-SupCon reaches 3-6% higher total F1s than its pair-wise counterpart even for smaller development set sizes, suggesting that this method is especially well suited for a multi-class matching scenario with the goal of recognizing a set of known products. The multi-class experiments further underline the usefulness of the two versions of the WDC Products benchmark for pair-wise and multi-class matching, as the performance of tested methods can be different depending on the entity matching formulation, as the experiments show.

Table 8.4: Results for multi-class matching over the dimensions development set size and amount of corner-cases. Results are micro-F1.

Development Set Size	Corner-Cases	Word-Occ	RoBERTa	R-SupCon
Small		63.30	36.63	82.30
Medium	80%	71.50	52.03	88.63
Large		79.40	78.77	89.33
Small		68.60	40.83	85.23
Medium	50%	76.10	61.33	89.80
Large		81.10	82.00	91.73
Small		66.60	39.83	87.87
Medium	20%	76.20	61.13	92.60
Large		81.30	83.37	93.03

8.5 Conclusion

This chapter contributed R-SupCon, a method for supervised contrastive learning, to the field of supervised entity matching. The method uses a supervised contrastive pre-training stage which does not rely on record pairs directly but instead uses entity group information in the training set, similar to JointBERT, to train on record level, subsequently automatically generating a large amount of record pairings in the training batches.

The experiments demonstrated that supervised contrastive pre-training followed by cross-entropy fine-tuning can improve the performance of matchers compared to only performing cross-entropy fine-tuning on multi-source and two-source benchmark tasks by 0.8-8.8% in F1-score, thereby setting a new state-of-the-art for the Abt-Buy, Amazon-Google and WDC LSPM computers benchmarks at the time of contribution. The evaluation using multiple training set sizes for the multi-source benchmark WDC LSPM computers shows that the matching performance improvement of R-SupCon is most pronounced for smaller training sets.

The proposed source-aware sampling strategy designed to reduce inter-source label noise during contrastive pre-training is crucial for achieving good performance on matching tasks without explicit entity group information in the training set, e.g. available product identifiers. Label-noise is inherently harmful for the matching task, as shown by using different sampling strategies for the two-source datasets. When label-noise exists, the contrastive training stage pushes noise-affected matching records apart or non-matching records closely together in the embedding space, resulting in erroneous matching decisions during inference. The application of data augmentation to the contrastive pre-training step showed that this further increased performance for all but one of the used datasets, especially for smaller training set sizes.

The comparison of R-SupCon on the dimensions of the WDC Products benchmark to methods purely trained using cross-entropy loss like Ditto [Li et al., 2020] and HierGAT [Yao et al., 2022] shows that R-SupCon achieves state-of-the-art performance on the amount of corner case and development set size dimensions when considering the fully seen scenario. R-SupCon is not suitable for scenarios involving unseen entities, as the matching performance of a RoBERTa model fine-tuned with cross-entropy loss leads to better generalization in these scenarios.

Compared to the JointBERT method introduced in Chapter 7, both methods make use of entity group information in the training set. JointBERT includes this information as a second objective during fine-tuning while R-SupCon has two separate stages, using the entity group information during the pre-training stage and subsequently fine-tuning only the last layer with cross-entropy loss. The comparison of both systems in this chapter and the results from Section 7.4.4 show that they face the same challenges in the unseen scenario, as the focus on a specific set of entity groups reduces generalization performance to unseen entities compared to pair-wise cross-entropy fine-tuning. While JointBERT requires multiple diverse entity descriptions for each entity group to reach a good performance, R-SupCon achieves a high performance despite only few available entity descriptions per entity group, achieving higher performance than Ditto and HierGAT for the seen scenario. This circumstance is likely due to the explicit learning of clusters of entity descriptions referring to the same entity in the vector space while JointBERT learns to incorporate information from the varied surface forms in each entity group. The latter potentially requires more examples than explicit metric learning to achieve the same level of performance. Testing this hypothesis is left to future work.

Compared to the DAEM system [Bai et al., 2023] by investigating the results on the benchmarks on which both systems were evaluated, R-SupCon achieves ∼5-10% higher F1 scores on the WDC LSPM computers benchmarks, as well as ∼4.5% higher F1 on the Abt-Buy benchmark. The idea behind DAEM is to create a general matching model during supervised contrastive pre-training using a variety of benchmarks and subsequently fine-tuning the model to a specific benchmark domain. Various differences exist between both systems that may cause the difference in results, for example, DAEM does not freeze the encoder weights after the pre-training stage. The system further does not include Abt-Buy as part of the pre-training and finally the domain adaptation architecture and adversarial training of the system, together with no source-aware sampling may further impact the resulting performance causing the overall lower performance compared to R-SupCon.

The Sudowoodoo [Wang et al., 2023] system applies unsupervised contrastive loss and investigates a self-supervised low-resource setting resulting in 13% and 20% lower performance on Abt-Buy and Amazon-Google, respectively, compared to R-SupCon. As Sudowoodoo does not use source-aware sampling nor entity group information during contrastive training, the method does not achieve the same performance as R-SupCon and a direct comparison of results is not possible as both methods aim at different use-cases.

Chapter 9

Prompt Engineering and Fine-tuning of Large Language Models for Entity Matching

9.1 Introduction

The major drawbacks of recent PLM-based methods like Ditto [Li et al., 2020], JointBERT, R-SupCon, and HierGAT [Yao et al., 2022] for entity matching are that PLMs need a lot of task-specific training examples for fine-tuning and that they are not very robust concerning unseen entities that were not part of the training data (see Sections 7.4.4 and 8.4.3 and [Akbarian Rastaghi et al., 2022]).

Generative large language models [Zhao et al., 2023] such as GPT, Llama, Gemini, and Mistral have the potential to address both of these shortcomings. Due to being pre-trained on large amounts of textual data as well as due to emergent effects resulting from the model size [Wei et al., 2022], LLMs often show a better zero-shot performance compared to PLMs and are more robust concerning unseen examples [Brown et al., 2020, Zhao et al., 2023].

This chapter contributes a comparison of the effectiveness of different prompting techniques and fine-tuning of LLMs to the field of entity matching. The results show that LLMs are a less task-specific training data-dependent and more robust alternative to PLM-based matchers. The models are evaluated in a zero-shot scenario and a scenario where task-specific training data is available and can be used to select demonstrations, generate matching rules, or fine-tune the LLM. The work presented in this thesis covers hosted and open source LLMs that can be run locally. At the time of this contribution, the usage of LLMs had only been investigated in an exploratory study [Narayan et al., 2022]. Concurrent and follow-up work since that then is discussed in Section 9.2. Figure 9.1 shows an example of how LLMs are used for entity matching. The two entity descriptions at the bottom of the figure are combined with the question of whether they refer to the same real-world entity in a prompt. The prompt is passed to the LLM, which generates the answer shown

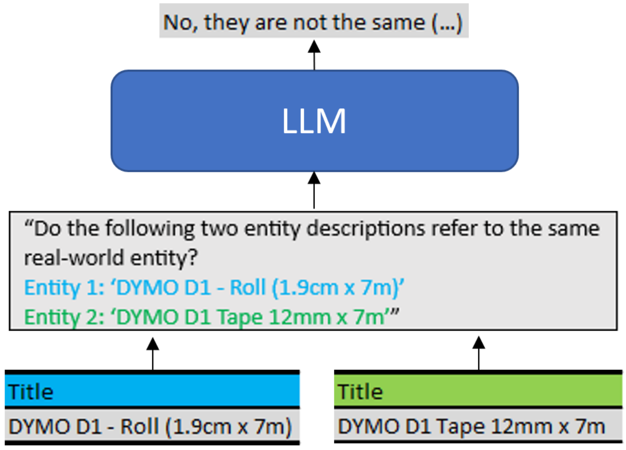


Figure 9.1: Process of combining two entity descriptions into a prompt which is afterward passed to the LLM.

at the top of Figure 9.1. All data and code used for the experiments presented in this chapter is available on GitHub[1].

The contributions of this chapter are:

- **Evaluation of Different Prompting Techniques for LLMs:** An evaluation of a wide range of zero-shot and few-shot, as well as rule-based prompts, comparing their impact on the entity matching task, building on an exploratory study [Narayan et al., 2022]. The sensitivity of different LLMs to changes in the prompt is further investigated, and it is shown that for some LLMs the performance is stable while it fluctuates strongly for others.
- **Comparison of LLMs to PLMs:** It is shown that GPT4 without any task-specific training data outperforms fully fine-tuned PLMs on 3 out of 4 e-commerce datasets and achieves a comparable performance for bibliographic data.
- **Comparison of Hosted and Open-Source LLMs:** A cost analysis of hosted LLMs is performed and it is shown that open-source LLMs can achieve a similar F1 performance as hosted LLMs given that a small amount of task-specific training data or matching knowledge in the form of rules is available.

Section 9.2 reviews the existing literature on the emerging field of using LLMs for entity matching. Section 9.3 introduces the experimental setup. Section 9.4 compares prompt designs and LLMs in the zero-shot setting, while Section 9.5 investigates whether model performance can be improved by providing demonstrations for in-context learning, adding matching knowledge in the form of natural

[1] https://github.com/wbsg-uni-mannheim/MatchGPT/tree/main/LLMForEM

language rules, and fine-tuning. Section 9.6 presents a cost analysis of the hosted LLMs and a runtime analysis of all LLMs. Finally, Section 9.7 concludes the chapter and summarizes the implications of the findings.

The work presented in this chapter builds on work previously published in [Peeters and Bizer, 2023] and has been published in [Peeters et al., 2025].

9.2 Related Work

At least since the release of ChatGPT by OpenAI, LLMs have become a focus in many research fields and business applications [Zhao et al., 2023]. The research area on entity linking has started to experiment with LLMs [Shi et al., 2024, Xin et al., 2024] and in the area of data integration many researchers have also turned to these models due to their strong zero-shot and few-shot performance compared to fine-tuning PLMs with thousands of labeled examples. For example, Brinkmann et al. [Brinkmann et al., 2023b] investigate the usefulness of LLMs for the task of product information extraction. Korini and Bizer [Korini and Bizer, 2023] apply LLMs for the task of column type annotation in tables.

Narayan et al. [Narayan et al., 2022] were the first to experiment with using an LLM (GPT3) for entity matching as part of a wider study that also covered data engineering tasks such as schema matching and missing value imputation. They showed that this specific LLM was able to reach the matching performance of a fully fine-tuned Ditto with just ten few-shot demonstration on multiple benchmarks.

The work presented in this chapter builds on this groundlaying work. Compared to Narayan et al., this chapter evaluates a wider range of LLM models using a larger set of prompt designs. The chapter presents a deeper analysis of the prompt sensitivity of the models and the prompt-to-model fit. It further investigates the utility of additional methods such as rule learning and LLM fine-tuning specifically for entity matching, which were not covered in previous work. The research works listed in the following are concurrent work to the contributions made in this chapter.

Zhang et al. [Zhang et al., 2023] explore the potential of LLMs in the context of data processing tasks. They discuss various prompting techniques such as zero-shot, few-shot, and batch prompting and provide an experimental evaluation of these techniques comparing various LLMs, including GPT3.5 and GPT4, with Ditto [Li et al., 2020] and Magellan [Konda et al., 2016] amongst others. The authors show that GPT4 achieves similar performance to a fully fine-tuned Ditto and underline the intrinsic interpretability of LLM decisions due to their ability to explain their decisions, as well as their adaptability to many tasks using a small amount of in-context examples. They conclude that LLMs have significant potential for entity matching processes, but bring several limitations that need to be addressed, like computational efficiency and cost concerns.

In their concurrent work, Zhang et al. [Zhang et al., 2024a] further experiment with instruction fine-tuning of a Llama2 model for multiple data preparation tasks at once. The relevant tasks are error detection, data imputation, schema matching, entity matching, and column type annotation. The model is small enough to be operated on a single consumer-grade GPU. Before the intstruction tuning step, the authors pre-tune the model on two more general datasets that are not related to the data integration domain. Their results show that a fine-tuned 13B Jellyfish model is able to achieve a similar matching performance as GPT4 and can even outperform it on some of the tested entity matching benchmarks.

Sisaengsuwanchai et al. [Sisaengsuwanchai et al., 2023] investigate the impact of various prompt engineering techniques on the performance of ChatGPT, including a cost analysis for each prompt. The authors test various ways of framing the prompt for entity matching including format of the record attributes, prompting for similarity scores, and few-shot prompting. They conclude that prompt engineering has a large impact on performance, but they also highlight that this varies depending on the used datasets, resulting in no prompt performing best on all datasets. Instead, they state that it is essential to find a good balance between finding the right prompt for the task and managing the costs.

Fan et al. [Fan et al., 2024b] present a cost-effective approach, BatchER, which consists of two components. Firstly, they experiment with batching multiple entity matching decisions together, and secondly, they add in-context demonstrations to reduce the cost of in-context learning that is incurred when only making a single decision in a prompt. The authors investigate various methods along the two components and conclude that diversity-based batching, based on selecting diverse record pairs from different clusters of a DBSCAN clustering together with a covering-based demonstration selection strategy achieves the best performance while minimizing cost. The covering-based strategy involves calculating relevance scores between the record pairs to be queried and the existing labeled training data to subsequently select demonstrations that collectively cover the feature space of relevant demonstrations.

Li et al. [Li et al., 2024] propose a framework that uses traditional similarity metrics such as Jaccard to select a set of record pairs for an LLM to solve using an uncertainty-based approach termed *Greedy Approximation Question Selection*. Their approach is similar to a blocking method as they try to keep the costs of using LLMs for the entity matching task below a specific budget by ensuring to give only the most relevant record pairs to the LLM.

Wang et al. [Wang et al., 2024] propose a framework to evaluate how LLMs perform on entity matching tasks, specifically focusing on three key paradigms: *Match*, *Compare*, and *Select*. *Match* means directly identifying whether two entities are the same, while *Compare* first compares multiple attributes of entities and then makes a decision based on the comparisons. Finally, *Select* presents a set of candidate entities, with the LLM having to select the one that matches a given entity. They combine these paradigms in their method called ComEM, which uses a smaller LLM to first rank and filter potential matches using the matching and comparing strategies to limit costs, while employing a larger LLM for the final selection step. As a result of this process, the method is mainly suitable for a clean-clean matching scenario as it is based on finding a single match across datasets and does not explicitly consider the case that multiple matches may exist. The authors show that they are able to outperform the self-supervised contrastive framework Sudowoodoo [Wang et al., 2023] on 7 of 8 benchmarks with their method.

Wadhwa et al. [Wadhwa et al., 2024] use large LLMs like GPT4 and Alpaca to solve a set of entity matching problems from the training sets of benchmarks, while providing the correct label and further prompt the models to also provide a chain-of-thought style natural language explanation of their decision. They then make use of these generated explanations to fine-tune a smaller language model, Flan-T5 base, and observe large improvements of 10% F1 on average compared to not providing explanations for this model in cross-domain generalization.

Steiner et al. [Steiner et al., 2024] focus on enhancing the capabilities of LLMs for entity matching through fine-tuning techniques that refine both the representation and generation of training examples. The experimental approach is multi-dimensional, examining how different training example representations, particularly those augmented with LLM-generated explanations, can improve entity matching performance and generalization across domains like products and publications. The study further explores the impact of training example selection strategies, using LLMs to filter misleading examples and generate additional synthetic examples. Their work directly builds on the experiments performed in this chapter, specifically the fine-tuning experiments in Section 9.5.

9.3 Experimental Setup

This section provides details about the large language models, the benchmark datasets, the serialization of entity descriptions, and the evaluation metrics that are used in the experiments.

Large Language Models:

Three hosted LLMs from OpenAI and three open-source LLMs that can be run on local GPUs are compared:

- **gpt-4o-mini-2024-07-18 (GPT-mini):** This hosted LLM from OpenAI offers lower API usage fees compared to GPT-4 and GPT-4o. It has a context window of 128K tokens. The training data cut-off date is October 2023.
- **gpt4-0613 (GPT-4):** This version of OpenAI's GPT-4 model from June 2023 has a context window of 8192 tokens. The training data cut-off date is September 2021.
- **gpt-4o-2024-08-06 (GPT-4o):** GPT-4o is OpenAI's flagship model with a 128K context and a training data cutoff date of October 2023.

- **Llama-2-70b-chat-hf (Llama2):** This Llama2 version[2] from Meta has 70B parameters and has been optimized for dialogue use cases. It has a context window of 4K tokens, and the training data cut-off date is September 2022.
- **Meta-Llama-3.1-70B-Instruct (Llama3.1):** This Llama3.1 model from Meta has 70B parameters and is optimized for dialogue use cases. It has a context window of 128K tokens and a training data cut-off date of December 2023.
- **Mixtral-8x7B-Instruct-v0.1 (Mixtral):** Mixtral is an open-source model that consists of 8 smaller models. It is developed by Mistral AI[3] and has a context window of 32K.

The following sections and tables use the model naming scheme introduced in the brackets above. The langchain[4] library is used for interaction with the OpenAI API and template-based prompt generation. The temperature parameter is set to 0 for all LLMs to reduce randomness. The temperature parameter adjusts the randomness of the model's outputs by scaling the logits before applying the softmax function. The experiments with open-source experiments are run on a local machine with an AMD EPYC 7413 processor, 1024GB RAM, and four nVidia RTX6000 GPUs.

The performance of the LLMs is compared to two PLM-based matching baselines, the RoBERTa-base [Liu et al., 2019b] model, which has been shown to reach high performance on the entity matching task (see Sections 7.4.3 and 8.4.3 and [Zeakis et al., 2023, Li et al., 2020, Peeters and Bizer, 2021]). The RoBERTa model is fine-tuned for entity matching on the respective development sets. The second baseline PLM is the Ditto [Li et al., 2020] system. Ditto introduces various data augmentation and domain knowledge injection modules. A RoBERTa-base language model is used at its core for the following experiments, and the data augmentation modules proposed in the original paper are applied where available.

RoBERTa-base is selected as a representative model for directly comparing PLMs to LLMs. Ditto combines this PLM with additional entity matching-specific functionality. Ditto also outperforms earlier matchers such as Deepmatcher [Mudgal et al., 2018] and DeepER [Ebraheem et al., 2018], and performs within a 2% F1 range compared to more complex methods such as SETEM [Ding et al., 2024] or HierGAT [Yao et al., 2022] on the datasets selected below.

The following benchmarks are used for the experiments in this chapter. The attributes used for the serialization of entity descriptions are presented in brackets after each benchmark name.

The first benchmark dataset used for the experiments is the most challenging version of the WDC Products benchmark (*brand*, *title*, *currency, price*), containing 80% corner cases. The benchmark was released at the end of the year 2022 after the training data cutoff date of the LLMs. The Abt-Buy (*title*, *price*), Walmart-Amazon

[2] https://huggingface.co/meta-llama/Llama-2-70b-chat-hf

[3] https://mistral.ai/

[4] https://www.langchain.com/

Table 9.1: Dataset statistics. In-context example selection and fine-tuning are performed on training and validation sets. Prompts are evaluated on test sets.

Dataset	Training set		Validation set		Test set	
	# Pos	# Neg	# Pos	# Neg	# Pos	# Neg
(WDC) - WDC Products	500	2,000	500	2,000	259	989
(A-B) - Abt-Buy	616	5,127	206	1,710	206	1,000
(W-A) - Walmart-Amazon	576	5,568	193	1,856	193	1,000
(A-G) - Amazon-Google	699	6,175	234	2,059	234	1,000
(D-S) - DBLP-Scholar	3,207	14,016	1,070	4,672	250	1,000
(D-A) - DBLP-ACM	1,332	6,085	444	2,029	250	1,000

(*brand*, *title*, *model number*, *price*) and Amazon-Google (*brand*, *title*, *price*) benchmarks using the versions released with the Deepmatcher paper [Mudgal et al., 2018] represent two-source entity matching benchmarks from the product domain. The DBLP-Scholar (*authors*, *title*, *venue*, *year*) and DBLP-ACM (*authors*, *title*, *venue*, *year*) benchmarks in their version from the Deepmatcher paper represent the task of matching bibliographic entries between two sources. WDC Products and Walmart-Amazon contain duplicates for some entities within the same dataset, representing a dirty-dirty matching scenario [Christophides et al., 2015], while the other datasets represent a clean-clean matching scenario.

Splits and Sampling

For WDC Products, the training, validation, and test split of size small [Peeters et al., 2024b] are used. The same evaluation splits established in the DeepMatcher paper [Mudgal et al., 2018] are used for the other benchmark datasets. Due to the large number of experiments performed against the OpenAI API and as these experiments, especially for long prompts and for using the GPT4 model, result in relevant API usage fees, all test sets are down-sampled to approximately 1250 entity pairs. Table 9.1 provides statistics on the number of positive pairs (matches) and negative pairs (non-matches) in the development and down-sampled test sets of all benchmarks used in the experiments.

Serialization

For the serialization of pairs of entity descriptions into prompts, each entity description is serialized as a string by concatenating attribute values. Figure 9.1 shows an example of this serialization practice for a pair of product offers. The same serialization method is applied for the bibliographic data.

For the serialization of pairs of entity descriptions (records) into prompts, each entity description is serialized into a single string by concatenating its attribute values using blanks as deliminator, e.g. $serialize(e)$:= Val_{A1} Val_{A2} ... Val_{An}.

Figure 9.1 shows an example of this serialization practice for a pair of product offers. The same serialization method is applied for the bibliographic data. All datasets contain textual attributes, e.g., the *title* of a product or publication, as well as numerical attributes like *price* and *year*. The decision was made not to add the names of the attributes themselves to the serialization string, as this negatively affected performance in preliminary experiments.

Evaluation

The responses gathered from the models are natural language text. To decide whether a response refers to a positive matching decision regarding a pair of entity descriptions, lower-casing is applied to the answer, which is subsequently parsed for the word *yes*. If not found, it is assumed that the model decides on not matching. This approach, while simple, is nevertheless effective as shown by Narayan et al. [Narayan et al., 2022].

9.4 Zero-shot Prompting

In the first scenario, the impact of different prompt designs on the entity matching performance of the LLMs is analyzed. The prompt sensitivity of the different models for the entity matching task is further investigated, and finally, the performance of the LLMs is compared to the PLM baselines.

Prompt Building Blocks

Prompts are constructed as a combination of smaller building blocks to allow the systematic evaluation of different prompt designs. Each prompt consists of at least a task description and the serialization of the pair of entity descriptions to be matched. In addition, the prompts may contain a specification of the output format. Alternative task descriptions that formulate the task as a question using simple or complex wording combined with domain-specific or general terms are evaluated. The alternative task descriptions are as follows:

- **domain-simple:** "Do the two product descriptions match?" / "Do the two publications match?"
- **domain-complex:** "Do the two product descriptions refer to the same real-world product?" / "Do the two publications refer to the same real-world publication?"
- **general-simple:** "Do the two entity descriptions match?"
- **general-complex:** "Do the two entity descriptions refer to the same real-world entity?"

A specification of the output format may follow the task description. For this, two formats are evaluated: (1) *free* which does not restrict the answer of the LLM,

Table 9.2: Average F1-scores over all datasets for the zero-shot experiments.

Prompt	All Datasets (Average F1)					
	GPT-mini	GPT-4	GPT-4o	LLama2	Llama3.1	Mixtral
domain-complex-force	85.29	88.91	**87.00**	66.60	**84.87**	**68.29**
domain-complex-free	**85.40**	**89.46**	80.31	**69.69**	72.06	62.13
domain-simple-force	50.41	86.10	82.72	57.24	60.52	41.65
domain-simple-free	33.65	87.92	63.53	50.40	36.94	43.20
general-complex-force	83.50	87.94	85.02	63.89	83.26	59.51
general-complex-free	83.13	87.85	55.81	62.73	80.54	61.50
general-simple-force	52.88	81.12	83.65	62.31	72.65	33.59
general-simple-free	45.49	85.07	64.67	52.77	63.38	36.12
Narayan-complex	56.13	86.70	50.65	56.34	36.16	32.04
Narayan-simple	75.15	86.92	45.64	67.81	37.32	30.94
Mean	65.10	86.80	69.90	60.98	62.77	46.90
Standard deviation	18.45	2.26	14.86	6.18	18.54	13.68

and (2) *force* which instructs the LLM to "Answer with 'Yes' if they do and 'No' if they do not". The prompt continues with the entity description pair to be matched, serialized as presented in Section 9.3. Figure 9.1 contains an example of a complete prompt that implements the prompt design *general-complex-free*. In addition to the prompts generated using these building blocks, the prompt designs proposed by Narayan et al. [Narayan et al., 2022] are evaluated. The main difference of these designs compared to the ones created from building blocks is that they put the serialization of the entities before the task description, while the latter order the task description and the serialization the other way around.

Effectiveness

Table 9.3 shows the results for each dataset separately. Table 9.2 shows the results of the zero-shot experiments averaged across all datasets. Regarding overall performance, the GPT4 model outperforms all other LLMs on all product datasets by at least 1% F1 achieving an absolute performance of 89% or higher on 5 of 6 datasets without requiring any task-specific training data. On the publication datasets, GPT-4o achieves nearly the same performance (0.1-1% F1). This gap increases on the product datasets to 1-3% F1, making the more recent model marginally worse than GPT4. GPT-mini performs up to 6% F1 worse than GPT-4o with only a marginal performance difference on 4 of 6 datasets. Among open-source LLMs, Llama3.1 consistently outperforms Llama2 by 1-21% F1. Llama3.1's performance is comparable to GPT-mini on all datasets. The Mixtral model performs less effectively on this task, lagging behind the other open-source models by 7-16% on 4 datasets. In summary, the results indicate that locally run open-source LLMs can perform similarly to OpenAI's GPT-mini model given that the right prompt is selected. However, if maximum performance is desired, none of the other LLMs can match GPT-4 in a zero-shot setting.

Table 9.3: Results (F1) of the zero-shot experiments for all datasets. Best results are set bold, second best are underlined.

Prompt	WDC Products						Abt-Buy						Walmart-Amazon					
	GPT-mini	GPT-4	GPT-4o	Llama2	Llama3.1	Mixtral	GPT-mini	GPT-4	GPT-4o	Llama2	Llama3.1	Mixtral	GPT-mini	GPT-4	GPT-4o	Llama2	Llama3.1	Mixtral
domain-complex-force	80.84	88.35	**87.64**	65.23	**83.67**	**53.37**	90.95	95.15	90.47	57.59	**89.84**	**82.20**	86.36	89.00	84.04	46.70	**84.85**	**70.44**
domain-complex-free	80.00	**89.61**	67.35	**69.09**	50.86	51.98	**91.93**	**95.78**	89.35	64.13	78.90	78.07	86.28	89.33	82.91	51.96	69.87	54.42
domain-simple-force	21.35	83.72	81.53	23.89	33.22	8.43	77.55	93.56	91.77	65.82	79.77	51.96	48.84	88.78	83.84	52.17	54.81	27.56
domain-simple-free	16.85	84.50	42.99	24.75	1.59	14.81	60.81	94.38	78.24	63.43	34.40	52.30	43.82	88.67	56.00	40.15	17.76	24.43
general-complex-force	78.80	85.83	87.02	66.02	81.89	42.04	89.88	94.40	90.67	56.70	88.11	79.02	**86.58**	**89.67**	83.67	44.69	84.14	50.56
general-complex-free	**81.15**	86.72	23.86	67.59	67.81	52.22	87.73	94.87	72.00	55.77	87.31	81.27	85.99	89.45	45.85	44.95	83.43	53.82
general-simple-force	20.71	77.39	82.48	46.54	62.57	9.89	80.67	93.23	**93.95**	**82.03**	88.72	54.04	46.33	86.41	**86.65**	**63.91**	72.05	21.20
general-simple-free	18.84	83.41	41.77	39.30	44.24	12.03	73.05	92.77	83.80	77.21	80.00	58.86	37.34	88.60	62.50	56.03	60.69	19.63
Narayan-complex	47.31	81.23	31.89	44.97	9.09	21.05	79.89	92.13	58.59	68.44	34.40	40.00	41.46	83.37	24.00	57.74	12.50	15.17
Narayan-simple	71.01	81.91	21.91	52.72	9.16	15.33	86.63	92.42	57.34	73.99	35.06	37.80	69.28	84.72	20.28	63.32	18.69	11.65
Mean	51.69	84.27	56.84	50.01	44.41	28.12	81.91	93.87	80.62	66.51	69.65	61.55	63.23	87.80	62.97	52.16	55.88	34.89
Standard deviation	27.96	3.42	25.66	16.25	28.83	18.31	9.22	1.17	13.01	8.50	23.24	16.32	20.44	2.08	24.39	7.69	27.56	19.39

Prompt	Amazon-Google						DBLP-Scholar						DBLP-ACM					
	GPT-mini	GPT-4	GPT-4o	LLama2	Llama3.1	Mixtral	GPT-mini	GPT-4	GPT-4o	LLama2	Llama3.1	Mixtral	GPT-mini	GPT-4	GPT-4o	LLama2	Llama3.1	Mixtral
domain-complex-force	70.98	75.61	**73.56**	**57.93**	**73.99**	**40.98**	**86.11**	88.44	**89.76**	**85.46**	84.29	75.40	96.51	96.90	96.53	86.69	92.59	87.33
domain-complex-free	**72.18**	75.57	61.17	56.29	60.59	24.91	86.06	89.78	84.03	85.11	80.87	**77.75**	95.97	96.71	**97.06**	91.57	91.24	85.66
domain-simple-force	24.55	75.32	58.00	29.86	19.33	13.39	41.51	77.21	84.35	84.07	77.17	60.39	88.65	98.03	96.85	87.62	**98.81**	88.15
domain-simple-free	19.48	74.51	35.85	17.16	5.76	17.69	7.63	88.20	73.79	81.53	67.69	59.67	53.30	97.28	94.29	75.35	94.41	**90.32**
general-complex-force	65.25	74.91	70.98	53.59	69.36	31.65	85.82	87.22	87.54	79.67	85.97	66.15	94.66	95.60	90.25	82.67	90.09	87.63
general-complex-free	64.09	74.38	20.44	49.48	68.45	29.45	85.66	87.50	76.85	76.42	**86.32**	68.01	94.16	94.16	95.87	82.18	89.93	84.23
general-simple-force	25.53	53.60	56.28	49.32	33.87	11.24	54.19	78.26	85.65	66.37	80.27	31.65	89.87	97.85	96.86	65.71	98.41	73.50
general-simple-free	17.56	66.67	32.19	37.79	25.35	12.70	40.75	81.47	72.20	51.06	73.99	38.59	85.40	97.47	95.58	55.21	95.98	74.88
Narayan-complex	18.45	**76.38**	13.90	39.63	10.40	3.36	56.34	**89.82**	78.82	42.99	55.27	27.40	93.31	97.27	96.67	84.26	95.32	85.26
Narayan-simple	42.24	75.70	5.71	48.71	6.58	2.51	84.12	88.37	72.82	70.47	59.40	35.06	**97.60**	**98.41**	95.77	**97.62**	95.04	83.30
Mean	42.03	72.27	42.81	43.98	37.37	18.79	62.82	85.63	80.58	72.32	75.12	54.01	88.94	96.97	95.57	80.89	94.18	84.03
Standard deviation	22.39	6.76	23.16	12.23	26.51	11.99	25.87	4.53	6.15	14.07	10.43	18.01	12.43	1.19	1.94	11.87	3.01	5.29

Sensitivity

Small variations in the prompts can have a large impact on the overall task performance [Narayan et al., 2022, Zhao et al., 2021, Liu et al., 2023]. This prompt sensitivity is measured as the standard deviation (SD) of the F1 scores of a model over all 10 prompt designs. This standard deviation is listed in the lower section of Tables 9.3 and 9.2. When comparing the prompt sensitivity of the models, the GPT4 model is most invariant to the wording of the prompt (mean standard deviation 2.26) while also achieving high results with most of the prompt designs. Comparing the sensitivity of GPT4 with all other models shows that they have a significantly higher prompt sensitivity (standard deviation 6.18 to 18.54 in Table 9.2).

Prompt to Model Fit

The best result for each model is listed in bold in Table 9.3, the second best result is underlined. This highlighting shows that there is no prompt design that performs best for most models. As a result, a general statement of how to design a prompt for the entity matching task cannot be made. Although the presented analysis is not exhaustive with regards to all potentially possible prompt designs, the results indicate that the best prompt depends on the model/dataset combination. While a good performing prompt can be found by testing a set of pre-defined prompts, automated approaches for prompt tuning and evolution could still further improve the results [Wang et al., 2022c, Fernando et al., 2024].

Comparison to PLM Baselines

The zero-shot performance of LLMs is compared to the performance of two PLM-based matchers: a fine-tuned RoBERTa model [Liu et al., 2019b] and a fine-tuned Ditto [Li et al., 2020] system. Table 9.4 shows the overall best results for each LLM compared to the two PLM-based matchers on all datasets. For three of the six datasets, GPT4 achieves higher performance than the best PLM baseline (2.65-4.71% F1), while the performance for the other three datasets is 3.69, 4.49 and 0.73% F1 lower. This shows that GPT4 without using any task-specific training data is able to reach comparable results or even outperform PLMs that were fine-tuned using thousands of training pairs (see Table 9.1). The reliance on large amounts of task-specific training data to achieve good performance is one of the main shortcomings of fine-tuned PLMs. The experiments in this section show that LLMs are a viable choice if task-specific training data is unavailable or costly.

Generalization

The previous chapters have demonstrated that PLM-based matchers struggle with generalizing to *unseen* (out-of-distribution) entities. In another set of experiments, each of the previously fine-tuned RoBERTa and Ditto models is applied, apart from

Table 9.4: Comparison of F1 scores of the best zero-shot prompt per model with PLM baselines. The "unseen" rows correspond to training on the dataset named in the column and applying the model to the WDC Products test set.

	WDC	A-B	W-A	A-G	D-S	D-A
GPT-mini	81.15	91.93	86.58	72.18	86.11	97.60
GPT-4	**89.61**	**95.78**	**89.67**	76.38	89.82	98.41
GPT-4o	87.64	93.95	86.65	73.56	89.76	97.06
Llama2	69.09	82.03	63.91	57.93	85.46	97.62
Llama3.1	83.67	89.84	84.85	73.99	86.32	98.81
Mixtral	53.37	82.20	70.44	40.98	77.75	90.32
RoBERTa	77.53	91.21	87.02	79.27	93.88	**99.14**
Ditto	84.90	91.31	86.39	**80.07**	**94.31**	99.00
Δ best LLM/PLM	4.71	4.47	2.65	-3.69	-4.49	-0.33
RoBERTa unseen	-	55.52	36.46	31.00	29.64	16.25
Ditto unseen	-	48.74	31.55	33.12	32.82	29.00
Δ RoBERTa unseen	-	-22.01	-41.07	-46.53	-47.89	-61.28
Δ Ditto unseen	-	-36.16	-53.35	-51.78	-52.08	-55.90

the one fine-tuned on WDC Products, to the WDC Products test set, which contains a different set of products that are unseen to these fine-tuned models. The results of these experiments are reported in the "RoBERTa Unseen" and "Ditto Unseen" rows at the bottom of Table 9.4. Compared to fine-tuning directly on the WDC Products development set (84.90% F1 for Ditto), the transfer of fine-tuned models leads to large drops in performance ranging from 36 to 56% F1 for Ditto and 22 to 61% F1 for RoBERTa. All LLMs achieve at least 8% F1 higher performance than the best transferred PLM while GPT4 outperforms the best PLM by 40% to 68%. These results show that LLMs have a general capability to perform entity matching as a result of their training procedure, while PLM-based matchers are fitted closely to entities of specific datasets after fine-tuning but are not capable of performing the task in a zero-shot manner.

9.5 Adding Training Data

Task-specific training data in the form of matching and non-matching entity pairs can be used to (1) add demonstrations to the prompts, (2) learn textual matching rules, and (3) fine-tune the LLMs. This section explores how the zero-shot results can be improved using task-specific training data. To make the results comparable, the same development sets (see Table 9.1) that were used to fine-tune the PLM-based matchers are used for all experiments in this section.

9.5.1 In-Context Learning

For the in-context learning experiments, each LLM is provided with a set of task demonstrations [Liu et al., 2022] as part of the prompt to help guide the model's decisions. The provided demonstrations are then followed by the specific entity description pair for which the model should generate an answer. Figure 9.2 shows an example of an in-context learning prompt with a single positive and a single negative demonstration. The number of demonstrations in each prompt is varied from 6 to 10 with an equal amount of positive and negative examples. For the selection of the demonstrations, three different heuristics are compared:

Task Desc.	Do the following two product descriptions match?
Demonstr-ations	Product 1: 'Title: DYMO D1 19 mm x 7 m' Product 2: 'Title: Dymo D1 (19mm x 7m – BoW)'
Answer	Yes.
Task Desc.	Do the following two product descriptions match?
Demonstr-ations	Product 1: 'Title: DYMO D1 Tape 24mm' Product 2: 'Title: Dymo D1 19mm x 7m'
Answer	No.
Task Desc.	Do the following two product descriptions match?
Task Input	Product 1: 'Title: DYMO D1 – Roll (1.9cm x 7m)' Product 2: 'Title: DYMO D1 Tape 12mm x 7m'

Figure 9.2: Example of a prompt containing a positive and a negative demonstration before asking for a decision.

- **Random**: As a baseline heuristic, the task demonstrations are drawn randomly from the training set of the respective benchmark.
- **Related**: Related demonstrations are selected from the training set of the respective benchmark with the idea of presenting correct matching decisions on highly similar products. This is done by calculating the Generalized Jaccard[5] similarity between the string representation of the pair to be matched and all positive and negative pairs in the corresponding training set. Afterwards, the pairs are sorted by similarity and the most similar positive and negative pairs are selected as demonstrations.
- **Hand-picked**: The hand-picked demonstrations were selected by a data engineer with the goals of being diverse and potentially helpful for corner case decisions. For the four datasets in the product domain, these examples are

[5] https://anhaidgroup.github.io/py_stringmatching/v0.3.x/GeneralizedJaccard.html

Table 9.5: Mean results for the in-context learning.

Prompt		All Datasets (Mean F1)					
	Shots	GPT4-mini	GPT4	GPT4o	LLama2	Llama3.1	Mixtral
Fewshot-related	6	73.76	90.24	90.41	65.44	82.12	50.51
	10	76.56	**90.80**	**91.21**	62.69	85.85	53.25
Fewshot-random	6	77.86	89.44	89.77	63.99	85.95	57.37
	10	80.51	89.05	89.85	65.62	**88.06**	53.94
Fewshot-handpicked	6	72.81	88.61	89.44	70.52	84.87	57.76
	10	73.93	88.76	89.52	69.91	87.60	51.03
Hand-written rules	0	81.49	87.65	86.36	51.22	85.57	**79.03**
Learned rules	0	84.14	86.64	84.96	44.23	84.11	74.53
Mean	-	77.63	88.90	88.94	61.70	85.51	59.68
Standard deviation	-	3.85	1.25	2.00	8.63	1.77	10.23
Best zero-shot	0	**85.51**	89.95	88.10	**76.01**	86.25	69.18
Δ Few-shot/zero-shot	-	-5.00	0.85	3.10	-5.49	1.81	-11.42
Δ Rules/zero-shot	-	-1.37	-2.29	-1.74	-24.78	-0.68	9.86

drawn from the WDC Products training set and were chosen to represent various product categories, as well as pairs where different attributes are important for the matching decision. For the two datasets from the publication domain, the examples are selected from the pool of training examples from DBLP-Scholar covering a range of distinct venues, publication years, and research areas.

Effectiveness

Table 9.5 shows the average results of the in-context experiments compared to the best zero-shot baselines. Table 9.6 shows the results for each dataset separately. Depending on the model/dataset combination, the usefulness of in-context learning differs. The GPT4 model, which is the best performing model in the zero-shot scenario, only improves significantly on Amazon-Google (9%) with marginal improvements on two datasets (0.6-1.5%) when supplying related demonstrations and an improvement of 2% on DBLP-Scholar with hand-picked demonstrations. GPT4's performance on WDC Products and Abt-Buy drops irrespective of the demonstration selection method, meaning that the model does not need the additional guidance in these cases. The GPT-4o model, on the other hand, sees improvements on all datasets when supplying demonstrations closing the gap to GPT-4 compared to zero-shot and even outperforming it for WDC Products. GPT-mini and Mixtral are not capable of using the in-context information as both models lose between 4 and 26% performance on most datasets. For all other LLMs, providing in-context examples usually leads to performance improvements, while the size of the improvements varies widely.

In summary, in-context learning improves the performance of LLMs for approximately 61% of the tested model/dataset combinations (see row Δ *Few-shot/zero-*

Table 9.6: Results (F1) of the few-shot and rule-based experiments. Best result is bold, second best is underlined.

Prompt		WDC Products						Abt-Buy						Walmart-Amazon					
	Shots	GPT-mini	GPT-4	GPT-4o	Llama2	Llama3.1	Mixtral	GPT-mini	GPT-4	GPT-4o	Llama2	Llama3.1	Mixtral	GPT-mini	GPT-4	GPT-4o	Llama2	Llama3.1	Mixtral
Fewshot-related	6	51.59	85.71	89.96	59.64	77.43	37.45	90.96	93.83	94.35	66.78	90.72	49.77	74.32	91.19	90.54	56.23	81.31	50.78
	10	57.87	86.45	**91.74**	53.23	**84.04**	41.93	**92.62**	94.35	94.87	63.78	92.80	52.44	74.85	**91.24**	**90.63**	56.15	86.89	54.70
Fewshot-random	6	67.48	86.55	87.69	57.25	77.53	53.80	88.19	94.12	95.26	62.99	92.65	55.75	75.00	88.89	88.62	60.68	86.34	53.68
	10	73.23	86.37	87.40	50.83	80.17	50.22	90.31	93.21	**95.55**	68.28	**93.75**	49.80	76.78	89.00	88.78	**67.44**	88.00	46.94
Fewshot-handpicked	6	55.56	87.23	88.28	58.26	75.45	48.12	88.06	93.36	95.49	71.56	89.23	47.79	70.59	88.84	88.94	66.52	86.65	50.64
	10	59.73	86.72	87.89	46.96	79.39	44.86	89.58	93.62	93.95	**82.21**	92.80	42.02	74.38	87.89	90.34	65.86	**88.53**	45.56
Hand-written rules	0	73.76	85.71	86.49	53.77	80.84	69.81	90.50	94.15	87.69	41.93	89.86	86.18	87.60	89.16	84.55	33.65	88.16	**83.08**
Learned rules	0	78.68	87.06	85.24	59.33	80.17	**70.25**	89.43	93.40	92.91	48.38	88.19	**88.83**	**87.94**	86.21	88.40	43.27	86.47	80.11
Mean	-	64.74	86.48	86.48	54.91	79.38	52.06	90.03	93.81	93.76	63.24	91.25	59.07	77.68	89.05	89.05	56.23	86.54	58.19
Standard deviation	-	9.25	0.52	0.52	4.22	2.44	11.38	1.48	0.40	0.39	11.95	1.89	16.83	6.04	1.54	1.54	11.30	2.13	13.83
Best zero-shot	0	**81.15**	**89.61**	87.64	**69.09**	83.67	53.37	91.93	**95.78**	93.95	82.03	89.84	82.20	86.58	89.67	86.65	63.91	84.85	70.44
Δ Few-shot/zero-shot	-	-7.92	-2.38	4.10	-9.45	0.37	0.43	0.69	-1.43	1.60	0.18	3.91	-26.45	-9.80	1.57	3.98	3.53	3.68	-15.74
Δ Rules/zero-shot	-	-2.47	-2.55	-1.15	-9.76	-2.83	16.88	-1.43	-1.63	-1.04	-33.65	0.02	6.63	1.36	-0.51	1.75	-20.64	3.31	12.64

Prompt		Amazon-Google						DBLP-Scholar						DBLP-ACM					
	Shots	GPT-mini	GPT-4	GPT-4o	Llama2	Llama3.1	Mixtral	GPT-mini	GPT-4	GPT-4o	Llama2	Llama3.1	Mixtral	GPT-mini	GPT-4	GPT-4o	Llama2	Llama3.1	Mixtral
Fewshot-related	6	61.58	84.27	82.95	62.48	65.28	39.11	75.83	88.00	86.44	69.11	80.18	53.82	88.25	98.41	**98.21**	78.37	97.78	72.12
	10	63.52	**85.21**	**83.46**	62.36	71.43	47.89	77.93	88.52	88.33	64.29	82.38	55.57	92.54	**99.01**	98.20	76.34	97.58	66.95
Fewshot-random	6	57.37	78.08	78.67	59.78	74.32	48.99	82.56	90.21	90.16	67.04	86.43	55.80	96.55	98.81	98.21	76.22	98.40	76.21
	10	60.56	78.76	78.61	59.92	**80.46**	46.17	84.98	89.30	**90.54**	69.63	**87.17**	56.13	97.19	97.66	98.21	77.64	98.80	74.39
Fewshot-handpicked	6	55.19	76.92	77.25	**64.65**	73.93	45.92	68.86	90.98	89.19	81.34	85.34	67.70	**98.61**	94.34	97.47	80.78	98.62	86.36
	10	58.49	76.57	77.22	63.89	79.52	49.18	62.96	**91.81**	90.06	73.56	86.13	55.61	98.42	95.97	97.66	86.96	**99.21**	68.97
Hand-written rules	0	66.37	72.47	73.83	46.39	71.90	**58.31**	76.74	87.34	89.30	53.72	84.82	**83.55**	93.95	97.09	96.30	77.88	97.85	**93.26**
Learned rules	0	68.28	73.50	72.39	59.77	71.88	48.55	84.00	89.42	78.59	8.43	81.95	78.38	95.42	90.25	92.25	46.21	95.97	81.04
Mean	-	61.42	78.22	78.22	59.91	73.59	48.02	76.73	89.45	89.45	60.89	84.30	63.32	95.12	96.44	96.44	75.05	98.03	77.41
Standard deviation	-	4.19	4.27	4.27	5.41	4.51	4.95	7.15	1.41	1.41	21.14	2.34	11.03	3.25	2.76	2.76	11.38	0.94	8.39
Best zero-shot	0	**72.18**	76.38	73.56	57.93	73.99	40.98	**86.11**	89.82	89.76	**85.46**	86.32	77.75	97.60	98.41	97.06	**97.62**	98.81	90.32
Δ Few-shot/zero-shot	-	-8.66	8.83	9.90	6.72	6.47	8.20	-1.13	1.99	0.78	-4.12	0.85	-10.05	1.01	0.60	1.15	-10.66	-0.01	-3.96
Δ Rules/zero-shot	-	-3.90	-2.88	0.27	1.84	-2.09	17.33	-2.11	-0.40	-0.46	-31.74	-1.50	5.80	-2.18	-1.32	-0.76	-19.74	-0.96	2.94

shot in Table 9.6). Providing demonstrations was not helpful for GPT4, which does not need the additional guidance on two datasets, as well as for the smaller models GPT-mini and Mixtral which suffer large performance drops on many datasets. As a result, the usefulness of in-context learning cannot be assumed but needs to be determined experimentally for each model/dataset combination.

Comparison of Selection Methods

The best demonstration selection method also varies depending on the dataset. The open-source LLMs achieve the best performance when random or hand-picked demonstrations are provided. In contrast, GPT-4 and GPT-4o achieve the highest scores on most datasets using related demonstrations, suggesting that these models are better able to understand and apply specific patterns from closely related examples to the current matching decision. The hand-picked demonstrations, while not helpful for the Llama models on their source dataset WDC Products, lead to improvements on all other product datasets. The same effect is visible for the hand-picked demonstrations transferred to DBLP-ACM.

9.5.2 Learning Matching Rules

In this set of experiments, instead of providing the models with in-context examples, a set of textual matching rules is provided in the prompt in order to guide the model to find the correct solution. Two kinds of rules are used in this regard, handwritten and learned rules. Handwritten rules are a set of binary rules created by defining which attributes need to match for the given domain to signify a match. The rules also inform the model of potential heterogeneity in these attributes, such as slight differences in surface form or value formats. For the learned rules, the hand-picked examples that were used for the in-context experiments are passed to GPT4, and the model is asked to generate matching rules from these examples. Similarly to the handwritten rules, they refer to specific attributes that should be matching and potential sources of heterogeneity that the GPT4 model extracted from the provided examples. A subset of these handwritten and learned rules for the product domain is depicted in Figure 9.3.

Effectiveness

Table 9.6 shows the results of providing matching rules in comparison to the best zero-shot prompt and the in-context experiments. The results show that GPT4 with matching rules does not improve on its best zero-shot performance and instead loses 1% to 3% F1 on all datasets. All other models see improvements on some datasets of 0.3% to 17% F1 over zero-shot depending on the model/dataset combination. Especially the Mixtral LLM, which has comparatively low performance compared to all other LLMs in the zero-shot and few-shot settings, significantly improves with the provision of rules on all datasets, gaining from 3 to 17% F1. In

Task Desc.	The following rules regarding product features need to be observed:
Hand-Written Rules	1. The brand of matching products must be the same if available. 2. Model names of matching products must be the same if available. 3. Model numbers of matching products must be the same if available. 4. Additional features of matching products must be the same if available. 5. Matching attributes may not have the exact same surface form due to different case, typos, value formats. 6. If an attribute is missing for one description, it is likely still a match if the existing attributes match.
Task Desc.	Do the following two product descriptions match? Answer with ‘Yes’ if they do and ‘No’ if they do not.
Task Input	Product 1: ‘Title: DYMO D1 – Roll (1.9cm x 7m)’ Product 2: ‘Title: DYMO D1 Tape 12mm x 7m’

Learned Rules	1. The product titles match if they both refer to the same core product, which means they have the same primary brand and product model … 2. Order of words and usage of separators (like hyphen, comma, slash, parentheses) does not matter … 3. The product titles match if they both contain the same key information, even if one product title contains additional details … 4. Exact matching of all specifications is not necessary. Titles match as long as the critical specs like model number, series, and brand are the same … 5. The product titles do not match if they refer to different models of the same brand … 6. …

Figure 9.3: Example of a prompt containing handwritten matching rules for the product domain. A subset of the learned rules is depicted below.

summary, the provision of matching rules can be helpful, especially for the open-source LLMs with Mixtral achieving its highest scores on all datasets using rules, but providing task demonstrations leads to higher performance gains than providing matching rules for all other models.

Sensitivity

Similarly to the zero-shot scenario, the prompt sensitivity of each LLM is measured across all few-shot and rule experiments. This standard deviation is listed in the lower section of Tables 9.5 and 9.6. When comparing the prompt sensitivity of the models with the zero-shot deviations across different prompt formulations, the average deviation from the mean has decreased for all models, showing that additional guidance in the form of demonstrations and rules leads to more robust results overall.

Table 9.7: Results for fine-tuning LLMs and subsequent transfer to all datasets. Left-most column shows the dataset used for fine-tuning.

		WDC	A-B	W-A	A-G	D-S	D-A
WDC Products	Llama2	66.81	75.98	72.83	54.77	41.74	28.86
	Llama3.1	72.05	83.47	76.92	63.97	65.25	80.91
	GPT-mini	88.89	92.49	88.61	77.78	86.82	97.28
Abt-Buy	Llama2	58.79	92.15	81.84	68.61	84.12	95.31
	Llama3.1	77.87	93.60	84.85	74.49	79.86	94.07
	GPT-mini	83.66	94.17	88.83	76.63	86.03	97.85
Walmart-Amazon	Llama2	49.71	88.13	90.57	66.50	64.57	87.74
	Llama3.1	51.12	89.92	91.01	73.76	82.62	95.41
	GPT-mini	72.64	94.94	**92.99**	82.06	89.31	97.85
Amazon-Google	Llama2	59.50	77.16	63.21	76.19	78.59	88.70
	Llama3.1	61.28	85.00	82.35	78.67	70.49	87.73
	GPT-mini	64.75	90.23	82.98	**87.11**	87.02	96.88
DBLP-Scholar	Llama2	29.41	49.48	59.52	46.55	92.80	97.45
	Llama3.1	35.11	74.27	77.15	58.59	92.37	97.84
	GPT-mini	57.70	84.07	84.02	73.33	**93.95**	97.64
DBLP-ACM	Llama2	6.15	29.96	29.60	16.22	66.84	99.20
	Llama3.1	15.33	49.82	41.90	25.00	83.37	**99.60**
	GPT-mini	31.54	85.25	67.99	49.43	89.70	99.40
Best zero-shot	Llama2	69.09	82.03	63.91	57.93	85.46	97.62
	Llama3.1	83.67	89.84	84.85	73.99	86.32	98.81
	GPT-mini	81.15	91.93	86.58	72.18	86.11	97.60
Δ best zero-shot	Llama2	-2.28	+10.12	+26.66	+18.26	+7.34	+1.58
	Llama3.1	-5.80	+3.76	+6.16	+4.68	+6.05	+0.79
	GPT-mini	+7.74	+3.01	+6.41	+14.93	+7.84	+1.80
Δ best GPT4	Llama2	-22.8	-3.63	+0.90	-0.19	+2.98	+0.79
	Llama3.1	-11.74	-2.18	+1.34	+2.29	+2.55	+1.19
	GPT-mini	-0.72	-0.84	+3.32	+10.73	+4.13	+0.99
Best GPT4	-	**89.61**	**95.78**	89.67	76.38	89.82	98.41

9.5.3 Fine-Tuning

In the next set of experiments, the GPT-mini, Llama2, and Llama3.1 models are fine-tuned using either the OpenAI API for gpt-mini or local hardware as described above for the Llama models. The training and validation sets of each dataset are used to train a fine-tuned model with the *domain-simple-force* prompt template and subsequently the fine-tuned models are applied with this same prompt template to all datasets test sets. GPT-mini is fine-tuned for 10 epochs using the default parameters suggested by OpenAI. For the Llama models, fine-tuning is performed for 10 epochs using 4-bit quantization to manage the high VRAM requirements of the 70B models together with Low-Rank Adaptation (LoRA) [Hu et al., 2022] for efficient model adaptation. The LoRA configuration includes the alpha parameter set to 16, a dropout rate of 0.1 and a rank (r) of 64. The learning rate is set to 2e-4.

Effectiveness

The results of the fine-tuned LLMs are shown in Table 9.7. The lower part of the table restates the best zero-shot and GPT4 results for comparison. When comparing the fine-tuning results with the best zero-shot performance (Section Δ best zero-shot in Table 9.7), a substantial improvement of 1% to 26% F1 is observable depending on the dataset for all models. Only the Llama models on WDC Products do not profit from fine-tuning. On four out of six datasets, the best fine-tuned Llama3.1 and GPT-mini models exceed the performance of zero-shot GPT4 by 1 to 10% F1 (see Section Δ *best GPT4* in Table 9.7).

In summary, fine-tuning the models leads to improved results compared to the zero-shot version of the model that rivals the performance of the best GPT4 prompts with the much cheaper GPT-mini model and consistently improves the performance of the Llama models by 1-26% F1 on 5 of 6 datasets, leaving Llama 3.1 only slightly behind GPT-mini on 4 datasets. Furthermore, the experiments show that the fine-tuned Llama models reach a similar performance or outperform GPT4 on 4 out of 6 datasets.

Generalization

A generalization effect is observed for the GPT-mini model fine-tuned on one dataset to datasets from related domains and across domains. Transferring models between related product domains leads to improved performance over the best zero-shot prompts for many combinations of datasets. The effect is especially visible for the combinations WDC Products, Abt-Buy and Walmart-Amazon that contain similar products. The transfer to Amazon-Google results in better performance than zero-shot for all of the mentioned product datasets. In contrast, the reverse transfer from Amazon-Google does not yield improved results. Furthermore, all GPT-mini models fine-tuned on the datasets from the product domain exhibit good generalization to the publication domain, resulting in improvements of 1-3% F1 over the best zero-shot result. Transferring fine-tuned models within the publication domain shows the same effect. The transfer does not work in the other direction as transferring a model fine-tuned for the publication domain leads to lower performance on the product datasets. For the Llama models this effect is only visible for some inter-product transfers for Llama2. Transfer of fine-tuned models to unseen benchmarks has received some research attention in the context of PLM-based matchers [Trabelsi et al., 2022, Tu et al., 2022] but often results in a significant loss of matching performance as shown in Table 9.4.

9.6 Cost and Runtime Analysis

Apart from pure matching performance, there are additional considerations such as data privacy requirements and the cost of using hosted LLMs which may result in the decision to use a less performant but cheaper hosted LLM or to run an

Table 9.8: Costs for hosted LLMs on WDC Products. The best performing prompt was selected for the analysis for each scenario.

	Zeroshot			6-Shot			10-Shot		
	GPT-mini	GPT-4	GPT-4o	GPT-mini	GPT-4	GPT-4o	GPT-mini	GPT-4	GPT-4o
F1 (Best prompt)	81.15	89.61	87.64	67.48	87.23	89.96	73.23	86.72	91.74
Mean # Tokens prompt	76	77	93	633	639	641	992	942	1,009
Mean # Tokens completion	89	40	1	2	2	2	2	2	2
Mean # Tokens combined	166	117	94	635	641	643	994	944	1,011
Token incr. to ZS	-	-	-	3.8x	5.5x	6.8	6x	8.1x	10.8x
Cost per prompt	0.006¢	0.474¢	0.024¢	0.01¢	2.056¢	0.162¢	0.015¢	3.037¢	0.254¢
Cost incr. to ZS (GPT-mini)	-	73x	3.8x	1.5x	319x	25x	2.4x	470x	39x
Cost incr. per Δ F1 to ZS	-	8.7x	0.6x	0.1x	52x	3x	0.3x	84x	3.7x

	Rules (written)			Rules (learned)			Fine-tune	
	GPT-mini	GPT-4	GPT-4o	GPT-mini	GPT-4	GPT-4o	Train	Inference
F1 (Best prompt)	73.76	85.71	86.49	78.68	87.06	85.24	-	88.89
Mean # Tokens prompt	213	214	213	815	817	815	97	88
Mean # Tokens completion	1	1	4	1	1	3	1	1
Mean # Tokens combined	214	215	217	816	818	818	98	89
Token incr. to ZS	1.3x	1.8x	2.3x	4.9x	7x	8.7x	0.6x	0.5x
Cost per prompt	0.003¢	0.649¢	0.057¢	0.012¢	2.458¢	0.207¢	0.280¢	0.003¢
Cost incr. to ZS (GPT-mini)	0.5x	101x	9x	1.9x	381x	32x	43x	0.5x
Cost incr. per Δ F1 to ZS	0.07x	22x	1.7x	0.8x	64x	7.8x	-	0.06x

open-source LLM on local hardware. The cost analysis presented in the following provides an overview of expected costs for hosted models. The purpose of the analysis is to give the reader general guidance of what to expect with regards to the cost dimension. A more in-depth analysis of costs including acquisition costs for GPUs and electricity for the open-source models is left to future work.

Costs: Table 9.8 lists the costs associated with the hosted LLMs across all experimental scenarios for the WDC Products dataset. The cost of using a hosted LLM is dependent on the length of the respective prompts, measured by the amount of tokens, and the current prices of the respective model. Thus, the results presented here are a snapshot as of August 2024 as the prices are subject to change. The costs of all used OpenAI models are compared along all of the previous prompting strategies. The prices for using the models at the time of comparison were as follows for 1 million prompt/completion tokens: \$0.15/\$0.60 for GPT-mini, \$30.00/\$60.00 for GPT-4, and \$2.50/\$10.00 for GPT-4o.

Table 9.8 shows that the in-context learning (6-shot, 10-shot) and the rule-based approaches (hand-written and learned) from Section 9.5 require between 1.3 and 11 times the amount of tokens per prompt compared to basic zero-shot prompting (see row *Token increase to ZS* in Table 9.8). For all of them, this is due to longer prompts, either because of the inclusion of few-shot demonstrations or rules. The fine-tuning approach, on the other hand, requires less tokens than zero

Table 9.9: Runtime in seconds per prompt (request) for all LLMs using the best prompts from the previous sections on the WDC Products dataset. Runtimes marked with * are for the quantized version of the model used for fine-tuning.

Model	Zeroshot	6-Shot	10-Shot	Rules (written)	Rules (learned)	Fine-Tune (Inference)
GPT-mini	1.54 s	0.46 s	0.51 s	0.47 s	0.47 s	0.46 s
GPT4	2.19 s	0.75 s	0.78 s	0.68 s	0.76 s	-
GPT4-o	0.51 s	0.48 s	0.53 s	0.48 s	0.49 s	-
Llama2	22.62 s	7.15 s	7.82 s	23.16 s	24.51 s	*0.30 s
Llama3.1	0.54 s	1.70 s	2.36 s	0.67 s	1.70 s	*0.30 s

shot, as the prompt chosen for fine-tuning uses the restricted output format *force* (see Section 9.4) whereas the best zero-shot prompt for GPT-mini uses the *free* format which allows the model to answer more verbosely. From a cost perspective, the in-context learning and the rule-based approaches increase the costs by 1.5 to 470 times compared to the cost of the zero-shot GPT-mini model. While GPT-mini is the cheapest model in this lineup, the GPT-4o model achieves significantly higher performance, often approaching or even surpassing GPT-4, at a fraction of GPT-4's cost. If many training examples are available, fine-tuning the GPT-mini model results in comparably high performance for a fraction of the cost of GPT-4o.

Runtime: Table 9.9 lists the average runtime per prompt for all LLMs. The selected prompts and the number of tokens used are the same as in Table 9.8. If the prompt allowed free-form answering, this leads to much longer runtimes compared to forcing the model to answer shortly. The large difference in runtimes between zero-shot Llama2 and Llama3.1 in Table 9.9 is an example of this. The runtimes of the hosted models are a snapshot of the API performance in August 2024 and may change at any time. Prompting GPT4 takes around 50% longer than the other two OpenAI models that have comparable runtimes if the answering scheme is the same. The locally hosted open-source LLM Llama2 requires the largest amount of time for most scenarios on the used local hardware (see Section 9.3), particularly when generating freely in zero-shot and rule-based cases, where its runtime is 10 to 33 times longer than that of GPT-4. On the other hand, the Llama3.1 model achieves a runtime comparable to the GPT models in most setups.

9.7 Conclusion

This chapter contributed a comparison of the effectiveness of prompting techniques and fine-tuning of LLMs, including an evaluation of the prompt sensitivity, as well as a comparison of open-source to hosted LLMs and LLMs to PLMs like Ditto [Li et al., 2020] to the field of entity matching.

The experiments showed that LLMs are a more robust and less task-specific training data dependent alternative compared to PLM-based matchers. The experiments using different prompt designs have shown that there is no single best prompt per model or per dataset, but that the best prompt needs to be estimated for each model/dataset combination. Adding demonstrations or matching rules to the prompts marginally profit models that already have a high zero-shot performance, while the performance of models with a comparatively low zero-shot performance increases.

The findings of this chapter can be summarized as follows: For use cases that do not involve many unseen entities and for which a decent amount of training data is available, PLM-based matchers are a suitable option which do not require much compute due to the smaller size of the models. For use cases that involve a relevant amount of unseen entities and for which it is costly to gather and maintain a decent-size training set, LLM-based matchers are the preferred choice due to their high zero-shot performance and ability to generalize to unseen entities. If using the best performing hosted LLMs is not an option due to their high usage costs, fine-tuning a cheaper hosted model is an alternative that can deliver a similar F1 performance. If using hosted models is no option due to privacy concerns, using an open-source LLM on local hardware can be an alternative providing slightly lower F1 performance given that some task-specific training data or domain-specific matching rules are available.

Compared to the pioneering work of Narayan et al. [Narayan et al., 2022] and concurrent work presented in Section 9.2, this chapter evaluates a wider range of LLM models using a larger set of prompt designs. In addition, it provides a deeper analysis of the prompt sensitivity of the models and the prompt-to-model fit. A comparison to the ComEM [Wang et al., 2024] system, which first filters and ranks possible matches across datasets before selecting a match without intricate prompt engineering, is not directly possible as the work presented in this thesis uses downsampled versions of the test sets and more recent LLMs than Wang et al. Given these limitations, all hosted LLMs and Llama3.1 achieve higher F1 values with the best prompt design in the zero-shot setting, as reported in this chapter, than the ones reported in the evaluation of the ComEM system on the relevant datasets. Wang et al. state that future work should incorporate prompt engineering into their proposed method, which would in turn allow a more thorough comparison of the impact of their method compared to the results presented in this thesis. Other concurrent work [Zhang et al., 2023, Sisaengsuwanchai et al., 2023] focusing on prompt engineering concludes similar findings as presented in this chapter regarding the necessity of a prompt search for each benchmark and model for best results, as well as the need to consider costs. The latter is a main concern of other recent research work on prompting [Li et al., 2024], with a focus on in-context learning [Fan et al., 2024b] and large language model fine-tuning [Zhang et al., 2024a] including the generation of explanations with more expensive LLMs to fine-tune smaller language models [Wadhwa et al., 2024, Steiner et al., 2024]. The next chapters of this thesis focus on the generation of similar explanations not for the purpose of fine-tuning, but explaining model decisions and automating error analysis in entity matching.

Part IV

Explainability and Error Analysis for Entity Matching

Chapter 10

Explaining Model Decisions for Entity Matching

10.1 Introduction

Deep learning-based approaches are ubiquitous in entity matching [Barlaug and Gulla, 2021]. Earlier non-neural frameworks like Magellan [Konda et al., 2016] that are based on attribute similarity-based features retain a certain level of interpretability of how the inputs translate to the outputs of the model. An example for this interpretability would be the importance value of a certain feature for, e.g., a logistic regression classifier that allows a statement of which features have a high relevance for the final decision, or a decision tree that can be easily interpreted by a human. This level of interpretability is no longer given for neural methods due to the black-box nature of these sub-symbolic methods. For a human observer, it is not possible to follow how the inputs to a neural model, e.g. all the tokens carrying the information about a product, to the output. When passing through the layers of the network, it is unclear to what extent certain information in the input is deemed important for a decision. Furthermore, as PLMs and LLMs are pre-trained on large corpora, it is not clear in how far information learned during pre-training plays a role in the decision compared to the actual tokens that are part of the current input. Some efforts have been made to explain sub-symbolic models for entity matching [Di Cicco et al., 2019, Baraldi et al., 2023, Paganelli et al., 2022] and are discussed in Section 10.2. Although these methods allow some insight into model matching decisions, they are based on either only explaining singular (local) model decisions for a specific record pair [Di Cicco et al., 2019, Baraldi et al., 2023] or focus on a specific model architecture and cannot be directly applied to other matching systems [Paganelli et al., 2022].

This chapter contributes two novel methods for explaining decisions of entity matching models to the field of entity matching. Both methods allow aggregating explanations of single model decisions into global insights about the models matching decisions, while existing work focuses on explaining decisions for single record pairs [Di Cicco et al., 2019, Baraldi et al., 2023]. The first method is based on the Mojito framework [Di Cicco et al., 2019] that itself is based on LIME [Ribeiro et al., 2016]. Mojito generates explanations for single entity matching model decisions, which the method presented in this thesis further aggregates into global insights of how the model performs entity matching. The second presented method is based on structured explanations generated by an LLM that can be automatically aggregated into global insights.

The first method investigates which words in the entity descriptions were considered relevant by the different models for specific matching challenges. For this, Mojito [Di Cicco et al., 2019] explanations are generated for sets of matching decisions that involve specific challenges. The generated Mojito word importance weights are then aggregated using manually annotated domain-specific word classes such as *model name*, *product attribute*, or *stop word*. The comparison of aggregated explanations of decisions of BERT-based models and Deepmatcher shows that the former are better at focusing on strong predictors such as *model numbers* while still being able to fall back to other word classes such as *model name* in cases where *model numbers* are not available.

The second method prompts an LLM to provide structured explanations for its decisions. These structured explanations contain attributes and tokens that the LLM considered relevant for its decision. In addition, each attribute in the explanations is accompanied by a model-generated importance and similarity score, which allows insights into the models matching decision for a specific pair of entity descriptions. These structured explanations can be subsequently automatically parsed and aggregated, similarly to the first method.

In summary, the contributions of this chapter are:

- **Method for Aggregating Mojito Explanations of Model Decisions into Global Insights:** A method that takes advantage of local explanations for model decisions from the recent Mojito explanation framework [Di Cicco et al., 2019] in combination with annotation of word classes to aggregate these explanations into global insights about the matching model. The results show that Transformer models can focus on strong predictors like model names or model numbers when they occur in record pairs and shift their focus to others if they are missing. The Deepmatcher RNN model and symbolic baselines show a more constant distribution of importance over all predictors. As it relies on an external explanation framework, the method is broadly applicable to matching classifiers.

- **Method for the Automatic Aggregation of LLM-based Structured Explanations into Global Insights:** A method that leverages an LLM to provide explanations for its own decisions in a structured format, including attributes relevant for the model decision accompanied by model-generated importance and similarity scores. These structured explanations can be automatically aggregated into global model insights without relying on external explanations and separate annotation of word classes.

Section 10.2 presents an overview of related work for explaining model decisions in the field of entity matching. Section 10.3 presents the first method for aggregating instance explanations based on Mojito. Section 10.4 presents the second method that aggregates structured explanations provided by an LLM to gain global insight into the model. Finally, Section 10.5 summarizes the contributions and findings of this chapter.

The work presented in this chapter has previously been published in [Peeters and Bizer, 2021] and in [Peeters et al., 2025] .

10.2 Related Work

This section gives an introduction to related work for explainability of black-box machine learning models and specific methods proposed for the entity matching task.

Explanations for Machine Learning Model Decisions

The explainability of machine learning models has been an important field of research since sub-symbolic or black-box models exist [Gilpin et al., 2018, Xu et al., 2019, Sun et al., 2021, Holzinger et al., 2022]. Such models are characterized by the inability of humans to understand the process that connects inputs to the model with its output, that is, its predictions [Xu et al., 2019]. Humans cannot attribute a prediction to certain input features or their combinations. The most prominent example of black box models are deep neural networks consisting of many connected layers that transform input features like text after conversion to the embedding space to a model decision [Goodfellow et al., 2016, Sun et al., 2021], e.g., a binary classification in the entity matching case.

Methods for explaining machine learning models are categorized into two families, (1) transparency by design and (2) post-hoc methods [Xu et al., 2019]. The former design the model in a way that makes them explainable as part of the model design, e.g. decision trees or neural networks containing an explanation component as part of the architecture. Post-hoc methods, on the other hand, try to explain model decisions after a black-box model has been trained. These methods can also be model-specific or model-agnostic, and they can be global or local. Local methods provide explanations for single input examples, while global methods try to explain the general workings of a model for a specific task, independently of a single example.

Examples of local agnostic methods are LIME [Ribeiro et al., 2016], which approximates a complex model locally with an interpretable surrogate, and Anchors [Ribeiro et al., 2018], which identify conditions under which predictions are invariant. Integrated gradients [Pruthi et al., 2020] is an example of a local method specific to neural networks, as it is based on tracing the gradients from input to output during backpropagation to identify the contributions of input features to output predictions. Examples of global methods include the works of Ibrahim et al. [Ibrahim et al., 2019], who develop techniques to generate explanations for the entire input space of a model by clustering local explanations, and Radulovic et al. [Radulovic et al., 2021], who propose methods for generating high-confidence interpretations for black-box models over all possible inputs by creating multiple specialized surrogate models in combination with correlating feature values to class labels to provide global explainability.

Explanations for Entity Matching Model Decisions

Understanding the decisions of a matching model is important for users to build trust towards the systems [Thirumuruganathan et al., 2019]. Explanations of model decisions can also be used for debugging matching pipelines. The prevalence of deep neural networks, especially PLMs over recent years in the field of entity matching, has led to research into the explainability of these matching systems [Ebaid et al., 2019, Di Cicco et al., 2019, Baraldi et al., 2023, Paganelli et al., 2022].

Ebaid et al. [Ebaid et al., 2019] propose a framework, ExplainER, which is composed of multiple model-agnostic explanation techniques like LIME [Ribeiro et al., 2016] and Anchors [Ribeiro et al., 2018] providing local explanations as well as the possibility of aggregating them into global insights via aggregation or if-then rules. ExplainER is based on similarity-based features like Jaccard string similarity and not all of its features are applicable to token-based textual models like the prevalent Transformer architecture that is part of most methods applied in this thesis.

DiCicco et al. [Di Cicco et al., 2019] present Mojito, an explainability method based on LIME [Ribeiro et al., 2016] adapted explicitly for the pair-wise entity matching case. LIME is based on slightly perturbing instances using, for example, word deletion to identify which words cause the label to change and learning a local surrogate model in the form of a linear regression using its coefficients as feature importances for the classifier to be explained. The Mojito method adds an additional perturbation method to LIME which is copying tokens between the records of a record pair. Mojito is used as the basis for the explainability method presented in Section 10.3.

Barlaug [Barlaug, 2023] presents a model-agnostic explainability method called LEMON based on LIME [Ribeiro et al., 2016]. Compared to LIME, the LEMON method provides explanations for both records in a pair, avoiding cross-record interaction effects, and further presents the concept of attribution potential, which allows users to understand what features would need to change to transform a non-matching decision into a matching decision.

Paganelli et al. [Paganelli et al., 2022] present an in-depth evaluation of BERT entity matching decisions using embedding similarity and attention scores in the layers. They conclude that the fine-tuning process is crucial as the pre-training knowledge encoded in the model is not enough to perform the task reliably. Although the Transformer does identify matching attributes, the actual decisions are based on contextual knowledge across multiple tokens rather than direct comparisons of single tokens between two records in a pair.

Baraldi et al. [Baraldi et al., 2023] present WYM, an example of an intrinsically locally interpretable system based on the idea of finding and extracting important decision units among entity descriptions for matchers during training. The authors concede that the usage of this method negatively impacts the performance of the matching systems as it is based on simple logistic regression, that is, increased explainability comes with the trade-off of reduced matching performance compared to using a neural method like Ditto [Li et al., 2020].

Research into explanations created by LLMs has recently come into focus in data integration and entity matching. In addition to the work presented in this thesis, Li et al. [Li et al., 2023a] use the generation of explanations successfully to improve the decisions of their Table-GPT model for table tasks, including entity matching. Wadwha et al. [Wadhwa et al., 2024] use LLM-generated explanations to augment training data to fine-tune smaller Transformer models, showing an increase in performance and out-of-domain generalization. Steiner et al. [Steiner et al., 2024] examine how different representations of training examples, augmented with LLM-generated explanations, can improve entity matching performance. The experiments of Steiner et al. are based on the LLM generated structured explanations presented in this chapter.

10.3 Explanations for Model Decisions Using Mojito

The method presented in this section investigates the global importance of words belonging to domain-specific word classes for the matching decisions based on aggregated explanations for single matching decisions of matching models. This analysis is based on the Mojito [Di Cicco et al., 2019] framework, which adapts the LIME algorithm [Ribeiro et al., 2016] for the use case of pairwise entity matching. The code for replicating these experiments is available on GitHub[1].

10.3.1 Generating Instance Explanations

LIME [Ribeiro et al., 2016] creates an explanation for a single matching decision as follows: The instance to be explained (pair of entity descriptions) is perturbed using word dropping, and labels for all perturbed instances are queried from the matching model. This set of instance/label pairs is then used to train a surrogate linear regression model, which is assumed to approximate the original model locally for that specific matching decision. The regression model coefficients are extracted and signify the importance of the respective words for the matching decision. The Mojito framework [Di Cicco et al., 2019] offers a second method for

[1] https://github.com/wbsg-uni-mannheim/jointbert

perturbing instances, which is the copying of tokens between entities. For the following demonstration of the presented method, only the default LIME word dropping method is applied for perturbation. Figure 10.1 shows an example of a LIME explanation generated with Mojito for a matching decision by a BERT model for a pair from the MWPD test set of the WDC LSPM benchmark (see Section 4.6.2). Words with orange coloring signify positive weight (pushing toward match), and blue words indicate negative weight (pushing towards non-match). The model's decision for non-match in this specific case seems to be based on the difference in RAM size and SSD size between the two Chromebooks, which is intuitively understandable for a human acquainted with the domain.

```
HP Chromebook 14 G4 - 14 Celeron N2840 2 GB RAM 16 SSD US OETC Consortium Store
HP Chromebook 14 G4 - 14 Celeron N2940 4 GB RAM 32 SSD US OETC Consortium Store
```

L0|hp L0|chromebook L0|14 L0|g4 L0|- L0|14
L0|celeron L0|n2840 L0|2 L0|gb L0|ram L0|16 L0|ssd
L0|us L0|oetc L0|consortium L0|store L1|hp R0|hp
R0|chromebook R0|14 R0|g4 R0|- R0|14 R0|celeron
R0|n2940 R0|4 R0|gb R0|ram R0|32 R0|ssd R0|us
R0|oetc R0|consortium R0|store R1|hp

Figure 10.1: LIME explanation example for a non-match classified correctly by the BERT model.

For these experiments, the MWPD test set of the WDC LSPM benchmark serves as a basis. It contains a variety of hard pairs and products with and without training examples in the associated training set. This allows for sampling interesting record pairs, whose explanations can give further insight into the model decision process when applying the method presented in this section. For the creation of the explanation evaluation dataset, 50 pairs of product offers, split into 25 positives and 25 negatives, are sampled for each of the following matching situations. This selection allows for a fine-grained evaluation of the generated explanations along these matching situations in the following section:

1. Both products had many training examples (>10)
2. Both products had few training examples (<5)
3. Both products had no training examples
4. Both offers contain a model number
5. At least one offer does not have a model number

In the next step, each of the words and numbers in all selected pairs *title* attributes is labeled using a set of common product-specific word classes that subsequently makes it possible to aggregate the explanations across different products that are described by different words that share the same common word class. For example *iPhone 10* or *Samsung S10* would share the common word class *model name*. In the aggregation step of the next section, these word classes will enable the aggregation of explanations for each instance into global modal insights. Table 10.1 shows an overview of these domain-specific word classes and examples of words that fall into each class.

Table 10.1: Product domain-specific word classes used for manual annotation of the explanation dataset.

Word Class	Examples
Model number	*DUAL-GTX1060-O6G, DT100G3/128GB, ...*
Brand name	*Asus, Apple, Hewlett-Packard, NVIDIA, . . .*
Model name	*Thinkpad 11e, GTX 1070, Core i7-4770k, ...*
Characteristic attribute	*16 GB RAM, 256 GB SSD, 4 GHz, . . .*
Stopword	*with, New, Wholesale, Laptop Outlet Direct, . . .*
Product type	*Hard Disk Drive, Processor, Graphics Card, . . .*
Other descriptive word	*Kit, OEM, High Performance, . . .*

10.3.2 Aggregating Instance Explanations

Mojito explanations are restricted to explaining single matching decisions by locally approximating the model around one specific decision. Mojito's explanations can be aggregated for multiple matching decisions representing a specific matching situation to identify the types of words onto which a model focuses in specific matching situations or challenges. For aggregating explanations and for abstracting from specific strings and numbers to common word classes relevant for the domain, such as *model number*, *brand* and *model name*, a three-step approach is used:

To understand the importance of domain-specific word classes in specific matching situations and for specific matching challenges, the five classes previously presented are used to analyze in how far the importance of specific word classes changes based on the premises of the class. In a first step, Mojito is applied to all 250 selected pairs, resulting in importance values for the matching decision for each token in the string representing each pair of records. In the second step, the generated Mojito word weights for the specific tokens are aggregated by word class for each matching situation separately. Finally, the resulting weight aggregations can be compared according to the matching situation between different models. To demonstrate the global model insights that can be gained by this method, a subset of the annotated pairs and matching situations are analyzed in the following sections: (1) pairs correctly classified by BERT, JointBERT, and Deepmatcher, as well as (2) pairs with and without model numbers.

Correct Matching Decisions

To gain a general overview of the differences between the three models BERT, JointBERT, and Deepmatcher along the selected subsets, explanation plots using pairs that all models correctly predicted for the *match* class are presented in Figure 10.2. The black lines across the swarm columns for each model denote the median value of that column. There are some notable differences visible between the Deepmatcher and BERT-based models. Both BERT-based models put a large fraction of the explanation weights on *model numbers*, which is reasonable as this word class is the strongest indicator of matching or non-matching descriptions for this domain. Compared to the BERT models, Deepmatcher puts less weight on the model numbers. The second highest weighted word class is the *model name* for the BERT-based models, which is the highest weighted for Deepmatcher. The model name is one of the strongest indicators for a matching or non-matching pair as often these are the strings where hard non-matching pairs differ in subtle ways (e.g., *IPhone X* vs. *IPhone Xs*). For matching pairs, on the other hand, it is unlikely that these strings differ apart from possible differences due to e.g. encoding or typographical errors. The differences between the models are minimal for the remaining word classes. The weighting of stopwords is low for all models, showing that all tested models are capable of ignoring irrelevant words.

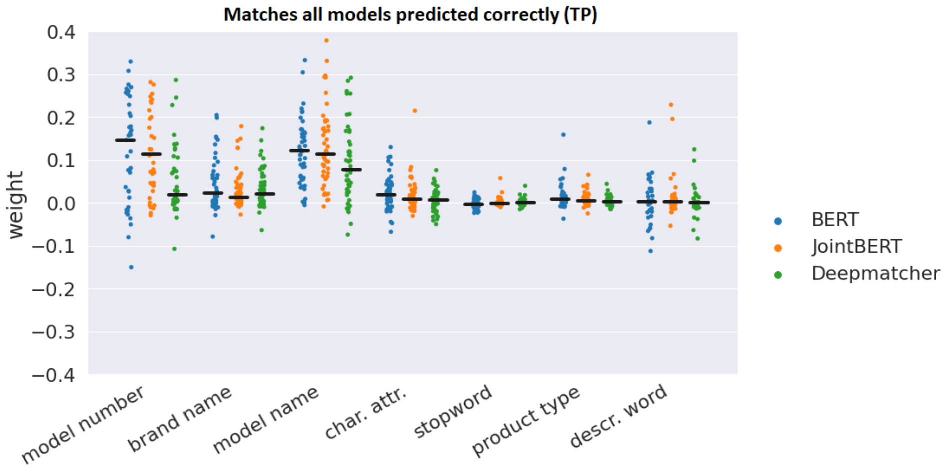


Figure 10.2: Aggregated explanations for pairs classified correctly by all models.

Role of Model Numbers

As model numbers are a strong indicator for identifying matches and non-matches, if they are available, focusing on the two subsets *solvable using model numbers* and *not solvable using only model numbers* allows to illustrate how model decisions change for each of these cases. Figures 10.3 and 10.4 show the resulting explanation plots for aggregating the explanations for these matching situations for matching pairs. For those cases that can be solved using model numbers, that is, where they are available for both offers, the BERT-based models focus strongly on the model numbers and to a lesser extent on the model name compared to the Deepmatcher model which is more focused on the model names and does not weight the model number as strongly in comparison (Figure 10.3).

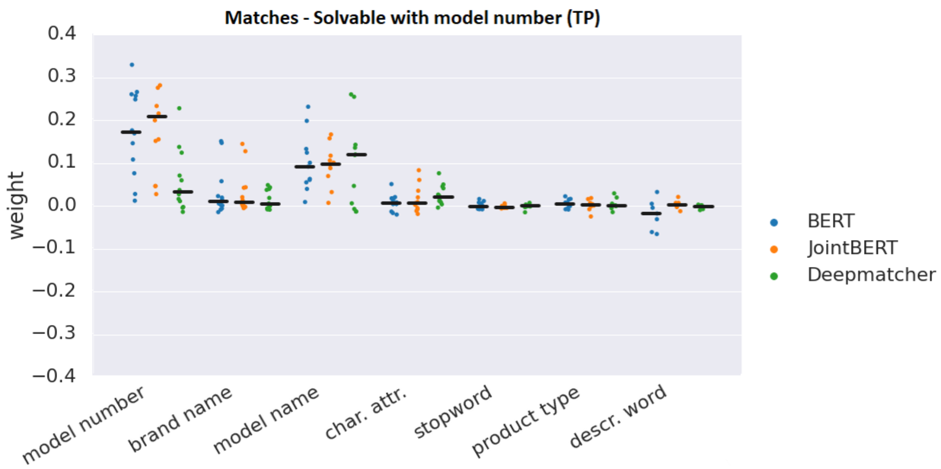


Figure 10.3: Aggregated explanations for the correctly classified pairs of the group "solvable by model number".

When looking at those pairs not solvable using model numbers, the BERT-based models shift their focus to using the model name instead (Figure 10.4). The median weight on the model name is higher than for Deepmatcher for these cases. In conclusion, the BERT-based models seem to be able to shift their focus to strong predictors such as model numbers if available and can adapt if the strongest predictors, i.e. model numbers, are missing when compared to Deepmatcher, which does not adapt based on the available predictors and focuses most on the model name in both matching situations.

Concluding Remarks

The presented method is applicable to any matching classifier that can be explained with the LIME/Mojito explanation methods. Although the annotation of strings into domain-specific word classes has been done manually for the presented experiments, future applications of this method could make use of LLMs to perform this annotation automatically instead. The next section, presenting the second method for aggregating local explanations to global insights, uses LLMs to generate structured explanations, as well as leverages their ability to perform information extraction to generate attributes similar to the word classes presented above, showing their ability for performing this task.

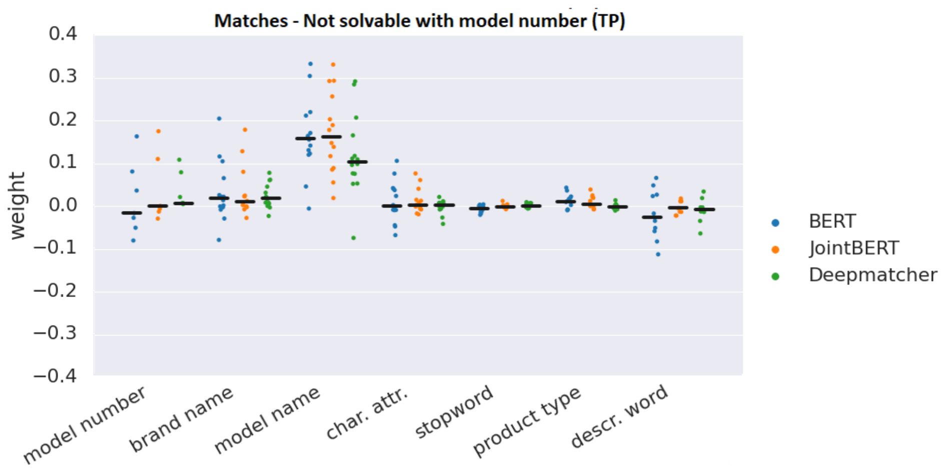


Figure 10.4: Aggregated explanations for the correctly classified pairs of the group "not solvable by model number".

10.4 Explanations for Model Decisions Using an LLM

The previously presented method allows for global insights into the matching system's decisions and is broadly applicable to machine learning models for entity matching. The method relies on the faithfulness of the LIME/Mojito explanations, which are based on approximating the original model locally using linear functions, to the actual decision process inside the original model. The second method presented in the following sections instead uses an LLM, which is queried for explanations of its matching decisions in a structured format. This method does not rely on an external explainability method for the model but instead uses the LLM itself to explain its decision in a structured format. Although large language models are commonly trained to respond using natural language, forcing a structure on the explanations of the model instead allows automatic parsing of the generated explanations. These explanations can then be used for various downstream applications. For example, they allow to gain insights into singular model decisions but can also be used to automatically aggregate explanations in a similar way as in the first method to allow for global insights into the model decisions. Furthermore, structured explanations can be used to enhance fine-tuning data [Wadhwa et al., 2024, Steiner et al., 2024] or to use an LLM to perform the creative process of generating error classes (as presented in Chapter 11) and support error analysis. All of these applications can directly support data engineers in understanding and improving entity matching pipelines. The code for replicating these experiments is available on GitHub[2].

[2] https://github.com/wbsg-uni-mannheim/MatchGPT/tree/main/LLMForEM

10.4.1 Generating Instance Explanations

For the generation of explanations, the LLM is first prompted to decide if a pair of entity descriptions matches (see Chapter 9) and subsequently the conversation is continued by directly asking the model to explain its decision using a second prompt. If no restrictions are imposed on the explanation format, the model would respond with natural language text that describes the different aspects that influenced its decision [Nananukul et al., 2024]. Instead of allowing free-text explanations, the model is instead prompted to organize its explanations into a fixed structure, which will allow automatic parsing and aggregation of the generated explanations.

Figure 10.5 shows examples of complete conversations for generating these structured explanations for pairs from the Walmart-Amazon and DBLP-Scholar datasets. After prompting and receiving a decision in the first exchange with the model, the conversation continues by passing a second prompt (the second user prompt in Figure 10.5). Specifically, a structured explanation format is requested and provided as an example that includes all attributes of both product offers that were used for the matching decision. As the descriptions of the record pairs are highly textual with a limited amount of structure in the original attribute schema, e.g. many important words for the matching decision are contained in the *title* attribute of a product, enforcing a structure to the explanations allows for a clearer comparison of which words were considered relevant to the matching decision.

This structure leads the model to perform information extraction implicitly using its knowledge of the domain. In addition to the extracted attributes, an importance value and a similarity value are also generated for the compared attributes. The sign of the importance values is requested to be negative if the attribute comparison contributed to a non-match decision and vice versa. The importance and similarity values are integral for a human to understand the decision process as presented by the model.

Subsequently, an explanation is queried from the GPT4 model for each pair in the test sets of both datasets for the best performing zero shot prompt for the datasets and the prompts from Chapter 9 using the prompt shown in Figure 10.5. The first response in the form of a structured explanation of the Walmart-Amazon product pair in Figure 10.5 shows that the model can extract various attributes from the serialized strings. The highest positive importance is assigned to the *model* followed by the *brand* and the *price*. Although none of these strings perfectly match, they are highly similar, the model correctly assigns them a high similarity and positive importance value and considers them indications for matching product offers. The model extracted the hard drive size from the first offer, which is missing in the description of the second offer, and assigned this circumstance a low negative importance score.

In the explanation for the DBLP-Scholar pair (lower part of Figure 10.5, the *authors* match perfectly, which the model also sees as important evidence for a match by assigning a positive importance of 0.3. The model further correctly as-

Actor	Prompt (Walmart-Amazon)
User	"Do the two entity descriptions refer to the same real-world entity? Answer with 'Yes' if they do and 'No' if they do not. **Entity 1:** night owl night owl zeus-810 16-channel h. 264 dvr kit with 1tb hard drive 773.82 **Entity 2:** night owl optics security zeus-810 16-channel 8-camera h. 264 dvr surveillance kit with d1 recording and hdmi output zeus-810 676.63"
AI	Yes.
User	Now **explain your decision in a structured format**, listing the attributes that you compared for reaching your decision. Each attribute should be accompanied by the attribute values and a score between -1 and 1 that shows the **importance of the attribute** for the decision. If the attribute influenced the decision towards non-match the importance score should be negative. If the attribute pointed towards a match, the importance score should be positive. Also provide a **similarity score for the attribute values**. If an attribute only occurs in one item, specify the value of that attribute for the other item as "missing". An example output is the following: attribute=brand - importance=0.05 - values=Logitech###Logitech - similarity=1.00 attribute=model - importance=-0.95 - values=MX G500###MX Master 3S - similarity=0.20 attribute=color - importance=0.00 - values=missing###Graphite - similarity=0.00'
AI	**attribute=brand** - importance=0.30 - values=Night Owl###Night Owl Optics - similarity=0.90 **attribute=model** - importance=0.40 - values=Zeus-810 16-channel H. 264 DVR kit###Zeus-810 16-channel 8-camera H. 264 DVR surveillance kit - similarity=0.85 **attribute=price** - importance=0.20 - values=773.82###676.63 - similarity=0.80 **attribute=hard drive** - importance=-0.10 - values=1TB###missing - similarity=0.00

Actor	Prompt (DBLP-Scholar)
User	"Do the two entity descriptions refer to the same real-world entity? Answer with 'Yes' if they do and 'No' if they do not. **Entity 1:** b george , j haritsa secure bufering in firm real-time database systems vldb j. 2000 **Entity 2:** b george , j haritsa secure transaction processing in firm real-time database systems sigmod conference , 1997.0"
AI	No.
User	Now explain your decision ... (same as in example above)
AI	**attribute=authors** - importance=0.30 - values=b george , j haritsa###b george , j haritsa - similarity=1.00 **attribute=title** - importance=-0.70 - values=secure bufering in firm real-time database systems###secure transaction processing in firm real-time database systems - similarity=0.60 **attribute=year** - importance=-0.50 - values=2000###1997.0 - similarity=0.00 **attribute=conference** - importance=-0.50 - values=vldb j.###sigmod conference - similarity=0.00

Figure 10.5: Conversation instructing the model to match an entity pair and subsequently asking for a structured explanation of the matching decision. Top: Walmart-Amazon, bottom: DBLP-Scholar.

Table 10.2: Aggregated structured explanations for Non-Matches and Matches on the DBLP-Scholar and Walmart-Amazon datasets.

	Matches			Non-Matches		
Attribute	Freq.	Mean Import.	St.Dev.	Freq.	Mean Import.	St.Dev.
DBLP-Scholar						
title	0.96	**0.59**	0.40	0.95	-0.40	0.38
authors	0.78	**0.65**	0.40	0.68	**-0.66**	0.34
conference	0.50	0.35	0.37	0.29	-0.11	0.29
year	0.46	0.26	0.37	0.43	-0.16	0.25
journal	0.14	0.40	0.43	0.05	-0.15	0.25
Walmart-Amazon						
brand	0.98	**0.78**	0.34	0.99	-0.04	0.34
price	0.92	-0.03	0.27	0.86	-0.16	0.25
model	0.81	**0.63**	0.51	0.82	**-0.77**	0.37
color	0.24	0.23	0.31	0.35	-0.06	0.23
product type	0.12	**0.64**	0.48	0.11	-0.42	0.50

signs a high negative importance to *year* and *conference*, which are reasonably different to support a non-match decision. Although the *title* overlaps in all but two words, the model still uses this as the most important evidence for predicting non-match. A sample of the generated explanations was manually verified against the corresponding model decisions by a human, confirming the connection between the explanations and the model decisions.

To evaluate the meaningfulness of the similarity values created by the model in the structured explanations, their Pearson correlation is calculated with the string similarity metrics Cosine and Generalized Jaccard. For the test sets of both datasets, the metrics are applied to each extracted attribute found in each explanation, and the correlation between them and the generated similarities is calculated. The similarities generated by the model have a strong positive correlation between 0.75-0.85 and 0.73-0.83 with Cosine and Generalized Jaccard, respectively, across all datasets. These results show that the model-generated similarity values have meaning, as they have a high correlation with standard string similarity metrics.

10.4.2 Aggregating Instance Explanations

After generating the local explanations for each pair of records in the test sets, the next step is to aggregate the local explanations into global insights. Structured explanations can be parsed to automatically extract attributes, importance scores, and similarity values. The extracted values can then be aggregated by attribute and the average importance scores for all attributes deemed relevant by the model for its decision can be calculated. An example of five of these aggregated average importances can be seen in Table 10.2 for both datasets. These five attributes were selected by a domain expert as they represent the most relevant attributes used to distinguish entities in the respective domains. The model frequently assigned high importance to the brand and model for matches, while the price was not considered relevant to the decisions on average. For non-matches, the model focuses on the model attribute with a more neutral importance on the brand attribute. For DBLP-Scholar, the model's focus is on differences and similarities of the title and author attributes of the publications for both matches and non-matches, while conference and year information only contribute to a lesser extent to the matching decisions.

10.5 Conclusion

This chapter contributed two methods for aggregating local explanations into global model insights to the field of entity matching. The first method is based on the aggregation of local explanations created by the Mojito framework [Di Cicco et al., 2019] into global insights into the model decisions by aggregating the feature importances of a set of manually annotated domain-specific word classes. The method was applied to the Deepmatcher [Mudgal et al., 2018], BERT [Devlin et al., 2019] and JointBERT entity matching methods to understand and differentiate how these systems perform entity matching. The results showed that Transformer-based methods are more capable of focusing on the most relevant predictors like model numbers if available, while falling back to other predictors such as the model name and brand if the model number is not listed in both records of a pair. The Deepmatcher system, on the other hand, showed a more even distribution of the importances attributed to these word classes. As this method is model-agnostic, it is applicable to any entity matching method for which explanations can be generated with LIME/Mojito. Furthermore, as an avenue for future work, LLMs could be used to automate the process of annotating word classes, allowing for easier scaling of the method by removing the manual process of annotation.

The second method leveraged the generative nature of an LLM to provide explanations for its own decisions in a structured format, including similarity and importance scores for the relevant attribute comparisons instead of free natural language text. This structured format allows to parse the explanations automatically. Due to the LLM inherent attribute extraction occurring as part of the explanation generation, the extracted attributes and their importances can be further aggregated to generate global insights into what features the LLM bases its decision on. This method is easily scalable to larger amounts of record pairs compared to the first presented method, as no annotation of domain-specific word classes or attributes is necessary. The usefulness of the model-generated similarity scores is supported by the high correlation of the model-generated similarity scores with string similarity metrics like Generalized Jaccard.

Both methods allow more global insight into decisions of neural matching methods compared to existing methods. The next chapter presents how structured explanations can be used to automate the error analysis process in entity matching.

Chapter 11

Automatic Error Analysis for Entity Matching

11.1 Introduction

Error analysis in entity matching involves manually inspecting errors made by matching systems and deriving a set of error classes to categorize them. These error classes are created along the types of error, false positives and false negatives, and are used as a classification for why certain errors occur across record pairs. This information is useful to decide on improvements in the entity matching pipeline, e.g. by changing data pre-processing or parameters of model training. Examples for such error classes could be, for the product domain, "model ignores small differences in model number leading to wrong classification as match" or, for the publication domain, "model is confused by noise in the year attribute leading to missing actual matches although all other attributes match". More examples of error classes will be presented later in Tables 11.1 and 11.2.

Deriving these error classes is a creative task that requires good reasoning capabilities. Although there are tools to facilitate manual inspection of errors[1], the task of generating error classes was not previously automated. This chapter contributes a method for automating error analysis to the field of entity matching. Specifically, the method demonstrates that an LLM can perform this creative task and derive meaningful error classes from the errors and associated structured explanations created before in Section 10.4. A qualitative analysis shows that LLM-generated error classes can be helpful in understanding why a model makes wrong decisions. Furthermore, an analysis of the ability of the LLM to sort errors into the created error classes shows that the LLM is capable of performing this task with a mean accuracy of 80%. This information can then subsequently be used to improve the matching system. The code for replicating these experiments is available on GitHub[2].

[1]https://github.com/olehmberg/winter

[2]https://github.com/wbsg-uni-mannheim/MatchGPT/tree/main/LLMForEM

In summary, the contribution of this chapter is:

- **Method for Automating Error Analysis with LLMs:** A method that uses an LLM to automatically identify potential causes of matching errors by analyzing explanations of wrong decisions and generating error classes, which was a manual creative task performed by humans beforehand. The LLM is also used to automatically sort errors into the generated error classes, reaching a mean accuracy of 80%, which can support data engineers in debugging and improving matching systems.

Section 11.2 presents an overview of related work for explaining model decisions in the field of entity matching. Section 11.3 presents the method for automatically generating error classes with an LLM. Section 11.4 evaluates the model's accuracy for assigning errors to the generated error classes. Finally, Section 11.5 summarizes the contributions and findings of this chapter.

The work presented in this chapter has been published in [Peeters et al., 2025].

11.2 Related Work

The analysis of errors made by entity matching systems is crucial for data engineers trying to improve entity matching pipelines. This process usually involves a large amount of manual work as it requires looking at large amounts of erroneous decisions made by a matching model and subsequently deriving steps to reduce or eliminate possible error sources. This *error reporting* process can be supported by tools like WInte.r[3] that automatically collects erroneous matching decisions and presents them to the user in a format that allows to view created matching rules and corresponding similarity values. Using this view, users can create error classes based on an analysis of the presented information and subsequently sort the errors made into these classes. Once this process is completed, the user has a clear overview of what kind of errors happen and how often they occur. Unfortunately, WInte.r does not support error analysis for deep neural matching methods.

Data engineers can be further supported by using explainability methods like those presented in Chapter 10 to obtain local or global explanations for model decisions [Ebaid et al., 2019, Di Cicco et al., 2019, Baraldi et al., 2023, Paganelli et al., 2022]. While these methods provide explanations, the task of error analysis still remains mostly manual as the engineers have to create a set of error classes that represent all of the single erroneous decisions in an aggregated form, which requires creative abilities that the mentioned supportive systems do not possess.

Lu et al. [Lu et al., 2023a] take a step toward automated error analysis by augmenting a model-based evaluation score [Sai et al., 2022] to automatically classify

[3]https://github.com/olehmberg/winter

Table 11.1: GPT4-turbo generated error classes for the DBLP-Scholar dataset and manually annotated number of errors falling into each class.

False Negatives (26 overall)	# errors
1. Year Discrepancy: Differences in publication years lead to false negatives, even when other attributes match closely.	8
2. Venue Variability: Variations in how the publication venue is listed (e.g., abbreviations, full names) cause mismatches.	14
3. Author Name Variations: Differences in author names, including initials, order of names, or inclusion of middle names, lead to false negatives.	9
4. Title Variations: Minor differences in titles, such as missing words or different word order, can cause false negatives.	11
5. Author List Incompleteness: Differences in the completeness of the author list, where one entry has more authors listed than the other.	11
False Positives (26 overall)	# errors
1. Overemphasis on Title Similarity: High similarity in titles leading to false positives, despite differences in other critical attributes.	15
2. Author Name Similarity Overreach: False positives due to high similarity in author names, ignoring discrepancies in other attributes.	16
3. Year and Venue Ignored: Cases where the year and venue match or are close, but other discrepancies are overlooked.	5
4. Partial Information Match: Matching based on partial information, such as incomplete author lists or titles, leading to false positives.	19
5. Misinterpretation of Publication Types: Confusing different types of publications (e.g., conference vs. journal) when other attributes match.	9

translation errors into a set of given error classes and subsequently refine the translation based on the errors found. In follow-up work, Lu et al. [Lu et al., 2023b] proposed error analysis prompting and showed that GPT4 can produce a human-like evaluation of machine translations. For this, they have the model break down the translations of a sequence into smaller sequences, assign machine-generated error classes to the translated text and subsequently score the translations based on the found errors. The authors conclude that this process significantly improves the model-based scoring of LLM translations. A direction of research related to error analysis is dedicated to the self-improvement of LLMs [Pan et al., 2024] aimed at addressing their inherent drawbacks such as hallucination, unfaithful reasoning, and the generation of toxic content.

11.3 Discovery of Error Classes

For automatically discovering error classes with the presented method, all erroneous decisions, together with the structured explanations of these decisions, must be collected. For this experiment, this is done for the zero-shot prompts and explanations used in Section 10.4. In the second step, a prompt is passed to the OpenAI GPT4-turbo model, which asks for the synthesis of error classes for false positive and false negative cases separately. Subsequently, the collected erroneous pairs together with their GPT4 created explanations are listed in the prompt. These are 26 false positives and 26 false negatives for the DBLP-Scholar test set and 26 false

Table 11.2: GPT4-turbo generated error classes for the Walmart-Amazon dataset and manually annotated number of errors falling into each class.

False Negatives (15 overall)	# errors
1. Model Number Mismatch: The system fails when there are slight differences in model numbers or product codes, even when other attributes match closely.	9
2. Attribute Missing or Incomplete: When one product listing includes an attribute that the other does not, the system may fail to recognize them as a match.	9
3. Minor Differences in Descriptions: Small differences in product descriptions or titles can lead to false negatives, such as slightly different wording or the inclusion/exclusion of certain features.	11
4. Price Differences: Even when products are very similar, significant price differences can lead to false negatives, as the system might weigh price too heavily.	12
5. Variant or Accessory Differences: Differences in product variants or accessories included can cause false negatives, especially if the system does not adequately account for these variations being minor.	7
False Positives (26 overall)	# errors
1. Overemphasis on Matching Attributes: The system might give too much weight to matching attributes like brand or model number, leading to false positives even when other important attributes differ.	23
2. Ignoring Minor but Significant Differences: The system fails to recognize important differences in product types, models, or features that aresignificant to the product identity.	21
3. Misinterpretation of Accessory or Variant Information: Including or excluding accessories or variants in the product description can lead to false positives if the system does not correctly interpret these differences.	8
4. Price Discrepancy Overlooked: The system might overlook significant price differences, assuming products are the same when they are not, particularly if other attributes match closely.	14
5. Condition or Quality Differences: Differences in the condition or quality of products (e.g., original vs. compatible, new vs. refurbished) are not adequately accounted for, leading to false positives.	2

<table>
<tr><th>Actor</th><th>Prompt</th></tr>
<tr><td>User</td><td>The following list contains false positive and false negative product pairs from the output of a product matching classification system. Given the product pairs and the associated explanations, come up with a set of error classes, separately for both false positives and false negatives, that explain why the classification systems fails on these examples.

False Negatives:

FN1
{Entity 1}
{Entity 2}
{Explanation}

{additional FNs}

False Positives:

FP1
{Entity 1}
{Entity 2}
{Explanation}

{additional FPs}</td></tr>
<tr><td>AI</td><td>Based on the provided examples of false negatives (FN) and false positives (FP), we can identify several error classes that explain why the classification system fails on these examples.

Error Classes for False Negatives:
1. Model Number Mismatch: The system fails when there are slight differences in model numbers or product codes, even when other attributes match closely.
2. {additional error classes}

Error Classes for False Positives:
1. Overemphasis on Matching Attributes: The system might give too much weight to matching attributes like brand or model number, leading to false positives even when other important attributes differ.
2. {additional error classes}</td></tr>
</table>

Figure 11.1: Prompt used for the automatic generation of error classes given false positives and false negatives.

positives and 15 false negatives for the Walmart-Amazon test set from Section 9.3. An example of this prompt and part of the answer of the LLM is presented in Figure 11.1.

Tables 11.1 and 11.2 show the resulting generated error classes for both datasets and, for each class, the number of errors that fall into these classes. The latter were manually annotated by three domain experts.

The summary numbers from the manual annotations show that the automatically created error classes are relevant and cover frequent and rarer errors. For example, for the DBLP-Scholar dataset, the first error class of false positives references putting too much emphasis on the similarity of publication titles, which is deemed correct by a human annotator for 15 of the 26 errors, while the third error class is relevant only for 5 of the errors, namely those where the model seemed to put too much emphasis on matching year and venue information in the pairs while ignoring crucial difference in the other attributes.

Actor	Prompt
User	Given the following error classes for a product matching classification system, please **classify the following product pair into all error classes** by their number **if they are relevant for this pair** and its explanation. Please give a short explanation of every decision as a list first. Finally, also provide a confidence score for each classification adhering to the JSON format of the following example: {"2":"90","4":"30","5":"75"} Error classes: 1. Model Number Mismatch: The system fails when there are slight differences in model numbers or product codes, even when other attributes match closely (e.g., FN2, FN7, FN8). 2. {remaining error classes} Now classify this pair: Original Label: Match Predicted Label: Non-Match {Entity 1} {Entity 2} {Explanation}
AI	Based on the provided information and the error classes, here's the classification for the product pair: 1. Model Number Mismatch: This error class is relevant because ... 2. {additional classifications} { "1": "95", "2": "60", "3": "50", "4": "70"}

Figure 11.2: Prompt used for the classification of errors into the generated error classes.

11.4 Assignment of Errors to Error Classes

This final experiment investigates whether GPT4-turbo can explicitly categorize errors into the created error classes it has generated. Such a categorization allows data engineers to drill down from the error classes to concrete example errors, which might give them hints on how to address the problem and improve the preprocessing or the matching system itself. The prompt shown in Figure 11.2 is used to prompt the LLM to categorize errors. Each prompt presents the error classes and, subsequently, the correct and predicted labels, and, finally, one pair of entity descriptions with its GPT4-created explanation. The model is asked to pick all error classes that apply to the pair and to provide a confidence value for each prediction.

Table 11.3 shows the accuracy values the GPT4-turbo model reaches on this task. On average, the model achieves a mean accuracy of more than 80% for most types of errors. Only the mean accuracy on Walmart-Amazons false positives is lower which is caused by the low accuracy of the first error class *Overemphasis on Matching Attributes* as the domain experts disagreed with the models classification in the first error class, more specifically the model rarely assigned this class while

Table 11.3: Accuracy of GPT4 for classifying errors into the created error classes.

	Walmart-Amazon		DBLP-Scholar	
Error class	FP	FN	FP	FN
1	34.62	86.67	92.31	96.15
2	84.62	73.33	76.92	92.31
3	84.62	73.33	76.92	73.08
4	76.92	100	100	88.46
5	84.62	86.67	92.31	88.46
Mean	73.08	84.00	87.69	87.69

the domain experts considered it relevant in 23 out of 26 cases. Apart from this disagreement, the model can accurately categorize errors, which shows the usefulness of LLMs for automating error analysis and supporting data engineers.

11.5 Conclusion

This chapter introduced a method for automatically generating error classes from matching errors using LLMs. The LLM GPT4-turbo was shown to be capable of performing the creative task of automatically deriving meaningful error classes from a set of matching errors in combination with their LLM-created structured explanations for the product and bibliographic matching domains. This automation of error analysis and the resulting error classes can support data engineers by pointing them to issues that they may otherwise have overlooked. The presented method is the first to automate this part of the error analysis process for entity matching.

Chapter 12

Conclusion

This chapter summarizes the contributions of this thesis to the field of entity matching, outlines the research impact, and discusses open issues and future directions. Section 12.1 first summarizes each part of this thesis and their contributions. Section 12.2 discusses the research impact of the thesis. Finally, Section 12.3 discusses open issues and future research directions that remain or were discovered as part of the work on this thesis.

12.1 Summary and Contributions

The following sections summarize the findings of each part of the thesis separately and highlight the contributions made to the field of entity matching. The structure of the following sections follows the same order as the structure of the thesis.

Part II: The Semantic Web as a Source of Training Data for Product Matching

This part of the thesis focused on the usefulness of using product identifiers annotated with Schema.org annotations in the Common Crawl as distant supervision by deriving labeled pairs for the product matching task. The research spanned the subsequent creation and evaluation of two large-scale benchmarks for product matching.

Chapter 4 first presented a process for clustering product offers using product identifiers such as GTINs and MPNs found in the Schema.org annotations in the Common Crawl. This process ensures high-quality clustering through various cleansing steps, resulting in the WDC Training Dataset for Large-Scale Product Matching. This dataset contains 26 million product offers from 79 thousand websites, grouped into 16 million clusters. Profiling this corpus demonstrated the potential to generate and automatically label a vast number of matching and non-matching pairs, making it the largest publicly available source of labeled data for product matching at its time of creation. The diversity and scale of this dataset exceeds previous benchmarks.

Building on this foundation, the first contribution of this part, the WDC LSPM benchmark was introduced, offering comprehensive training, validation, and testing splits for product categories such as *computers*, *cameras*, *watches*, and *shoes*. The benchmark's varying training set sizes, ranging from 2,000 to 65,000 labeled pairs, provide a robust resource for evaluating multi-source entity matching systems. The largest combined set contains over 210,000 pairs, making it the most extensive and diverse benchmark of its kind at the time of creation. This benchmark also includes a specialized test set for the *computers* category that contains a set of matching challenges, facilitating more fine-grained evaluation.

Further experiments demonstrated the effectiveness of using the WDC LSPM benchmark and additional training pairs for training both neural and non-neural product matchers. In particular, the Deepmatcher [Mudgal et al., 2018] RNN model achieved a performance of more than 90% F1, further improving with end-to-end training of the fastText embeddings. Comparisons with non-neural baselines confirmed the advantages of deep neural networks in handling highly textual data. Continued fine-tuning using relevant pairs from the dataset maintained high performance also for new products, highlighting the utility of Schema.org annotated product identifiers on the Web as distant supervision for ongoing model improvement.

The research also explored the impact of label noise on model performance, revealing that while a 6% noise level does not hinder high-performance training, higher noise levels significantly degrade performance. Furthermore, experiments with the BERT [Devlin et al., 2019] transformer model underscored the value of large amounts of diverse training pairs sourced from Schema.org annotations on the Web, showing notable performance improvements when combining masked language modeling with the product matching task. This led to an F1 performance increase of more than 97% for the *computers* category of the WDC LSPM benchmark (see Section 4.7.5).

Chapter 5 presented the second contribution, the WDC Products benchmark. This benchmark builds on the work from Chapter 4 by providing a multi-dimensional benchmark created using Schema.org annotated product identifiers as distant supervision. This benchmark, derived from the December 2020 Common Crawl, includes 27 variants to evaluate systems in the dimensions (1) amount of corner cases, (2) amount of unseen entities in the test set, and (3) development set size. It uniquely supports both pair-wise and multi-class matching tasks, offering the first fine-grained evaluation framework along these three dimensions for entity matching.

In conclusion, the WDC LSPM and WDC Products benchmarks provide new resources for the research community, complementing the existing benchmarks with larger and more diverse benchmarks, as well as fine-grained evaluation along various matching challenges, supporting the development and assessment of more accurate and robust product matching systems.

Part III: Deep Neural Networks for Entity Matching

This part of the thesis has made several contributions to the field of supervised entity matching, particularly through the exploration and enhancement of Transformer-based models using the benchmarks presented in Part II in combination with existing entity matching benchmarks. The work has addressed challenges in cross-lingual entity matching, as well as proposed the dual-objective learning method JointBERT, the supervised contrastive learning method R-SupCon, and an extensive evaluation of prompting techniques and fine-tuning for generative large language models.

Chapter 6 contributed the first evaluation of cross-lingual fine-tuning using mono- and multi-lingual Transformers for product matching. It showed that incorporating English-language offer pairs into the training set significantly improves performance for a low-resource language, which was German in these experiments. Multilingual BERT, pre-trained on large amounts of text in multiple languages, achieved higher performance than mono-lingual models. This approach also highlighted the potential to reduce labeling costs by leveraging automatically created training pairs from high-resource languages such as English by using Schema.org annotations as distant supervision, thus contributing to more efficient data integration processes.

Chapter 7 contributed JointBERT, a dual-objective fine-tuning method for Transformers. By leveraging both binary matching labels and entity group information as part of its dual-objective training approach, JointBERT outperformed state-of-the-art models like Ditto by 1-5% in F1 score for seen entities on the benchmarks WDC LSPM and DI2KG-monitor, provided multiple entity descriptions were available for each entity. This method is particularly applicable in scenarios where available entity identifiers in entity descriptions facilitate the discovery of entity groups, such as product identifiers in the product domain. However, JointBERT showed reduced performance for unseen entities, indicating that it is best used in contexts with available training data for each entity.

Chapter 8 contributed R-SupCon, a method for supervised contrastive learning that improved matching performance on multiple benchmarks by 0.8-8.8% in F1 score setting a new state-of-the-art for the Abt-Buy, Amazon-Google and WDC LSPM computers benchmarks at the time of contribution. This method uses a supervised contrastive pre-training stage that generates a large amount of record pairings inside the training batches using entity group information similar to JointBERT. This is followed by a cross-entropy fine-tuning stage for classification of record pairs. R-SupCon is especially effective with smaller training sets. Experiments with benchmarks containing duplicates in each source further showed the importance of the proposed source-aware sampling strategy to avoid introducing label noise. Comparisons with other contrastive methods like DAEM [Bai et al., 2023] and Sudowoodoo [Wang et al., 2023] underlined the performance of R-SupCon, particularly due to its effective use of entity group information and source-aware sampling.

Chapter 9 contributed a comparison of the effectiveness of different prompting techniques and fine-tuning of generative LLMs and compared the performance of these models with PLMs like Ditto [Li et al., 2020]. The experiments showed that the best prompt needs to be estimated for each model/dataset combination. The most powerful LLMs offer robust performance across different prompts and datasets while requiring no or only a few training examples in the form of few-shot examples or provided matching rules derived from examples. The evaluation provided insights into the trade-offs between LLMs and PLMs, particularly in terms of zero-shot performance, training data requirements, and computational costs. The chapter concluded that LLM-based matchers are preferable for scenarios involving unseen entities and costly training data acquisition, whereas PLM-based matchers are suitable for tasks with sufficient training data and low availability of computational resources.

Part IV: Explainability and Error Analysis for Entity Matching

The research presented in this part of the thesis focused on explainability of neural entity matching models decisions and the potential of automating error analysis with LLMs.

Chapter 10 contributed two novel methods for aggregating local explanations to derive global insights into model decisions. The first method, leveraging the Mojito framework, aggregates feature importances of annotated domain-specific word classes to generate global explanations. Applied to the Deepmatcher, BERT, and JointBERT entity matching methods, this approach revealed how these systems prioritize different predictors in different situations, highlighting the adapting focus of Transformer-based methods on relevant features like model numbers, compared to the more constant importance scores of the Deepmatcher system. Notably, this model-agnostic method is versatile and can be used together with any entity matching system for which the LIME/Mojito methods can be applied.

The second method uses the generative capabilities of an LLM to create structured explanations of its own decisions that include model-generated similarity and importance scores for relevant attribute comparisons. This structured format enables automatic parsing and aggregation of attributes and their importances, facilitating the generation of global insights into the features influencing LLM decisions. Finally, the model-generated similarity scores demonstrated high correlation with established string similarity metrics like Generalized Jaccard providing increased trust into the provided explanations.

Chapter 11 presented a novel method for automating the generation of error classes from matching errors using LLMs. This chapter demonstrated the LLM's capability to creatively derive meaningful error classes from sets of matching errors and their structured explanations in both product and bibliographic domains. The automated error analysis process can support data engineers in identifying and addressing issues, presenting the first approach to automating error classification in entity matching.

12.2 Research Impact

This section discusses the impact of the benchmarks and methods presented in this thesis on other research efforts. Especially the WDC LSPM benchmark has been adopted in the evaluation of entity matching methods in the community and has been part of the evaluation of pioneering work like the Ditto [Li et al., 2020] matching system. The following paragraphs are not an exhaustive list but list some of the higher-profile publications in terms of citations and very recent work that has used the benchmarks and methods presented in this thesis.

The WDC LSPM benchmark was released in 2019 and has been used in the evaluation of the pioneering Ditto [Li et al., 2020, Li et al., 2021] matching system, which was one of the first entity matching systems using PLMs. The benchmark was also used to evaluate the DADER system for domain adaptation [Tu et al., 2022], the HierGAT system using hierarchical graph attention [Yao et al., 2022] and the FlexER system [Genossar et al., 2023]. The benchmark continues to be used also in very recent work in entity matching [Sun et al., 2024, Alves et al., 2024, Zhang et al., 2024b].

The WDC Products benchmark was released at the end of 2022 and has been used so far for the evaluation of the entity matching systems APrompt4EM [Xia et al., 2024] , GraLMatch [de Meer Pardo et al., 2025] and SETEM [Ding et al., 2024]. WDC Products has also been used for evaluating blocking methods as part of the SC-Block [Brinkmann et al., 2024] system.

The work on cross-lingual matching presented in Chapter 6 has been cited as inspiration in similar works [Możdżonek et al., 2022], where especially Alves et al. [Alves et al., 2024] perform a broader evaluation of multi-lingual Transformers for the transfer to the Portuguese language using additional English training data. This work also uses the WDC LSPM benchmark for this purpose.

The JointBERT method presented in Chapter 7 has been cited in the following recent works on entity matching [Paganelli et al., 2022, Zeakis et al., 2023, Li et al., 2023b, Wang et al., 2023, Genossar et al., 2023]. JointBERT has directly influenced the works of Ye et al. [Ye et al., 2022] and Zhang et al. [Zhang et al., 2024b] who proposed JointMatcher and EMBA respectively. Both systems are built directly on the dual-objective training method introduced by JointBERT.

The R-SupCon method presented in Chapter 8 has been cited in the following recent publications [Almagro et al., 2023, Paganelli et al., 2023, Xia et al., 2024, de Meer Pardo et al., 2025, Huang et al., 2024]. Paganelli et al. [Paganelli et al., 2023] use R-SupCon as part of their experimental evaluation of various entity matching models and conclude that it is the only method in their experiments that provides embeddings whose similarity directly relates to matches or non-matches compared to the other investigated methods such as Ditto [Li et al., 2020]. Almagro et al. [Almagro et al., 2022] directly build on R-SupCon and introduce an additional hard negative mining method to the batch building process showing that this further improves performance over default R-SupCon. In the area of blocking for entity matching, Brinkmann et al. [Brinkmann et al., 2024] use the pre-training stage of R-SupCon as part of a blocking scheme in combination with nearest neighbor search that runs 1.5-4 times faster than previous blocking methods on various datasets.

The work on the evaluation of prompting techniques and fine-tuning for LLMs presented in Chapter 9 has been cited by several research works on entity matching that are concurrently investigating LLMs [Remadi et al., 2024, Fan et al., 2024b, Wadhwa et al., 2024, Wang et al., 2024, Huang and Zhao, 2024].

12.3 Open Issues and Future Research Directions

This section discusses open issues regarding the topics presented in this thesis and directions for future work in the area of benchmarks and methods for entity matching.

Although both benchmarks presented in this thesis improve the field of benchmarking entity matching systems with their size, diversity, and fine-grained evaluation, some open issues remain. The matching challenges in WDC Products for example are not exhaustive. Additional challenges may include cross-language matching, for which no large publicly available benchmark covering multiple languages exists to the best of the authors' knowledge, or varying structuredness in the available schemata of each source. Both WDC benchmarks present very textual matching tasks, although multiple modalities may be relevant. For example, product images may have a significant impact on matching performance when comparing textual corner cases like a mobile phone and an accessory for that specific phone where the textual descriptions can be highly similar, but the images may clearly refer to different products. Finally, both benchmarks presented cover only the product domain, while many benchmarks from different domains are required for a comprehensive evaluation of entity matching methods and their transferability. Current publicly available benchmarks for domains like publications, restaurants, and companies face the same shortcomings as previous product matching benchmarks, leading to a research gap and the need for larger, more diverse benchmarks that allow for fine-grained evaluation of matching challenges, which may also be specific to the relevant domain. The continued adoption of Schema.org and the availability of many different entity classes in the vocabulary such as *LocalBusiness*, *Movie*, or *Restaurant* could be leveraged in the future to build more benchmarks using distant supervision similar to the product-based WDC benchmarks to close this gap.

The cross-language learning experiments with multi-lingual Transformers have shown the potential of such models in combination with training examples from a high-resource language like English to improve matching performance for German when the amount of available German training data is scarce. Future work in this area could extend the evaluations to more languages from different cultures with a vastly different language-historical background, such as, for example, Asian languages. Method-wise, additional focus could be directed towards leveraging more sophisticated multilingual embeddings, large language models and transfer learning techniques that have been proven to work well from the area of natural language processing to enhance the performance of entity matching systems across diverse languages and dialects. For example, LLMs could be combined with a retrieval-augmented generation approach to select in-context examples from other languages to improve matching performance in low-resource languages. Furthermore, addressing the challenges of cultural and contextual variations in multilingual data could further improve the robustness of these models on other entity matching domains.

The advances in deep learning, especially the Transformer-based language models, have led to most recent entity matching methods adopting these models which has led to new state-of-the-art results on most publicly available entity matching benchmarks. During the experiments with PLM- and LLM-based models in this thesis, a set of unsolved challenges has crystallized that are avenues for future work on entity matching methods. As shown in this thesis, PLM-based methods struggle with unseen entities, which is a highly relevant challenge for real-world use-cases. While these methods achieve higher matching performance levels compared to non-neural methods on such entities, there is still a significant loss of performance visible in such situations or when applying models in other domains. Due to the high zero-shot performance of LLMs for entity matching, matching methods built on these models could be better suited for handling this case, although the application of LLMs at scale in practice can prove challenging due to the high costs of LLM-based methods.

Future work may focus on building an entity matching system that solves the disadvantages of each method while preserving the advantages. The presented prompting experiments showed that LLMs have the potential to solve two of these shortcomings by requiring only a few training examples to match or exceed the performance of PLMs while performing better for unseen entities, even after domain-specific fine-tuning. These advantages come with a higher cost regarding infrastructure and operational costs, either locally or as a hosted solution. A system that combines both LLMs and PLMs in a way that each type of model can take advantage of their strengths could be a potential next step in neural entity matching methods. Especially, the general connections between different dimensions like operational costs, model size, the impact of the training method (pre-training and fine-tuning) on the final matching performance in different settings should be a focus of research. The strong zero-shot performance of LLMs further prompts a need for research into the question in how far the existing benchmark datasets are contaminated due to inclusion in the pre-training data of the models. Regarding training data selection for pre-training and fine-tuning, a research direction of interest may be to investigate what a good training set should cover to achieve high performance on seen and unseen entities while minimizing the amount of required labeled training pairs.

Finally, the explainability of entity matching models is crucial for gaining trust in model decisions. Although this thesis introduced methods for aggregating local explanations into global insights, there is still a need for more intuitive and user-friendly explanation techniques. Future research could investigate the faithfulness [Jacovi and Goldberg, 2020] of LLM generated explanations to the actual model decisions in more depth, as well as the usage of LLMs also for explaining other model types like PLMs. For usability in practical use-cases the development of interactive visualization tools that provide more granular insights into model decisions are a logical next step. Explainability methods could further be a tool to facilitate the research into what a training set for pre-training and fine-tuning should cover to achieve high matching performance in seen and unseen scenarios. Automation of error analysis using LLMs, as demonstrated in this thesis, is another promising direction. However, broader experiments on more entity matching domains are needed to gauge the general accuracy and efficiency of error class generation and error classification for a wide range of use-cases. Future work could focus on integrating more sophisticated machine learning techniques, such as meta-learning and active learning, to refine error class generation and classification processes.

List of Figures

List of Tables

Bibliography

[Adar et al., 2009] Adar, E., Skinner, M., and Weld, D. S. (2009). Information Arbitrage across Multi-lingual Wikipedia. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 94–103.

[Akbarian Rastaghi et al., 2022] Akbarian Rastaghi, M., Kamalloo, E., and Rafiei, D. (2022). Probing the Robustness of Pre-trained Language Models for Entity Matching. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3786–3790.

[Almagro et al., 2023] Almagro, M., Almazán, E., Ortego, D., and Jiménez, D. (2023). LEA: Improving Sentence Similarity Robustness to Typos using Lexical Attention Bias. In *Proceedings of the 29th Conference on Knowledge Discovery and Data Mining*, pages 36–46.

[Almagro et al., 2022] Almagro, M., Jiménez, D., Ortego, D., Almazán, E., and Garcia, E. M. (2022). Block-SCL: Blocking Matters for Supervised Contrastive Learning in Product Matching. In *Proceedings of the 2022 ACM SIGIR Workshop on eCommerce*.

[Alrashed et al., 2021] Alrashed, T., Paparas, D., Benjelloun, O., et al. (2021). Dataset or Not? A Study on the Veracity of Semantic Markup for Dataset Pages. In *Proceedings of the 20th International Semantic Web Conference*, pages 338–356.

[Alves et al., 2024] Alves, A. L. F., Baptista, C. d. S., Barbosa, L., and Araujo, C. B. M. (2024). Cross-Lingual Learning Strategies for Improving Product Matching Quality. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, pages 313–320.

[Arora et al., 2017] Arora, S., Liang, Y., and Ma, T. (2017). A Simple but Tough-to-beat Baseline for Sentence Embeddings. In *Proceedings of the 5th International Conference on Learning Representations*.

[Aslam et al., 2004] Aslam, J., Pelekhov, E., and Rus, D. (2004). The Star Clustering Algorithm for Static and Dynamic Information Organization. *Journal of Graph Algorithms and Applications*, 8(1):95–129.

[Bai et al., 2023] Bai, H., Shen, D., Dou, W., Nie, T., and Kou, Y. (2023). Domain-Generic Pre-Training for Low-Cost Entity Matching via Domain Alignment and Domain Antagonism. In *2023 International Joint Conference on Neural Networks*, pages 1–7.

[Bansal et al., 2004] Bansal, N., Blum, A., and Chawla, S. (2004). Correlation Clustering. *Machine Learning*, 56:89–113.

[Baraldi et al., 2023] Baraldi, A., Buono, F. D., Guerra, F., Paganelli, M., and Vincini, M. (2023). An Intrinsically Interpretable Entity Matching System. In *Proceedings 26th International Conference on Extending Database Technology*, pages 645–657.

[Barlaug, 2023] Barlaug, N. (2023). LEMON: Explainable Entity Matching. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):8171–8184.

[Barlaug and Gulla, 2021] Barlaug, N. and Gulla, J. A. (2021). Neural Networks for Entity Matching: A Survey. *ACM Transactions on Knowledge Discovery from Data*, 15(3):52:1–52:37.

[Beltagy et al., 2019] Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: A Pre-trained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3606–3611.

[Bernstein et al., 2011] Bernstein, P. A., Madhavan, J., and Rahm, E. (2011). Generic Schema Matching, Ten Years Later. *Proceedings of the VLDB Endowment*, 4(11):695–701.

[Bhagavatula et al., 2013] Bhagavatula, C. S., Noraset, T., and Downey, D. (2013). Methods for Exploring and Mining Tables on Wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, pages 18–26.

[Bilenko et al., 2006] Bilenko, M., Kamath, B., and Mooney, R. J. (2006). Adaptive Blocking: Learning to Scale Up Record Linkage. In *Proceedings of the Sixth International Conference on Data Mining*, pages 87–96.

[Bilenko and Mooney, 2003] Bilenko, M. and Mooney, R. J. (2003). Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 39–48.

[Bishop and Nasrabadi, 2006] Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern Recognition and Machine Learning*, volume 4. Springer.

[Bizer et al., 2019] Bizer, C., Primpeli, A., and Peeters, R. (2019). Using the Semantic Web as a Source of Training Data. *Datenbank-Spektrum*, 19(2):127–135.

[Bleiholder and Naumann, 2009] Bleiholder, J. and Naumann, F. (2009). Data Fusion. *ACM Computing Surveys*, 41(1):1–41.

[Bogatu et al., 2021] Bogatu, A., Paton, N. W., Douthwaite, M., Davie, S., and Freitas, A. (2021). Cost–effective Variational Active Entity Resolution. In *Proceedings of the 37th International Conference on Data Engineering*, pages 1272–1283.

[Böhm et al., 2012] Böhm, C., De Melo, G., Naumann, F., and Weikum, G. (2012). LINDA: Distributed Web-of-Data-Scale Entity Matching. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2104–2108.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[Breiman, 2017] Breiman, L. (2017). *Classification and Regression Trees*. Routledge.

[Brinkmann et al., 2023a] Brinkmann, A., Primpeli, A., and Bizer, C. (2023a). The Web Data Commons Schema.org Data Set Series. In *Companion Proceedings of the ACM Web Conference 2023*, pages 136–139.

[Brinkmann et al., 2023b] Brinkmann, A., Shraga, R., and Bizer, C. (2023b). ExtractGPT: Exploring the Potential of Large Language Models for Product Attribute Value Extraction. *arXiv preprint arXiv:2310.12537*.

[Brinkmann et al., 2024] Brinkmann, A., Shraga, R., and Bizer, C. (2024). SC-Block: Supervised Contrastive Blocking within Entity Resolution Pipelines. In *In Proceedings of the 21st European Semantic Web Conference*, pages 121–142.

[Brown et al., 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., et al. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

[Brunner and Stockinger, 2020] Brunner, U. and Stockinger, K. (2020). Entity Matching with Transformer Architectures - a Step Forward in Data Integration. In *Proceedings of the 23rd International Conference on Extending Database Technology*, pages 463–473.

[Cafarella et al., 2018] Cafarella, M., Halevy, A., Lee, H., et al. (2018). Ten Years of Webtables. *Proceedings of the VLDB Endowment*, 11(12):2140–2149.

[Caruana, 1997] Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28:41–75.

[Chapman et al., 2020] Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L.-D., et al. (2020). Dataset Search: A Survey. *The VLDB Journal*, 29(1):251–272.

[Chen et al., 2018] Chen, M., Tian, Y., Chang, K.-W., Skiena, S., and Zaniolo, C. (2018). Co-Training Embeddings of Knowledge Graphs and Entity Descriptions for Cross-Lingual Entity Alignment. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3998–4004.

[Chen et al., 2017] Chen, M., Tian, Y., Yang, M., and Zaniolo, C. (2017). Multilingual Knowledge Graph Embeddings for Cross-Lingual Knowledge Alignment. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1511–1517.

[Chen et al., 2020] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607.

[Chen et al., 2019] Chen, X., Xu, Y., Broneske, D., Durand, G. C., Zoun, R., et al. (2019). Heterogeneous Committee-based Active Learning for Entity Resolution (HeALER). In *Proceedings of the 23rd European Conference on Advances in Databases and Information Systems*, pages 69–85.

[Chierichetti et al., 2014] Chierichetti, F., Dalvi, N., and Kumar, R. (2014). Correlation Clustering in MapReduce. In *Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining*, pages 641–650.

[Chopra et al., 2005] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Proceedings of the 2005 Conference on Computer Vision and Pattern Recognition*, pages 539–546.

[Christen, 2008] Christen, P. (2008). Automatic Record Linkage Using Seeded Nearest Neighbour and Support Vector Machine Classification. In *Proceedings of the 14th Conference on Knowledge Discovery and Data Mining*, pages 151–159.

[Christen, 2012a] Christen, P. (2012a). *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer-Verlag, Berlin Heidelberg.

[Christen, 2012b] Christen, P. (2012b). A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 24(9):1537–1555.

[Christen and Goiser, 2007] Christen, P. and Goiser, K. (2007). Quality and Complexity Measures for Data Linkage and Deduplication. In *Quality Measures in Data Mining*, pages 127–151. Springer, Berlin, Heidelberg.

[Christen and Vatsalan, 2013] Christen, P. and Vatsalan, D. (2013). Flexible and Extensible Generation and Corruption of Personal Data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 1165–1168.

[Christophides et al., 2020] Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., and Stefanidis, K. (2020). An Overview of End-to-End Entity Resolution for Big Data. *ACM Computing Surveys*, 53(6):127:1–127:42.

[Christophides et al., 2015] Christophides, V., Efthymiou, V., and Stefanidis, K. (2015). *Entity Resolution in the Web of Data*. Synthesis Lectures on Data, Semantics, and Knowledge. Springer International Publishing, Cham.

[Clark et al., 2020] Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *Proceedings of the 8th International Conference on Learning Representations*.

[Cochinwala et al., 2001] Cochinwala, M., Kurien, V., Lalk, G., and Shasha, D. (2001). Efficient Data Reconciliation. *Information Sciences*, 137(1-4):1–15.

[Cohen, 2000] Cohen, W. W. (2000). Data Integration using Similarity Joins and a Word-based Information Representation Language. *ACM Transactions on Information Systems*, 18(3):288–321.

[Cohen et al., 2003] Cohen, W. W., Ravikumar, P., Fienberg, S. E., et al. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. In *IIWeb*, volume 3, pages 73–78.

[Cohen and Richman, 2002] Cohen, W. W. and Richman, J. (2002). Learning to Match and Cluster Large High-dimensional Data Sets for Data Integration. In *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining*, pages 475–480.

[Conneau et al., 2020] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., et al. (2020). Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.

[Crescenzi et al., 2021] Crescenzi, V., De Angelis, A., Firmani, D., Mazzei, M., Merialdo, P., et al. (2021). Alaska: A Flexible Benchmark for Data Integration Tasks. *arXiv preprint arXiv:2101.11259*.

[Dabre et al., 2020] Dabre, R., Chu, C., and Kunchukuttan, A. (2020). A Survey of Multilingual Neural Machine Translation. *ACM Computing Surveys*, 53(5):99:1–99:38.

[de Meer Pardo et al., 2025] de Meer Pardo, F., Lehmann, C., Gehrig, D., Nagy, A., Nicoli, S., et al. (2025). GraLMatch: Matching Groups of Entities with Graphs and Language Models. In *Proceedings of the 28th International Conference on Extending Database Technology*, pages 1–12.

[Deng et al., 2020] Deng, X., Sun, H., Lees, A., Wu, Y., and Yu, C. (2020). TURL: Table Understanding through Representation Learning. *Proceedings of the VLDB Endowment*, 14(3):307–319.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186.

[Dey et al., 1998] Dey, D., Sarkar, S., and De, P. (1998). Entity Matching in Heterogeneous Databases: A Distance-based Decision Model. In *Proceedings of the Thirty-first Hawaii International Conference on System Sciences*, volume 7, pages 305–313.

[Di Cicco et al., 2019] Di Cicco, V., Firmani, D., Koudas, N., Merialdo, P., and Srivastava, D. (2019). Interpreting Deep Learning Models for Entity Resolution: An Experience Report Using LIME. In *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, pages 8:1–8:4.

[Ding et al., 2024] Ding, H., Dai, C., Wu, Y., Ma, W., and Zhou, H. (2024). SETEM: Self-ensemble Training with Pre-trained Language Models for Entity Matching. *Knowledge-Based Systems*, 293:111708.

[Doan et al., 2012] Doan, A., Halevy, A., and Ives, Z. (2012). *Principles of Data Integration*. Elsevier.

[Dolan and Brockett, 2005] Dolan, W. B. and Brockett, C. (2005). Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*, pages 9–16.

[Dong and Naumann, 2009] Dong, X. L. and Naumann, F. (2009). Data Fusion: Resolving Data Conflicts for Integration. *Proceedings of the VLDB Endowment*, 2(2):1654–1655.

[Draisbach et al., 2019] Draisbach, U., Christen, P., and Naumann, F. (2019). Transforming Pairwise Duplicates to Entity Clusters for High-quality Duplicate Detection. *Journal of Data and Information Quality*, 12(1):1–30.

[Draisbach and Naumann, 2017] Draisbach, U. and Naumann, F. (2017). DuDe: The Duplicate Detection Toolkit. In *Proceedings of the International Workshop on Quality in Databases*, page 7.

[Dziugaite et al., 2020] Dziugaite, G. K., Drouin, A., Neal, B., Rajkumar, N., Caballero, E., et al. (2020). In Search of Robust Measures of Generalization. In *Advances in Neural Information Processing Systems*, volume 33, pages 11723–11733.

[Ebaid et al., 2019] Ebaid, A., Thirumuruganathan, S., Aref, W. G., Elmagarmid, A., and Ouzzani, M. (2019). EXPLAINER: Entity Resolution Explanations. In *Proceedings of the 35th International Conference on Data Engineering*, pages 2000–2003.

[Eberius et al., 2015] Eberius, J., Braunschweig, K., Hentsch, M., et al. (2015). Building the Dresden Web Table Corpus: A Classification Approach. In *Proceedings of the 2nd International Symposium on Big Data Computing*, pages 41–50.

[Ebraheem et al., 2018] Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., and Tang, N. (2018). Distributed Representations of Tuples for Entity Resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467.

[Elfeky et al., 2002] Elfeky, M., Verykios, V., and Elmagarmid, A. (2002). TAILOR: A Record Linkage Toolbox. In *Proceedings of the 18th International Conference on Data Engineering*, pages 17–28.

[Elmagarmid et al., 2007] Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16.

[Fan et al., 2024a] Fan, J., Tu, J., Li, G., Wang, P., Du, X., et al. (2024a). Unicorn: A Unified Multi-Tasking Matching Model. *ACM SIGMOD Record*, 53(1):44–53.

[Fan et al., 2024b] Fan, M., Han, X., Fan, J., Chai, C., Tang, N., et al. (2024b). Cost-effective In-context Learning for Entity Resolution: A Design Space Exploration. In *Proceedings of the 40th International Conference on Data Engineering*, pages 3696–3709.

[Fellegi and Sunter, 1969] Fellegi, I. P. and Sunter, A. B. (1969). A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210.

[Fernando et al., 2024] Fernando, C., Banarse, D. S., Michalewski, H., Osindero, S., and Rocktäschel, T. (2024). Promptbreeder: Self-Referential Self-Improvement via Prompt Evolution. In *Proceedings of the 41st International Conference on Machine Learning*.

[Ferrara et al., 2011] Ferrara, A., Montanelli, S., Noessner, J., and Stuckenschmidt, H. (2011). Benchmarking Matching Applications on the Semantic Web. In *The Semantic Web: Research and Applications*, Lecture Notes in Computer Science, pages 108–122.

[Gal, 2011] Gal, A. (2011). *Uncertain Schema Matching*. Morgan & Claypool Publishers.

[Gao, 2020] Gao, D. (2020). Deep Hierarchical Classification for Category Prediction in E-commerce System. In *Proceedings of the 3rd Workshop on e-Commerce and Natural Language Processing*, pages 64–68.

[Gao et al., 2021] Gao, T., Yao, X., and Chen, D. (2021). SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

[Genossar et al., 2023] Genossar, B., Shraga, R., and Gal, A. (2023). FlexER: Flexible Entity Resolution for Multiple Intents. *Proceedings of the ACM on Management of Data*, 1(1):42:1–42:27.

[Gilpin et al., 2018] Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., et al. (2018). Explaining Explanations: An Overview of Interpretability of Machine Learning. In *Proceedings of the 5th International Conference on Data Science and Advanced Analytics*, pages 80–89.

[Gionis et al., 2007] Gionis, A., Mannila, H., and Tsaparas, P. (2007). Clustering Aggregation. *ACM Transactions on Knowledge Discovery from Data*, 1(1):4–es.

[Gokhale et al., 2014] Gokhale, C., Das, S., Doan, A., Naughton, J. F., Rampalli, N., et al. (2014). Corleone: Hands-off Crowdsourcing for Entity Matching. In *Proceedings of the 2014 ACM International Conference on Management of Data*, pages 601–612.

[Gomez-Cabrero et al., 2014] Gomez-Cabrero, D., Abugessaisa, I., Maier, D., Teschendorff, A., Merkenschlager, M., et al. (2014). Data Integration in the Era of Omics: Current and Future Challenges. *BMC Systems Biology*, 8:1–10.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.

[Graf et al., 2022] Graf, M., Laskowski, L., Papsdorf, F., Sold, F., Gremmelspacher, R., et al. (2022). Frost: A Platform for Benchmarking and Exploring Data Matching Results. *Proceedings of the VLDB Endowment*, 15(12):3292–3305.

[Guha et al., 2016] Guha, R. V., Brickley, D., and Macbeth, S. (2016). Schema.org: Evolution of Structured Data on the Web. *Communications of the ACM*, 59(2):44–51.

[Gunel et al., 2020] Gunel, B., Du, J., Conneau, A., and Stoyanov, V. (2020). Supervised Contrastive Learning for Pre-trained Language Model Fine-tuning. In *Proceedings of the 9th International Conference on Learning Representations*.

[Hassan and Mihalcea, 2009] Hassan, S. and Mihalcea, R. (2009). Cross-lingual Semantic Relatedness using Encyclopedic Knowledge. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1201.

[Hassanzadeh et al., 2009] Hassanzadeh, O., Chiang, F., Lee, H. C., and Miller, R. J. (2009). Framework for Evaluating Clustering Algorithms in Duplicate Detection. *Proceedings of the VLDB Endowment*, 2(1):1282–1293.

[Hassanzadeh and Miller, 2009] Hassanzadeh, O. and Miller, R. J. (2009). Creating Probabilistic Databases from Duplicated Data. *The VLDB Journal*, 18(5):1141–1166.

[He et al., 2020] He, P., Liu, X., Gao, J., and Chen, W. (2020). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. In *Proceedings of the 9th International Conference on Learning Representations*.

[Hecht and Gergle, 2010] Hecht, B. and Gergle, D. (2010). The Tower of Babel Meets Web 2.0: User-generated Content and its Applications in a Multilingual Context. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 291–300.

[Hernández and Stolfo, 1995] Hernández, M. A. and Stolfo, S. J. (1995). The Merge/Purge Problem for Large Databases. *SIGMOD Record*, 24(2):127–138.

[Hildebrandt et al., 2020] Hildebrandt, K., Panse, F., Wilcke, N., and Ritter, N. (2020). Large-Scale Data Pollution with Apache Spark. *IEEE Transactions on Big Data*, 6(2):396–411.

[Holzinger et al., 2022] Holzinger, A., Saranti, A., Molnar, C., Biecek, P., and Samek, W. (2022). Explainable AI Methods-A Brief Overview. In *Proceedings of the International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*, pages 13–38.

[Hu et al., 2022] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., et al. (2022). LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of the 10th International Conference on Learning Representations*.

[Hu et al., 2020] Hu, J., Ruder, S., Siddhant, A., Neubig, G., Firat, O., et al. (2020). XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalisation. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4411–4421.

[Hu et al., 2019] Hu, K., Gaikwad, S. N. S., Hulsebos, M., et al. (2019). VizNet: Towards A Large-Scale Visualization Learning and Benchmarking Repository. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12.

[Huang et al., 2022] Huang, J., Hu, W., Bao, Z., Chen, Q., and Qu, Y. (2022). Deep Entity Matching with Adversarial Active Learning. *The VLDB Journal*, 32(1):229–255.

[Huang and Zhao, 2024] Huang, Q. and Zhao, T. (2024). Leveraging Large Language Models for Entity Matching. *arXiv preprint arXiv:2405.20624*.

[Huang et al., 2024] Huang, W., Melo, A., and Pan, J. Z. (2024). A Large-scale Offer Alignment Model for Partitioning Filtering and Matching Product Offers. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2880–2884.

[Hulsebos et al., 2023] Hulsebos, M., Demiralp, Ç., and Groth, P. (2023). GitTables: A Large-Scale Corpus of Relational Tables. *Proceedings of the ACM on Management of Data*, 1(1):30:1–30:17.

[Ibrahim et al., 2019] Ibrahim, M., Louie, M., Modarres, C., and Paisley, J. (2019). Global Explanations of Neural Networks: Mapping the Landscape of Predictions. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 279–287.

[Ioannou et al., 2013] Ioannou, E., Rassadko, N., and Velegrakis, Y. (2013). On Generating Benchmark Data for Entity Matching. *Journal on Data Semantics*, 2(1):37–56.

[Isele and Bizer, 2013] Isele, R. and Bizer, C. (2013). Active Learning of Expressive Linkage Rules using Genetic Programming. *Journal of Web Semantics*, 23:2–15.

[Jaccard, 1912] Jaccard, P. (1912). The Distribution of the Flora in the Alpine Zone. 1. *New Phytologist*, 11(2):37–50.

[Jacovi and Goldberg, 2020] Jacovi, A. and Goldberg, Y. (2020). Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205.

[Jaiswal et al., 2020] Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. (2020). A Survey on Contrastive Self-supervised Learning. *Technologies*, 9(1):2.

[James et al., 2013] James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An Introduction to Statistical Learning*, volume 112. Springer.

[Jaro, 1989] Jaro, M. A. (1989). Advances in Record-linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420.

[Jin et al., 2021] Jin, D., Sisman, B., Wei, H., Dong, X. L., and Koutra, D. (2021). Deep Transfer Learning for Multi-source Entity Linkage via Domain Adaptation. *Proceedings of the VLDB Endowment*, 15(3).

[Junghanns et al., 2017] Junghanns, M., Petermann, A., Neumann, M., and Rahm, E. (2017). Management and Analysis of Big Graph Data: Current Systems and Open Challenges. *Handbook of Big Data Technologies*, pages 457–505.

[Kanza et al., 2018] Kanza, S., Stolz, A., Hepp, M., et al. (2018). What Does an Ontology Engineering Community Look Like? A Systematic Analysis of the schema.org Community. In *Proceedings of the 15th European Semantic Web Conference*, pages 335–350.

[Kärle et al., 2016] Kärle, E., Fensel, A., Toma, I., and Fensel, D. (2016). Why Are There More Hotels in Tyrol than in Austria? Analyzing Schema.org Usage in the Hotel Domain. In *Information and Communication Technologies in Tourism 2016*, pages 99–112.

[Kasai et al., 2019] Kasai, J., Qian, K., Gurajada, S., Li, Y., and Popa, L. (2019). Low-resource Deep Entity Resolution with Transfer and Active Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5851–5861.

[Kaya and Bilge, 2019] Kaya, M. and Bilge, H. Ş. (2019). Deep Metric Learning: A Survey. *Symmetry*, 11(9):1066.

[Khosla et al., 2020] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., et al. (2020). Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

[Kirielle et al., 2022] Kirielle, N., Christen, P., and Ranbaduge, T. (2022). TransER: Homogeneous Transfer Learning for Entity Resolution. In *Proceedings of the 25th International Conference on Extending Database Technology*, pages 118–130.

[Klyne, 2004] Klyne, G. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. *http://www.w3.org/TR/rdf-concepts/*.

[Konda et al., 2016] Konda, P., Das, S., Suganthan G. C., P., Doan, A., Ardalan, A., et al. (2016). Magellan: Toward Building Entity Matching Management Systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208.

[Köpcke et al., 2010] Köpcke, H., Thor, A., and Rahm, E. (2010). Evaluation of Entity Resolution Approaches on Real-World Match Problems. *Proceedings of the VLDB Endowment*, 3(1-2):484–493.

[Korini and Bizer, 2023] Korini, K. and Bizer, C. (2023). Column Type Annotation using ChatGPT. In *CEUR Workshop Proceedings*, volume 3462, pages 1–12.

[Korini et al., 2022] Korini, K., Peeters, R., and Bizer, C. (2022). SOTAB: The WDC Schema.org Table Anotation Benchmark. In *CEUR Workshop Proceedings*, volume 3320, pages 14–19.

[Kuhn, 1955] Kuhn, H. W. (1955). The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

[Lacoste-Julien et al., 2013] Lacoste-Julien, S., Palla, K., Davies, A., Kasneci, G., Graepel, T., et al. (2013). SIGMa: Simple Greedy Matching for Aligning Large Knowledge Bases. In *Proceedings of the 19th ACM International Conference on Knowledge Discovery and Data Mining*, pages 572–580.

[Lan et al., 2020] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., et al. (2020). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *Proceedings of the 8th International Conference on Learning Representations*.

[Lassila et al., 2001] Lassila, O., Hendler, J., and Berners-Lee, T. (2001). The Semantic Web. *Scientific American*, 284(5):34–43.

[Lee et al., 2020] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., et al. (2020). BioBERT: A Pre-trained Biomedical Language Representation Model for Biomedical Text Mining. *Bioinformatics*, 36(4):1234–1240.

[Lehmberg, 2019] Lehmberg, O. (2019). *Web Table Integration and Profiling for Knowledge Base Augmentation*. Doctoral dissertation, University of Mannheim.

[Lehmberg et al., 2016] Lehmberg, O., Ritze, D., Meusel, R., et al. (2016). A Large Public Corpus of Web Tables containing Time and Context Metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 75–76.

[Levenshtein et al., 1966] Levenshtein, V. I. et al. (1966). Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710.

[Li et al., 2024] Li, H., Feng, L., Li, S., Hao, F., Zhang, C. J., et al. (2024). On Leveraging Large Language Models for Enhancing Entity Resolution. *arXiv preprint arXiv:2401.03426*.

[Li et al., 2023a] Li, P., He, Y., Yashar, D., et al. (2023a). Table-GPT: Table-tuned GPT for Diverse Table Tasks. *arXiv preprint arXiv:2310.09263*.

[Li et al., 2020] Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. (2020). Deep Entity Matching with Pre-Trained Language Models. *Proceedings of the VLDB Endowment*, 14(1):50–60.

[Li et al., 2023b] Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. (2023b). Effective Entity Matching with Transformers. *The VLDB Journal*, 32(6):1215–1235.

[Li et al., 2021] Li, Y., Li, J., Suhara, Y., Wang, J., Hirota, W., and Tan, W.-C. (2021). Deep Entity Matching: Challenges and Opportunities. *Journal of Data and Information Quality (JDIQ)*, 13(1):1–17.

[Lim et al., 1996] Lim, E.-P., Srivastava, J., Prabhakar, S., and Richardson, J. (1996). Entity Identification in Database Integration. *Information Sciences*, 89(1):1–38.

[Liu et al., 2022] Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., et al. (2022). What Makes Good In-Context Examples for GPT-3? In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114.

[Liu et al., 2023] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., et al. (2023). Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9).

[Liu et al., 2019a] Liu, X., He, P., Chen, W., and Gao, J. (2019a). Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

[Liu et al., 2019b] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., et al. (2019b). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

[Lopez, 2008] Lopez, A. (2008). Statistical Machine Translation. *ACM Computing Surveys*, 40(3):8:1–8:49.

[Loster et al., 2021] Loster, M., Koumarelas, I., and Naumann, F. (2021). Knowledge Transfer for Entity Resolution with Siamese Neural Networks. *Journal of Data and Information Quality*, 13(1):2:1–2:25.

[Lu et al., 2023a] Lu, Q., Ding, L., Xie, L., Zhang, K., Wong, D. F., and Tao, D. (2023a). Toward Human-Like Evaluation for Natural Language Generation with Error Analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5892–5907.

[Lu et al., 2023b] Lu, Q., Qiu, B., Ding, L., Xie, L., and Tao, D. (2023b). Error Analysis Prompting Enables Human-like Translation Evaluation in Large Language Models: A Case Study on ChatGPT. *Preprint*.

[McNamee et al., 2011] McNamee, P., Mayfield, J., Lawrie, D., Oard, D., and Doermann, D. (2011). Cross-Language Entity Linking. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 255–263.

[McVitie and Wilson, 1970] McVitie, D. G. and Wilson, L. B. (1970). Stable Marriage Assignment for Unequal Sets. *BIT Numerical Mathematics*, 10(3):295–309.

[Meduri et al., 2020] Meduri, V. V., Popa, L., Sen, P., and Sarwat, M. (2020). A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching. In *Proceedings of the 2020 ACM International Conference on Management of Data*, pages 1133–1147.

[Meusel et al., 2015] Meusel, R., Bizer, C., and Paulheim, H. (2015). A Web-scale Study of the Adoption and Evolution of the schema.org Vocabulary over Time. In *Proceedings of the 2015 Conference on Web Intelligence, Mining and Semantics*, pages 1–11.

[Meusel et al., 2014] Meusel, R., Petrovski, P., and Bizer, C. (2014). The WebDataCommons Microdata, RDFa and Microformat Dataset Series. In *Proceedings of the 13th International Semantic Web Conference*, Lecture Notes in Computer Science, pages 277–292.

[Mika, 2015] Mika, P. (2015). On schema.org and why it Matters for the Web. *IEEE Internet Computing*, 19(4):52–55.

[Mika and Potter, 2012] Mika, P. and Potter, T. (2012). Metadata Statistics for a Large Web Corpus. *Linked Data on the Web*, 937.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 26th Conference on Neural Information Processing Systems*, pages 3111–3119.

[Miller et al., 2000] Miller, R. J., Haas, L. M., and Hernández, M. A. (2000). Schema Mapping as Query Discovery. In *Proceedings of the 26th International Conference on Very Large Data Bases*.

[Mitlöhner et al., 2016] Mitlöhner, J., Neumaier, S., Umbrich, J., et al. (2016). Characteristics of Open Data CSV Files. In *Proceedings of the 2nd International Conference on Open and Big Data*, pages 72–79.

[Monge, 1997] Monge, A. E. (1997). An Efficient Domain-independent Algorithm for Detecting Approximately Duplicate Database Records. In *Proceedings of the ACM SIGMOD Workshop on Research Issues on Knowledge Discovery and Data Mining*.

[Monge and Elkan, 1996] Monge, A. E. and Elkan, C. P. (1996). The Field Matching Problem: Algorithms and Applications. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 267–270.

[Możdżonek et al., 2022] Możdżonek, M., Wróblewska, A., Tkachuk, S., and Łukasik, S. (2022). Multilingual Transformers for Product Matching – Experiments and a New Benchmark in Polish. In *Proceedings of the 2022 IEEE International Conference on Fuzzy Systems*, pages 1–8.

[Mrini et al., 2022] Mrini, K., Nie, S., Gu, J., Wang, S., Sanjabi, M., et al. (2022). Detection, Disambiguation, Re-ranking: Autoregressive Entity Linking as a Multi-Task Problem. In *Findings of the Association for Computational Linguistics*, pages 1972–1983.

[Mudgal et al., 2018] Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., et al. (2018). Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34.

[Mugeni and Amagasa, 2022] Mugeni, J. B. and Amagasa, T. (2022). A Graph-Based Blocking Approach for Entity Matching Using Contrastively Learned Embeddings. *ACM SIGAPP Applied Computing Review*, 22(4):37–46.

[Mugeni et al., 2023] Mugeni, J. B., Lynden, S., Amagasa, T., and Matono, A. (2023). AdapterEM: Pre-trained Language Model Adaptation for Generalized Entity Matching using Adapter-tuning. In *Proceedings of the 27th International Database Engineered Applications Symposium*, pages 140–147.

[Musavi et al., 1994] Musavi, M., Chan, K., Hummels, D., and Kalantri, K. (1994). On the Generalization Ability of Neural Network Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):659–663.

[Nananukul et al., 2024] Nananukul, N., Sisaengsuwanchai, K., and Kejriwal, M. (2024). Cost-efficient Prompt Engineering for Unsupervised Entity Resolution in the Product Matching Domain. *Discover Artificial Intelligence*, 4(1):56.

[Narayan et al., 2022] Narayan, A., Chami, I., Orr, L., and Ré, C. (2022). Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment*, 16(4):738–746.

[Narducci et al., 2013] Narducci, F., Palmonari, M., Semeraro, G., et al. (2013). Cross-language Semantic Matching for Discovering Links to e-gov Services in the LOD Cloud. In *CEUR Workshop Proceedings*, volume 992, pages 21–32.

[Naumann et al., 2006] Naumann, F., Bilke, A., Bleiholder, J., and Weis, M. (2006). Data Fusion in Three Steps: Resolving Schema, Tuple, and Value Inconsistencies. *IEEE Data Engineering Bulletin*, 29(2):21–31.

[Naumann and Herschel, 2022] Naumann, F. and Herschel, M. (2022). *An Introduction to Duplicate Detection*. Springer Nature.

[Newcombe and Kennedy, 1962] Newcombe, H. B. and Kennedy, J. M. (1962). Record Linkage: Making Maximum use of the Discriminating Power of Identifying Information. *Communications of the ACM*, 5(11):563–566.

[Nie et al., 2019] Nie, H., Han, X., He, B., Sun, L., Chen, B., et al. (2019). Deep Sequence-to-Sequence Entity Matching for Heterogeneous Entity Resolution. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 629–638.

[O'Hare et al., 2019] O'Hare, K., Jurek-Loughrey, A., and de Campos, C. (2019). A Review of Unsupervised and Semi-supervised Blocking Methods for Record Linkage. In *Linking and Mining Heterogeneous and Multi-view Data*, pages 79–105. Springer International Publishing, Cham.

[On et al., 2006] On, B.-W., Koudas, N., Lee, D., and Srivastava, D. (2006). Group Linkage. In *Proceedings of the 23rd IEEE International Conference on Data Engineering*, pages 496–505.

[Ouyang et al., 2022] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

[Özsu and Valduriez, 2020] Özsu, M. T. and Valduriez, P. (2020). *Principles of Distributed Database Systems*. Springer International Publishing, Cham.

[Paganelli et al., 2022] Paganelli, M., Buono, F. D., Baraldi, A., and Guerra, F. (2022). Analyzing How BERT Performs Entity Matching. *Proceedings of the VLDB Endowment*, 15(8):1726–1738.

[Paganelli et al., 2023] Paganelli, M., Tiano, D., and Guerra, F. (2023). A Multi-facet Analysis of BERT-based Entity Matching Models. *The VLDB Journal*, pages 1–26.

[Pan et al., 2024] Pan, L., Saxon, M., Xu, W., Nathani, D., Wang, X., et al. (2024). Automatically Correcting Large Language Models: Surveying the Landscape of Diverse Self-correction Strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506.

[Pan et al., 2015] Pan, X., Papailiopoulos, D., Oymak, S., Recht, B., Ramchandran, K., et al. (2015). Parallel Correlation Clustering on Big Graphs. *Advances in Neural Information Processing Systems*, 28.

[Papadakis et al., 2023a] Papadakis, G., Efthymiou, V., Thanos, E., Hassanzadeh, O., and Christen, P. (2023a). An Analysis of One-to-One Matching Algorithms for Entity Resolution. *The VLDB Journal*, 32(6):1369–1400.

[Papadakis et al., 2022] Papadakis, G., Fisichella, M., Schoger, F., et al. (2022). How to Reduce the Search Space of Entity Resolution: with Blocking or Nearest Neighbor Search? *arXiv preprint arXiv:2202.12521*.

[Papadakis et al., 2023b] Papadakis, G., Fisichella, M., Schoger, F., et al. (2023b). Benchmarking Filtering Techniques for Entity Resolution. In *Proceedings of the 39th IEEE International Conference on Data Engineering*, pages 653–666.

[Papadakis et al., 2021] Papadakis, G., Ioannou, E., Thanos, E., and Palpanas, T. (2021). *The Four Generations of Entity Resolution*. Synthesis Lectures on Data Management. Springer International Publishing, Cham.

[Papadakis et al., 2020] Papadakis, G., Skoutas, D., Thanos, E., and Palpanas, T. (2020). Blocking and Filtering Techniques for Entity Resolution: A Survey. *ACM Computing Surveys*, 53(2):31:1–31:42.

[Papadakis et al., 2016] Papadakis, G., Svirsky, J., Gal, A., and Palpanas, T. (2016). Comparative Analysis of Approximate Blocking Techniques for Entity Resolution. *Proceedings of the VLDB Endowment*, 9(9):684–695.

[Parikh et al., 2016] Parikh, A., Täckström, O., Das, D., and Uszkoreit, J. (2016). A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., et al. (2011). Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Peeters and Bizer, 2021] Peeters, R. and Bizer, C. (2021). Dual-Objective Fine-Tuning of BERT for Entity Matching. *Proceedings of the VLDB Endowment*, 14(10):1913–1921.

[Peeters and Bizer, 2022a] Peeters, R. and Bizer, C. (2022a). Cross-Language Learning for Product Matching. In *Companion Proceedings of the Web Conference 2022*, pages 236–238.

[Peeters and Bizer, 2022b] Peeters, R. and Bizer, C. (2022b). Supervised Contrastive Learning for Product Matching. In *Companion Proceedings of the Web Conference 2022*, pages 248–251.

[Peeters and Bizer, 2023] Peeters, R. and Bizer, C. (2023). Using ChatGPT for Entity Matching. In *Proceedings of the 27th Conference on Advances in Databases and Information Systems*, pages 221–230.

[Peeters et al., 2020a] Peeters, R., Bizer, C., and Glavaš, G. (2020a). Intermediate Training of BERT for Product Matching. In *CEUR Workshop Proceedings*, volume 2726, pages 2–6.

[Peeters et al., 2024a] Peeters, R., Brinkmann, A., and Bizer, C. (2024a). The Web Data Commons Schema.org Table Corpora. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1079–1082.

[Peeters et al., 2024b] Peeters, R., Der, R. C., and Bizer, C. (2024b). WDC Products: A Multi-Dimensional Entity Matching Benchmark. In *Proceedings of the 27th International Conference on Extending Database Technology*, pages 22–33.

[Peeters et al., 2020b] Peeters, R., Primpeli, A., Wichtlhuber, B., and Bizer, C. (2020b). Using schema.org Annotations for Training and Maintaining Product Matchers. In *Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics*, pages 195–204.

[Peeters et al., 2025] Peeters, R., Steiner, A., and Bizer, C. (2025). Entity Matching using Large Language Models. In *Proceedings of the 28th International Conference on Extending Database Technology*.

[Pei et al., 2020] Pei, S., Yu, L., Yu, G., and Zhang, X. (2020). REA: Robust Cross-lingual Entity Alignment Between Knowledge Graphs. In *Proceedings of the 26th ACM International Conference on Knowledge Discovery & Data Mining*, pages 2175–2184.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

[Petrovski et al., 2017] Petrovski, P., Primpeli, A., Meusel, R., and Bizer, C. (2017). The WDC Gold Standards for Product Feature Extraction and Product Matching. In *E-Commerce and Web Technologies*, Lecture Notes in Business Information Processing, pages 73–86.

[Phang et al., 2018] Phang, J., Févry, T., and Bowman, S. R. (2018). Sentence Encoders on Stilts: Supplementary Training on Intermediate Labeled-data Tasks. *arXiv preprint arXiv:1811.01088*.

[Pikuliak et al., 2021] Pikuliak, M., Šimko, M., and Bieliková, M. (2021). Cross-Lingual Learning for Text Processing: A Survey. *Expert Systems with Applications*, 165:113765.

[Primpeli and Bizer, 2020] Primpeli, A. and Bizer, C. (2020). Profiling Entity Matching Benchmark Tasks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3101–3108.

[Primpeli and Bizer, 2022] Primpeli, A. and Bizer, C. (2022). Impact of the Characteristics of Multi-source Entity Matching Tasks on the Performance of Active Learning Methods. In *The Semantic Web*, Lecture Notes in Computer Science, pages 113–129.

[Primpeli et al., 2019] Primpeli, A., Peeters, R., and Bizer, C. (2019). The WDC Training Dataset and Gold Standard for Large-Scale Product Matching. In *Workshop on e-Commerce and NLP, Companion Proceedings of the 2019 World Wide Web Conference*, pages 381–386.

[Pruksachatkun et al., 2020] Pruksachatkun, Y., Phang, J., Liu, H., Htut, P. M., Zhang, X., et al. (2020). Intermediate-Task Transfer Learning with Pretrained Models for Natural Language Understanding: When and Why Does It Work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247.

[Pruthi et al., 2020] Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. (2020). Estimating Training Data Influence by Tracing Gradient Descent. In *Advances in Neural Information Processing Systems*, volume 33, pages 19920–19930.

[Qiu et al., 2015] Qiu, D., Barbosa, L., Dong, X. L., Shen, Y., and Srivastava, D. (2015). Dexter: Large-scale Discovery and Extraction of Product Specifications on the Web. *Proceedings of the VLDB Endowment*, 8(13):2194–2205.

[Radulovic et al., 2021] Radulovic, N., Bifet, A., and Suchanek, F. (2021). Confident Interpretations of Black Box Classifiers. In *Proceedings of the 2021 International Joint Conference on Neural Networks*, pages 1–8.

[Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A. (2001). A Survey of Approaches to Automatic Schema Matching. *The VLDB Journal*, 10:334–350.

[Reimers and Gurevych, 2019] Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992.

[Reinsel et al., 2018] Reinsel, D., Gantz, J., and Rydning, J. (2018). The Digitization of the World from Edge to Core. *Framingham: International Data Corporation*, 16:1–28.

[Remadi et al., 2024] Remadi, A., El Hage, K., Hobeika, Y., and Bugiotti, F. (2024). To Prompt or not to Prompt: Navigating the use of Large Language

Models for Integrating and Modeling Heterogeneous Data. *Data & Knowledge Engineering*, 152:102313.

[Ribeiro et al., 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144.

[Ribeiro et al., 2018] Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-Precision Model-Agnostic Explanations. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

[Rinser et al., 2013] Rinser, D., Lange, D., and Naumann, F. (2013). Cross-Lingual Entity Matching and Infobox Alignment in Wikipedia. *Information Systems*, 38(6):887–907.

[Ruder, 2017] Ruder, S. (2017). An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098*.

[Ruder et al., 2019] Ruder, S., Vulić, I., and Søgaard, A. (2019). A Survey of Cross-lingual Word Embedding Models. *Journal of Artificial Intelligence Research*, 65:569–631.

[Saeedi et al., 2017] Saeedi, A., Peukert, E., and Rahm, E. (2017). Comparative Evaluation of Distributed Clustering Schemes for Multi-Source Entity Resolution. In *Proceedings of the 21st Conference on Advances in Databases and Information Systems*, pages 278–293.

[Sai et al., 2022] Sai, A. B., Mohankumar, A. K., and Khapra, M. M. (2022). A Survey of Evaluation Metrics used for NLG Systems. *ACM Computing Surveys*, 55(2):1–39.

[Sanh et al., 2022] Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., et al. (2022). Multitask Prompted Training Enables Zero-Shot Task Generalization. In *Proceedings of the 10th International Conference on Learning Representations*.

[Sarawagi, 2008] Sarawagi, S. (2008). Information Extraction. *Foundations and Trends in Databases*, 1(3):261–377.

[Sarawagi and Bhamidipaty, 2002] Sarawagi, S. and Bhamidipaty, A. (2002). Interactive Deduplication using Active Learning. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining*, pages 269–278.

[Saveta et al., 2015] Saveta, T., Daskalaki, E., Flouris, G., Fundulaki, I., Herschel, M., and Ngomo, A.-C. N. (2015). LANCE: Piercing to the Heart of Instance Matching Tools. In *Proceedings of the 14th International Semantic Web Conference*, pages 375–391.

[Schroff et al., 2015] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.

[Seaborne and Prud'hommeaux, 2008] Seaborne, A. and Prud'hommeaux, E. (2008). SPARQL query language for RDF. *W3C Recommendation, W3C*.

[Sevgili et al., 2022] Sevgili, Ö., Shelmanov, A., Arkhipov, M., Panchenko, A., and Biemann, C. (2022). Neural Entity Linking: A Survey of Models Based on Deep Learning. *Semantic Web*, 13(3):527–570.

[Shah et al., 2018] Shah, K., Kopru, S., and Ruvini, J. D. (2018). Neural Network based Extreme Classification and Similarity Models for Product Matching. In *Proceedings of the 2018 Conference of the Association for Computational Linguistics, Volume 3*, pages 8–15.

[Shi et al., 2024] Shi, S., Xu, Z., Hu, B., and Zhang, M. (2024). Generative Multimodal Entity Linking. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pages 7654–7665.

[Shraga et al., 2020] Shraga, R., Gal, A., and Roitman, H. (2020). ADnEV: Cross-Domain Schema Matching Using Deep Similarity Matrix Adjustment and Evaluation. *Proceedings of the VLDB Endowment*, 13(9):1401–1415.

[Sil et al., 2018] Sil, A., Kundu, G., Florian, R., and Hamza, W. (2018). Neural Cross-Lingual Entity Linking. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, volume 32, pages 5464–5472.

[Silla and Freitas, 2011] Silla, C. N. and Freitas, A. A. (2011). A Survey of Hierarchical Classification across Different Application Domains. *Data Mining and Knowledge Discovery*, 22(1):31–72.

[Simo-Serra et al., 2015] Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., et al. (2015). Discriminative Learning of Deep Convolutional Feature Point Descriptors. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*, pages 118–126.

[Sisaengsuwanchai et al., 2023] Sisaengsuwanchai, K., Nananukul, N., and Kejriwal, M. (2023). How Does Prompt Engineering Affect ChatGPT Performance on Unsupervised Entity Resolution? *arXiv preprint arXiv:2310.06174*.

[Steiner et al., 2024] Steiner, A., Peeters, R., and Bizer, C. (2024). Fine-tuning Large Language Models for Entity Matching. *arXiv preprint arXiv:2409.08185*.

[Steorts et al., 2014] Steorts, R. C., Ventura, S. L., Sadinle, M., and Fienberg, S. E. (2014). A Comparison of Blocking Methods for Record Linkage. *Privacy in Statistical Databases*, pages 253–268.

[Suhara et al., 2022] Suhara, Y., Li, J., Li, Y., Zhang, D., Demiralp, Ç., et al. (2022). Annotating Columns with Pre-trained Language Models. In *Proceedings of the 2022 International Conference on Management of Data*, pages 1493–1503.

[Sun et al., 2024] Sun, C., Xu, Y., Shen, D., and Nie, T. (2024). Matching Feature Separation Network for Domain Adaptation in Entity Matching. In *Proceedings of the ACM Web Conference 2024*, pages 1975–1985.

[Sun et al., 2021] Sun, X., Yang, D., Li, X., Zhang, T., Meng, Y., et al. (2021). Interpreting Deep Learning Models in Natural Language Processing: A Review. *arXiv preprint arXiv:2110.10470*.

[Thirumuruganathan et al., 2019] Thirumuruganathan, S., Ouzzani, M., and Tang, N. (2019). Explaining Entity Resolution Predictions: Where are we and What needs to be done? In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pages 1–6.

[Trabelsi et al., 2022] Trabelsi, M., Heflin, J., and Cao, J. (2022). DAME: Domain Adaptation for Matching Entities. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1016–1024.

[Tu et al., 2022] Tu, J., Fan, J., Tang, N., Wang, P., Chai, C., et al. (2022). Domain Adaptation for Deep Entity Resolution. In *Proceedings of the 2022 International Conference on Management of Data*, pages 443–457.

[Van Dongen, 2000] Van Dongen, S. (2000). Graph Clustering by Flow Simulation. *PhD thesis, University of Utrecht*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., et al. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

[Wadhwa et al., 2024] Wadhwa, S., Krishnan, A., Wang, R., Wallace, B. C., and Kong, C. (2024). Learning from Natural Language Explanations for Generalizable Entity Matching. *arXiv preprint arXiv:2406.09330*.

[Wang et al., 2019] Wang, A., Hula, J., Xia, P., Pappagari, R., McCoy, R. T., et al. (2019). Can You Tell Me How to Get Past Sesame Street? Sentence-Level Pretraining Beyond Language Modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4465–4476.

[Wang et al., 2016] Wang, H., Li, J., and Gao, H. (2016). Efficient Entity Resolution based on Subgraph Cohesion. *Knowledge and Information Systems*, 46(2):285–314.

[Wang et al., 2011] Wang, J., Li, G., Yu, J. X., and Feng, J. (2011). Entity Matching: How Similar Is Similar. *Proceedings of the VLDB Endowment*, 4(10):622–633.

[Wang et al., 2021a] Wang, J., Li, Y., and Hirota, W. (2021a). Machamp: A Generalized Entity Matching Benchmark. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4633–4642.

[Wang et al., 2022a] Wang, J., Meng, F., Zheng, D., Liang, Y., Li, Z., Qu, J., and Zhou, J. (2022a). A Survey on Cross-Lingual Summarization. *Transactions of the Association for Computational Linguistics*, 10:1304–1323.

[Wang et al., 2022b] Wang, P., Wu, J., and Chen, X. (2022b). Multimodal Entity Linking with Gated Hierarchical Fusion and Contrastive Training. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 938–948.

[Wang et al., 2022c] Wang, P., Zeng, X., Chen, L., Ye, F., Mao, Y., et al. (2022c). PromptEM: Prompt-Tuning for Low-Resource Generalized Entity Matching. *Proceedings of the VLDB Endowment*, 16(2):369–378.

[Wang et al., 2023] Wang, R., Li, Y., and Wang, J. (2023). Sudowoodo: Contrastive Self-supervised Learning for Multi-purpose Data Integration and Preparation. In *Proceedings of the 39th IEEE International Conference on Data Engineering*, pages 1502–1515.

[Wang et al., 2024] Wang, T., Lin, H., Chen, X., Han, X., Wang, H., et al. (2024). Match, Compare, or Select? An Investigation of Large Language Models for Entity Matching. *arXiv preprint arXiv:2405.16884*.

[Wang et al., 2022d] Wang, T., Lin, H., Fu, C., Han, X., Sun, L., et al. (2022d). Bridging the Gap between Reality and Ideality of Entity Matching: A Revisting and Benchmark Re-Construction. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pages 3978–3984.

[Wang et al., 2018] Wang, Z., Lv, Q., Lan, X., and Zhang, Y. (2018). Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357.

[Wang et al., 2021b] Wang, Z., Ng, P., Nallapati, R., and Xiang, B. (2021b). Retrieval, Re-ranking and Multi-task Learning for Knowledge-base Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 347–357.

[Wei et al., 2022] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., et al. (2022). Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research.*

[Winkler, 1990] Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter model of Record Linkage. *United States Bureau of the Census.*

[Wolf et al., 2020] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., et al. (2020). Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

[Wu et al., 2020] Wu, R., Chaba, S., Sawlani, S., Chu, X., and Thirumuruganathan, S. (2020). ZeroER: Entity Resolution using Zero Labeled Examples. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1149–1164.

[Xia et al., 2024] Xia, Y., Chen, J., Li, X., and Gao, J. (2024). APrompt4EM: Augmented Prompt Tuning for Generalized Entity Matching. *arXiv preprint arXiv:2405.04820.*

[Xin et al., 2024] Xin, A., Qi, Y., Yao, Z., Zhu, F., Zeng, K., et al. (2024). LLMAEL: Large Language Models are Good Context Augmenters for Entity Linking. *arXiv preprint arXiv:2407.04020.*

[Xu et al., 2019] Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., and Zhu, J. (2019). Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges. In *Proceedings of the 8th International Conference on Natural Language Processing and Chinese Computing*, pages 563–574.

[Xu and Wang, 2024] Xu, Z. and Wang, N. (2024). Low-Resource Entity Resolution with Domain Generalization and Active Learning. *Neurocomputing*, 599:128131.

[Yang et al., 2024] Yang, J., Zhou, K., Li, Y., and Liu, Z. (2024). Generalized Out-of-Distribution Detection: A Survey. *International Journal of Computer Vision*, pages 1–28.

[Yang et al., 2022] Yang, Y., Wu, Z., Yang, Y., Lian, S., Guo, F., et al. (2022). A Survey of Information Extraction based on Deep Learning. *Applied Sciences*, 12(19):9691.

[Yao et al., 2022] Yao, D., Gu, Y., Cong, G., Jin, H., and Lv, X. (2022). Entity Resolution with Hierarchical Graph Attention Networks. In *Proceedings of the 2022 International Conference on Management of Data*, pages 429–442.

[Ye et al., 2022] Ye, C., Jiang, S., Zhang, H., Wu, Y., Shi, J., et al. (2022). JointMatcher: Numerically-aware Entity Matching Using Pre-trained Language Models with Attention Concentration. *Knowledge-Based Systems*, 251:109033.

[Yin et al., 2020] Yin, P., Neubig, G., Yih, W.-t., and Riedel, S. (2020). TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426.

[Zeakis et al., 2023] Zeakis, A., Papadakis, G., Skoutas, D., and Koubarakis, M. (2023). Pre-trained Embeddings for Entity Resolution: An Experimental Analysis. *Proceedings of the VLDB Endowment*, 16(9):2225–2238.

[Zhan et al., 2021] Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., et al. (2021). Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512.

[Zhang et al., 2023] Zhang, H., Dong, Y., Xiao, C., and Oyamada, M. (2023). Large Language Models as Data Preprocessors. *arXiv preprint arXiv:2308.16361*.

[Zhang et al., 2024a] Zhang, H., Dong, Y., Xiao, C., and Oyamada, M. (2024a). Jellyfish: Instruction-Tuning Local Large Language Models for Data Preprocessing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8754–8782.

[Zhang et al., 2024b] Zhang, J., Sun, H., and Ho, J. C. (2024b). EMBA: Entity Matching using Multi-Task Learning of BERT with Attention-over-Attention. In *Proceedings of the 27th International Conference on Extending Database Technology*, pages 281–293.

[Zhang and Balog, 2020] Zhang, S. and Balog, K. (2020). Web Table Extraction, Retrieval, and Augmentation: A Survey. *ACM Transactions on Intelligent Systems and Technology*, 11(2):1–35.

[Zhang and Yang, 2021] Zhang, Y. and Yang, Q. (2021). A Survey on Multitask Learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609.

[Zhang et al., 2020] Zhang, Z., Bizer, C., Peeters, R., and Primpeli, A. (2020). MWPD2020: Semantic Web challenge on Mining the Web of HTML-embedded product data. In *CEUR Workshop Proceedings*, volume 2720, pages 2–18.

[Zhao and He, 2019] Zhao, C. and He, Y. (2019). Auto-EM: End-to-end Fuzzy Entity-Matching Using Pre-trained Deep Models and Transfer Learning. In *Proceedings of the 30th World Wide Web Conference*, pages 2413–2424.

[Zhao et al., 2023] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., et al. (2023). A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223*.

[Zhao et al., 2021] Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. (2021). Calibrate Before Use: Improving Few-Shot Performance of Language Models. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12697–12706.

[Zhu et al., 2016] Zhu, L., Ghasemi-Gol, M., Szekely, P., Galstyan, A., and Knoblock, C. A. (2016). Unsupervised Entity Resolution on Multi-type Graphs. In *Proceedings of the 15th International Semantic Web Conference*, pages 649–667.