Reducing the Labeling Effort for Entity Resolution using Distant Supervision and Active Learning

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

vorgelegt von

Anna Primpeli
aus Athen

Mannheim, 2022

Dekan: Dr. Bernd Lübcke, Universität Mannheim
Referent: Professor Dr. Christian Bizer, Universität Mannheim
Korreferent: Professor Dr. Heiko Paulheim, Universität Mannheim

Tag der mündlichen Prüfung: 20. Juli 2022

Acknowledgements

Looking back at this long and hard journey towards obtaining a Ph.D., one can only realize that it takes not only a lot of work but also a great amount of support, which I was blessed to have throughout all these years.

First, I would like to express my gratitude to my primary supervisor Prof. Dr. Christian Bizer for his guidance and mentoring. He was the one to show me the way toward research and guide me out of the research deadends that frequently appeared during my studies. I will always be grateful for his constant and active presence as well as for his generous offering of time and inspiration.

Transferring the work of so many years to paper is arguably a difficult task. Therefore, I would like to sincerely thank my secondary supervisor Prof. Dr. Heiko Paulheim for his valuable feedback and comments, which helped improve my thesis. Thanks also to Dr. Samuel Baguley, Chia-Chien Hung, Ralph Peeters, Alexander Brinkmann, and Dr. Martin Mittelbach for their time and great effort in proofreading my thesis and providing me with useful advice. In addition, I would like to thank the other two members of my Ph.D. defense committee, Prof. Dr. Margret Keuper and Prof. Dr. Rainer Gemulla for their availability as well as their challenging questions and discussion during my defense.

I have always felt grateful to be surrounded by the greatest family and friends one could wish for. I am thankful to my mother Chara, my father Dimitris, and my sister Eirini for their unconditional love which I have been receiving since I can remember myself. Friends are one's extended family and I have been very lucky to have built strong friendships that have survived through hardships, distance, and time. Thanks to all my friends for always being there for me, also during my Ph.D. studies despite my busy schedule and my occasional absent-mindedness.

Finally but most importantly, I would like to deeply thank my dear husband and best friend Johannes for his constant love as well as his emotional and practical support during all this time. From motivating me in the early insecure days to proofreading, again and again, my thesis at the very end, you have contributed the most to my success. You believed in me more than I could ever believe in myself and have always been present to cheer with me in my successes and comfort me in my failures. Your patience, help, and encouragement were a light in my way. Thank you for traveling with me on this long journey which would have been much harder, if not impossible, without you. I dedicate this work to you.

Abstract

Entity resolution is the task of identifying records in one or more data sources which refer to the same real-world object. It is often treated as a supervised binary classification task in which a labeled set of matching and non-matching record pairs is used for training a machine learning model.

Acquiring labeled data for training machine learning models is expensive and time-consuming, as it typically involves one or more human annotators who need to manually inspect and label the data. It is thus considered a major limitation of supervised entity resolution methods. In this thesis, we research two approaches, relying on distant supervision and active learning, for reducing the labeling effort involved in constructing training sets for entity resolution tasks with different profiling characteristics.

Our first approach investigates the utility of semantic annotations found in HTML pages as a source of distant supervision. We profile the adoption growth of semantic annotations over multiple years and focus on product-related schema.org annotations. We develop a pipeline for cleansing and grouping semantically annotated offers describing the same products, thus creating the WDC Product Corpus, the largest publicly available training set for entity resolution. The high predictive performance of entity resolution models trained on offer pairs from the WDC Product Corpus clearly demonstrates the usefulness of semantic annotations as distant supervision for product-related entity resolution tasks.

Our second approach focuses on active learning techniques, which have been widely used for reducing the labeling effort for entity resolution in related work. Yet, we identify two research gaps: the inefficient initialization of active learning and the lack of active learning methods tailored to multi-source entity resolution. We address the first research gap by developing an unsupervised method for initializing and further assisting the complete active learning workflow. Compared to active learning baselines that use random sampling or transfer learning for initialization, our method guarantees high anytime performance within a limited labeling budget for tasks with different profiling characteristics.

We address the second research gap by developing ALMSER, the first active learning method which uses signals inherent to multi-source entity resolution tasks for query selection and model training. Our evaluation results indicate that exploiting such signals for query selection alone has a varying effect on model performance across different multi-source entity resolution tasks. We further investigate this finding by analyzing the impact of the profiling characteristics of multi-source entity resolution tasks on the performance of active learning methods which use different signals for query selection.

Zusammenfassung

Entity Resolution bezeichnet die Aufgabe, Datensätze in einer oder mehreren Datenquellen zu identifizieren die das gleiche Realweltobjekt beschreiben. Sie wird häufig als überwachte binäre Klassifizierungsaufgabe behandelt, bei der gelabelte übereinstimmende und nicht übereinstimmende Datensatzpaare für das Training eines maschinellen Lernmodells verwendet werden.

Die Beschaffung von gelabelten Daten für das Training von maschinellen Lernmodellen ist kostspielig und zeitaufwändig, da in der Regel ein oder mehrere menschliche Annotatoren die Daten manuell prüfen und labeln müssen. Dies stellt eine wesentliche Einschränkung der überwachten Methoden zur Entity Resolution dar. In dieser Arbeit erforschen wir zwei Ansätze, welche sich auf Distant Supervision und Active Learning stützen. Beide Ansätze reduzieren den Labeling-Aufwand, der bei der Erstellung von Trainingsdaten für Entity Resolution-Aufgaben mit unterschiedlichen Eigenschaften entsteht.

Unser erster Ansatz untersucht den Nutzen von semantischen Annotationen in HTML-Seiten als Quelle für die Distant Supervision. Wir bestimmen den Zuwachs von semantischen Annotationen über mehrere Jahre hinweg und fokussieren uns auf produktbezogene schema.org-Annotationen. Wir entwickeln eine Pipeline zur Bereinigung und Gruppierung semantisch annotierter Angebote, die dieselben Produkte beschreiben und erstellen so den WDC Product Corpus - das größte öffentlich verfügbare Trainingsset für Entity Resolution. Die hohe Genauigkeit von Entity Resolution-Modellen, die auf Datensatzpaaren aus dem WDC Product Corpus trainiert wurden, zeigt deutlich die Nützlichkeit semantischer Annotationen als Distant Supervision für produktbezogene Entity Resolution-Aufgaben.

Unser zweiter Ansatz konzentriert sich auf Active Learning-Methoden, die in verwandten Arbeiten zur Reduzierung des Labeling-Aufwands für Entity Resolution untersucht wurden. Dabei stellen wir zwei Forschungslücken fest: die ineffiziente Initialisierung des Active Learnings sowie das Fehlen von Active Learning-Methoden, die speziell auf Multi-Source Entity Resolution zugeschnitten sind. Die erste Forschungslücke adressieren wir, indem wir eine unüberwachte Methode zur Initialisierung und Unterstützung des gesamten Active Learning-Prozesses entwickeln. Im Vergleich zu Active Learning-Methoden, welche Zufallsstichproben oder Transfer Learning für die Initialisierung verwenden, garantiert unsere Methode bei einem begrenzten Labeling-Budget für Aufgaben mit unterschiedlichen Eigenschaften eine konstant hohe Genauigkeit.

Die zweite Forschungslücke wird durch die Entwicklung von ALMSER adressiert - der ersten Active Learning-Methode, die für die Abfrageselektion und das Modelltraining Multi-Source-bezogene Signale verwendet. Unsere Evaluierungsergebnisse zeigen, dass die Nutzung dieser Signale für die Abfrageselektion allein unterschiedliche Effekte auf die Modellleistung bei verschiedenen Multi-Source-Entity Resolution-Aufgaben hat. Diese unterschiedlichen Effekte analysieren wir anhand des Einflusses von Eigenschaften der Multi-Source Entity Resolution-Aufgaben auf die Leistung von Active Learning-Methoden, die verschiedene Signale für die Abfrageselektion verwenden.

Contents

Chapter 1

Introduction

Over the last years, we experience a rapid increase in data generated by businesses, governmental institutions, as well as individuals. The large amounts of generated data enable a variety of applications. For example, more and more businesses offer their products on the Web to enable online shopping. Similarly, individuals can publish advertisements for trading products or services in consumer-to-consumer online environments. These large collections of data need to be effectively processed, managed and analyzed in order to serve their purposes [Christen, 2012]. A considerable amount of applications, enabled by the existence of such large data collections, rely on comparing and aggregating data from disparate sources. For instance, price comparison websites gather offers from different vendors which refer to the same product and provide the customers a comparison overview of the different prices for which the product is available. In the business domain, having a unified view of enterprise data residing in multiple databases across the company eases the execution of different operations, such as financial reporting, and supports decision making [Doan et al., 2012]. The task of providing unified access to data originating from multiple data sources is known as data integration [Dong and Srivastava, 2015]. Data integration has received increasing attention over the last years. In 2005, enterprise information integration software was estimated to have revenues of at least 0.5 billion U.S. dollars [Halevy et al., 2005], while in 2019, the total revenue amounted to 3.37 billion U.S. dollars.[1]

Entity Resolution A central task in data integration is to identify records in one or multiple data sources which describe the same real-world object [Christen, 2012]. This task is known as entity resolution. Interestingly, entity resolution is also known under many alternative terms, such as entity matching, record linkage, duplicate detection, and object identification [Christen, 2012]. Following the price comparison website example, entity resolution aims at finding all matching offers published by one or more vendors, i.e. offers that describe the same product. An

[1] Source: IDC report #US46651120, Worldwide Big Data and Analytics Software Market Shares, 2019: Investment in Data Continues (July 2020).

Table 1.1: Matching song records with different textual representations.

Song title	Song length	Artist	Release title
Let it bee	4:03	The Beatles	Let it be
Let it be (Beatles song)	243	Beatles	Let it be (Beatles album)
Let it be	-	The Beatles	Let it be

important challenge in entity resolution is the absence of commonly used object identifiers [Christen, 2012]. To circumvent this challenge, entity resolution methods need to reconcile the heterogeneity of the descriptions of records that refer to the same real-world object. The descriptions of records can be of different modalities, such as text or images.

Resolving the textual heterogeneity of matching records has been studied for decades [Fellegi and Sunter, 1969]. The reasons for the underlying textual heterogeneity are twofold. On the one hand, different data providers may choose different ways of presenting a real-world object. On the other hand, the published data are often subject to errors, such as typos and misspellings. Table 1.1 and Figure 1.1 show two examples of groups of matching records with different textual representations. Table 1.1 presents three records that refer to the song *Let it be* from the band *The Beatles*. In order for an entity resolution method to recognize that the three records refer to the same song, it needs to overcome the difference in the record attribute values, e.g. *Let it be (Beatles song)* versus *Let it be* and *4:03* versus *243*, spelling errors, e.g. *Let it bee*, as well as missing values. Figure 1.1 shows two records that describe the same smartphone product with one long span of text. The challenge for an entity resolution method, in this case, is to focus on and compare the identifying words for the smartphone product, e.g. *Samsung Galaxy S20* and *6.5 inches* and ignore less important words, e.g. *comes with* and *powered*. The different challenges entailed in entity resolution tasks, such as the ones presented in Table 1.1 and Figure 1.1, are known to have an effect on the performance of entity resolution methods [Köpcke et al., 2010; Mudgal et al., 2018].

Entity resolution methods can be distinguished into *non-learning* and *learning*, considering the available expert knowledge [Papadakis et al., 2021]. For the example of Table 1.1, a non-learning entity resolution method would require an expert to define heuristics that can group together the matching song records. Such a heuristic could be the following: *if more than 60% of the words of the shortest in length song and release titles are contained within the longer ones, then the records are matching*. Designing such heuristics requires substantial manual effort and domain knowledge. Treating entity resolution as a learning task circumvents the need for manually designing matching heuristics by employing machine learning models for solving the task [Christophides et al., 2015]. Supervised entity resolution methods constitute a specific type of learning techniques that use a labeled set, typically in the form of matching and non-matching record pairs, to train a binary classification model [Christophides et al., 2020; Papadakis et al., 2021]. The trained model can then predict matching and non-matching relations between new record pairs.

Samsung Galaxy S20 FE 5G is powered by the Qualcomm SM8250 Snapdragon 865 Octa-core processor with Adreno 650 MP11 GPU. The device comes with a big display screen of 6.5 inches. The rear camera of the smartphone consists of a triple-camera of 12 MP (wide) + 8 MP (telephoto) 3x optical zoom + 12 MP

The phone comes with a 6.50-inch touchscreen display with a resolution of 1080x2400 pixels. Samsung Galaxy S20 FE 5G is powered by an octa-core Qualcomm Snapdragon 865 processor. It comes with 8GB of RAM. As far as the cameras are concerned, the Samsung Galaxy S20 FE 5G on the rear packs a 12-megapixel primary camera; a 12-megapixel camera, and an 8-megapixel camera.

Figure 1.1: Matching phone records with different textual representations.

Reducing the Labeling Effort Acquiring an adequately large and representative set of labeled data for training machine learning models is an expensive and time-consuming task, as it requires one or more experts to manually inspect and label the data [Papadakis et al., 2021; Wu et al., 2020]. There exist various techniques to address this problem and reduce the labeling effort required for solving entity resolution tasks, including the following [Papadakis et al., 2021; Roh et al., 2019]:

- **Transfer learning** methods leverage existing labeled data from one or more related entity resolution tasks in order to train a machine learning model for a target entity resolution task for which no or limited training data are available [Thirumuruganathan et al., 2018; Zhao and He, 2019].
- **Unsupervised learning** methods relinquish the need for labeled training data and learn an entity resolution model using techniques such as clustering [Saeedi et al., 2017] and self-learning [Hou et al., 2019; Wu et al., 2020].
- **Distantly supervised** methods use weakly labeled data, i.e. training data which are labeled automatically based on heuristics. Such heuristics can be extracted from structured web data found in knowledge bases [Mintz et al., 2009] or in semantically annotated HTML pages [Meusel and Paulheim, 2014]. Typically, weakly labeled data are subject to some degree of noise, compensated by their large size [Roh et al., 2019].
- **Active learning** methods reduce the labeling effort that is required for learning an effective entity resolution model by including the human annotator in the learning loop and carefully choosing a small number of record pairs to be labeled [Papadakis et al., 2021; Settles, 2012]. The labeling process is guided iteratively by a so-called query strategy which, given the past responses of the human annotator and a measure of informativeness, assesses how informative the available unlabeled record pairs are. Only the record pairs assessed as most informative are selected for labeling [Settles, 2012].

In this thesis, we focus on using **distant supervision** as well as **active learning** methods, with the goal of reducing the labeling effort required for training machine learning models to effectively solve entity resolution tasks with **different profiling characteristics**.

Considering that the predictive performance of an entity resolution method can vary given the challenges of the task to be solved [Köpcke et al., 2010; Mudgal et al., 2018], we define a set of profiling dimensions, capturing central aspects of the tasks. We profile existing entity resolution benchmark tasks along these dimensions and identify groups of tasks entailing similar characteristics and challenges.

Towards using distant supervision to reduce the labeling effort for entity resolution, we turn our focus on the Semantic Web and explore its potential as a source of training data. To do so, we perform a three-step analysis. First, we profile the growth of Semantic Web annotations over multiple years. Second, we explore the utility of semantically annotated identifiers as supervision for entity resolution tasks of two specific domains, i.e. product and local business. Third, we focus on the product domain and develop a pipeline for cleansing and grouping semantically annotated offers describing the same real-world products. The output of the implemented pipeline is the WDC Product Corpus for entity resolution, comprising 26.5 million records of offers, grouped in 16.3 million clusters describing the same products. Combining pairwise the intra- and inter-cluster records allows us to generate the largest and most heterogeneous, in terms of the number of data sources, publicly available training set for entity resolution so far. The high predictive performance of entity resolution models trained on offer pairs from the WDC Product Corpus clearly demonstrates the usefulness of semantic annotations as distant supervision for product-related entity resolution tasks.

Towards using active learning to reduce the labeling effort for entity resolution, we focus on two research gaps identified in related work: the inefficient initialization of active learning and the lack of active learning methods tailored to multi-source entity resolution tasks.

We address the initialization issue in an unsupervised fashion. We develop a method that assigns unsupervised matching and non-matching labels to all available unlabeled record pairs before active learning starts. The unsupervised labeled record pairs assist not only the initialization step but also the complete active learning workflow.

With the purpose of reducing the labeling effort for multi-source entity resolution tasks, we develop ALMSER, an active learning method for multi-source entity resolution. ALMSER uses signals, inherent to multi-source entity resolution tasks, for query selection and model training. ALMSER comes with two query strategies ALMSERgraph and ALMSERgroup; the first query strategies tailored to multi-source entity resolution. We evaluate the distinct components of ALMSER and observe that exploiting multi-source-related signals for query selection has a different effect on model performance which depends on the multi-source entity resolution task at hand. We further investigate this finding by analyzing the impact of the profiling characteristics of multi-source entity resolution tasks on active

learning methods exploiting different signals for query selection. To enable our analysis, we propose a set of dimensions for profiling multi-source entity resolution tasks and curate a continuum of multiple tasks along these dimensions using ALMSERgen, the first multi-source entity resolution task generator.

The rest of this introductory chapter is structured into four sections. Section 1.1 motivates the methodology followed in our work towards the goal of reducing the labeling effort for efficiently solving entity resolution tasks with different profiling characteristics. Section 1.2 summarizes the main contributions of the thesis. Section 1.3 presents a short overview of the main topics addressed in each part and chapter of this thesis. Finally, in Section 1.4, we list the publications containing parts of our research.

1.1 Motivation

Entity Resolution Entity resolution has been studied for decades [Fellegi and Sunter, 1969] and remains a highly active research field across different communities, such as health researchers, statisticians, business, and computer scientists [Christen, 2012]. In 2021 alone, the estimated amount of publications related to entity resolution was over three thousand.[2]

Supervised Entity Resolution Since 2000, machine learning has reshaped the research on entity resolution, with supervised entity resolution methods achieving state-of-the-art results [Doan et al., 2012]. Supervised entity resolution methods employ labeled data, typically in the form of matching and non-matching record pairs, for training a machine learning model [Papadakis et al., 2021]. Obtaining labeled training data is one of the major limitations of supervised entity resolution methods [Papadakis et al., 2021]. For example, deep learning-based entity resolution methods require hundreds to thousands of manually labeled record pairs for training in order to achieve a good model performance [Ebraheem et al., 2018; Mudgal et al., 2018]. Even in the case of non-deep learning-based entity resolution methods, a substantial increase in model performance can be achieved when hundreds of labeled data are available for training [Köpcke et al., 2010]. Therefore, reducing the labeling effort is of significant importance. In this thesis, we aim to reduce the labeling effort involved in constructing training sets for learning entity resolution models using distant supervision and active learning.

The Semantic Web as Distant Supervision for Entity Resolution Millions of websites have started to add semantic annotations within their HTML pages in order to make web data more easily consumable for web applications [Guha et al., 2015]. Semantic annotations in HTML pages are enabled by domain-independent

[2]Calculated with dimensions.ai and the following search query: *YEAR=2021 AND TITLE.contains("entity resolution" OR "entity linkage" OR "entity matching" OR "identity resolution" OR "record linkage" OR "data deduplication" OR "duplicate detection").*

markup formats as well as annotation vocabularies. Being able to easily extract and understand web content not only serves the main goal of semantic annotations, i.e. making web data more easily consumable, but also makes the Web a rich source of structured data, which can be used as supervision for multiple downstream tasks. Indeed, semantic annotations have been used as a source of distant supervision for different applications, such as information extraction [Foley et al., 2015; Meusel and Paulheim, 2014]. However, they have not been explored in the context of entity resolution. In our work, we fill this gap and explore the potential of using semantic annotations found in HTML pages as distant supervision for entity resolution.

Semantic annotations are known to suffer from errors [Meusel and Paulheim, 2015b]. The errors occur due to the general noisy nature of the Web as well as the different levels of knowledge and understanding of the semantic vocabularies from the side of the webmasters [Meusel and Paulheim, 2015b]. In our work, we focus on product-related semantic annotations, identify common errors and develop a cleansing pipeline to overcome them.

Active Learning for Entity Resolution Active learning has been widely used to reduce the labeling effort required for learning powerful entity resolution models [Meduri et al., 2020]. Yet, we identify two gaps in related work, which we address in this thesis: the inefficient initialization of active learning and the lack of active learning methods tailored to multi-source entity resolution tasks.

Initializing active learning is a non-trivial task known to suffer from the cold start problem [Tejada et al., 2001]. The cold start problem refers to the lack of adequate labeled data in the first active learning iterations. A common practice for initializing active learning for entity resolution is to sample and manually label a seeding set of matching and non-matching record pairs [Nafa et al., 2020; Qian et al., 2017]. Such an initialization approach is expensive, as it increases the overall labeling effort. Alternatively, transfer learning has been exploited for initializing active learning [Kasai et al., 2019]. However, this assumes that there exist abundant labeled record pairs of a similar topical domain which can be efficiently transferred to the task at hand. Other rule-based entity resolution methods rely on the random initialization of the rule-based model and completely relinquish the need for labeled data for starting active learning [Isele and Bizer, 2013; Ngomo and Lyko, 2012]. Nevertheless, rule-based models have been shown to underperform other types of machine learning models, such as random forests [Meduri et al., 2020].

We address the initialization issue of active learning in an unsupervised fashion. In comparison to existing works, our method comes at no additional labeling cost for the initialization step, does not assume abundant labeled data from a related task, and uses a random forest model for learning.

Although active learning has been widely used to tackle entity resolution tasks with records originating from two data sources, it has been barely applied for multi-source entity resolution tasks. Multi-source entity resolution tasks entail certain characteristics which are complementary to the ones of two-source tasks and can

be helpful for reducing the labeling effort in an active learning setting. Although existing active learning methods, typically evaluated on two-source entity resolution tasks, are also applicable to multi-source tasks, they do not focus on additional signals provided in multi-source settings. In our work, we address this gap and develop an active learning method tailored to multi-source entity resolution.

Profiling Entity Resolution Tasks Given the same labeled data for training, the performance of entity resolution methods, relying on different types of machine learning models, has been shown to vary significantly [Köpcke et al., 2010; Mudgal et al., 2018]. For example, it has been shown that traditional machine learning-based models, such as random forests, can excel on entity resolution tasks with structured records, like the one presented in Table 1.1 [Mudgal et al., 2018]. However, this is not always the case for tasks with textual records, like the one presented in Figure 1.1, for which deep learning-based models yield, in general, better results [Li et al., 2020; Mudgal et al., 2018]. Therefore, uncovering the specific challenges associated with different entity resolution tasks is essential for understanding the strengths and weaknesses of different entity resolution methods.

Existing approaches for categorizing entity resolution tasks focus only on the properties of the data sources but ignore the influence of the training set on the difficulty of the task [Mudgal et al., 2018]. Mudgal et al. [2018] categorize entity resolution tasks into the following three categories depending on the profile of the records of the data sources to be matched: (i) structured, i.e. the records contain multiple attributes with short values, (ii) textual, i.e. the records contain attributes with long textual values, and (iii) dirty, i.e. the records contain misplaced or missing values across their attributes. However, textuality, structuredness, and density alone are not enough to understand the specific challenges associated with an entity resolution task. For example, matching two data sources with records having textual descriptions and no corner cases, i.e. all matches have high textual similarity and all non-matches have low textual similarity, is less challenging than matching data sources with textual records and many corner cases.

In our work, we propose a set of dimensions for profiling entity resolution tasks that capture properties of the data sources as well as of the training set and thus go beyond the dimensions used in related work for categorizing entity resolution tasks.[3]

1.2 Contributions

In this section, we summarize the seven contributions of the thesis. We abbreviate each of the contributions with [C1] to [C7]. The abbreviations will be used as a reference throughout the chapters.

[3]The profiling of entity resolution tasks is different from the profiling of relational data, which is known as data profiling.

1. **[C1] Profiling Dimensions** We define a set of dimensions for profiling entity resolution tasks that capture properties of the records to be matched as well as of the labeled set of record pairs used for training. The proposed profiling dimensions go beyond the ones used in related work for categorizing entity resolution tasks that focus solely on the profile of the records [Mudgal et al., 2018]. We use the profiling dimensions to compare and group together existing entity resolution benchmark tasks and uncover the specific challenges that the entity resolution methods need to overcome for each group.

2. **[C2] Adoption Trends of Semantic Annotations** We monitor and analyze the adoption of semantic annotations on the Web during the period 2012 to 2020 by extracting structured data from the Common Crawl web corpus.[4] Along with this analysis, we are the first to investigate the potential of using semantic annotations describing entities of the local business and product topical domains as distant supervision for entity resolution.

3. **[C3] WDC Product Corpus** We exploit semantically annotated product-related data extracted from the October 2016 Common Crawl web corpus and curate the WDC Product Corpus for entity resolution. The curated corpus contains 26.5 million records of offers originating from 79 thousand websites, describing 16.3 million distinct products. By combining pairwise the distinct records, we can derive the largest publicly available training set for entity resolution. The derived training set is several orders of magnitude larger than DI2KG [Crescenzi et al., 2021], the second largest training set for entity resolution, which contains offers for 70 thousand distinct products deriving from 71 different e-commerce websites.

4. **[C4] Unsupervised Bootstrapping of Active Learning for Entity Resolution** We develop an unsupervised entity resolution method for initializing and further assisting the complete active learning workflow. Compared to existing initialization methods, our method comes at no additional labeling cost and does not assume abundant labeled data of a related task. We compare our method to symbolic and subsymbolic baselines which apply random sampling and transfer learning for initialization. Our evaluation on six entity resolution tasks with different profiling characteristics shows that our method consistently outperforms symbolic baselines using the HeALER query strategy [Chen et al., 2019] and random initialization by up to 48% in F1 score in the early active learning iterations. Within a labeling budget of 100 record pairs, our method outperforms the symbolic baselines by up to 3% when random sampling is used for initialization and up to 4.2% when transfer learning from non-highly related entity resolution tasks is applied for initializing active learning. Finally, compared to subsymbolic active learning baselines inspired by the work of Kasai et al. [2019], our method consistently outperforms for a labeling budget of 500 record pairs by up to

[4] https://commoncrawl.org/

32% in F1 score and entails significantly shorter waiting times among the active learning iterations, i.e. up to 13.9 seconds in contrast to a maximum of 16 minutes per iteration with batch queries in the case of the subsymbolic baselines.

5. **[C5] Active Learning for Multi-Source Entity Resolution** We develop ALMSER, an active learning algorithm for multi-source entity resolution. ALMSER comes with two query strategies ALMSERgraph and ALMSERgroup; the first query strategies tailored to multi-source entity resolution. Our evaluation on five multi-source entity resolution tasks with different profiling characteristics shows that, within a labeling budget of 200 record pairs, ALMSER consistently outperforms active learning baseline methods that do not use multi-source-related signals and apply either a commonly used margin-based query strategy [Meduri et al., 2020], or the HeALER query strategy [Chen et al., 2019]. Our evaluation results show that the performance gain of ALMSER is more significant in the early active learning iterations than in the later ones. ALMSERgraph outperforms the HeALER and the margin-based baselines by up to 5.5 and 13.4 percentage points in F1 score, respectively, with 75 labeled record pairs. With the same labeling effort, ALMSERgroup outperforms the HeALER and the margin-based baselines by up to 2.3 and 11.6 percentage points in F1 score, respectively. After having labeled 200 record pairs, both ALMSERgraph and ALMSERgroup outperform the HeALER and the margin-based baselines by up to 1.9 and 4.8 percentage points, respectively.

6. **[C6] Multi-Source Entity Resolution Tasks Profiling and Generation** We propose a set of dimensions for profiling multi-source entity resolution tasks. Given the dimensions, we develop ALMSERgen, a multi-source entity resolution generator tool. In contrast to existing entity resolution task generators [Ioannou et al., 2013; Saveta et al., 2015], ALMSERgen covers multi-source task-related desiderata, such as the existence of groups of two-source tasks with similar patterns, as well as the overlap of entities across the different data sources within the multi-source entity resolution setting.

7. **[C7] Impact of the Profile of Multi-Source Entity Resolution Tasks on Active Learning** We are the first to study the impact of the profiling characteristics of multi-source entity resolution tasks on active learning methods exploiting different signals for selecting informative pairs for labeling. We evaluate three active learning methods using the HeALER [Chen et al., 2019], ALMSERgraph, and ALMSERgroup query strategies which exploit committee-based uncertainty, graph, and grouping signals, respectively. Our evaluation is conducted against 252 tasks generated with ALMSERgen as well as five benchmark tasks. By analyzing the results, we identify four patterns for explaining the contribution of graph and grouping signals with respect to the profile of the tasks.

1.3 Outline

We structure the thesis into three parts. We illustrate the structural outline of the thesis in Figure 1.2. The different colors correspond to the distinct parts while the layering of the blocks denotes the dependencies among the different chapters, i.e. chapters depicted in higher layers depend on the chapters of the lower ones. In this section, we give a short overview of the topics addressed in each part and chapter of this thesis.

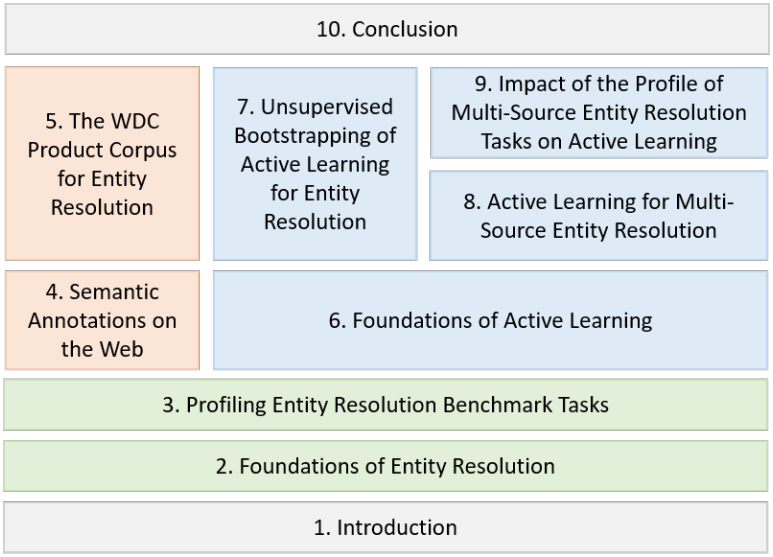


Figure 1.2: An overview of the thesis outline.

1.3.1 Part I: Entity Resolution

In the first part of the thesis, we provide the foundations of entity resolution and profile existing entity resolution benchmark tasks, covering the contribution [C1] of the thesis.

Chapter 2: Foundations of Entity Resolution In this chapter, we provide a background on entity resolution. Additionally, we formally define the entity resolution problem, as considered in our work. We present the steps of the entity resolution workflow and give an overview of different families of methods for each step. Finally, we cover common evaluation metrics for entity resolution methods.

Chapter 3: Profiling Entity Resolution Benchmark Tasks In order to evaluate and compare different entity resolution methods, a wide range of entity resolution benchmark tasks has been developed and made publicly available. Different methods have been shown to perform better than others, given the characteristics of

the task. Therefore, understanding the difficulty and diversity of entity resolution benchmark tasks is essential for selecting appropriate entity resolution methods and comparing their performance. In this chapter, we define a set of profiling dimensions that capture central challenges of entity resolution tasks and use them to group benchmark tasks with similar characteristics.

1.3.2 Part II: The Semantic Web as Distant Supervision for Entity Resolution

In the second part of the thesis, we explore the potential of using the Semantic Web as a source of distant supervision for entity resolution tasks. This part covers the contributions [C2] and [C3] of the thesis.

Chapter 4: Semantic Annotations on the Web In this chapter, we first discuss the technical realization of semantic annotations and give an overview of the main markup formats and the schema.org vocabulary. Next, we analyze the adoption trends of semantic annotations in the period 2012 to 2020. Finally, we set a specific focus and explore the potential of generating distantly labeled data for entity resolution tasks using semantic annotations related to products and businesses.

Chapter 5: The WDC Product Corpus for Entity Resolution The findings of the previous chapter indicate that more and more websites markup semantically their HTML content using the schema.org vocabulary, while there exist large amounts of markedup product-related entities accompanied by semantically annotated identifiers which can be used for extracting distantly labeled data for product-related entity resolution tasks. However, considering the noisy nature of the Web, errors in semantic annotations may reduce the quality of the distantly labeled data. In this chapter, we focus on schema.org product annotations and we develop a pipeline for extracting, cleansing, and grouping semantically annotated offers that refer to the same real-world products. The result of this pipeline is the WDC Product Corpus. We profile the curated corpus and evaluate its cleanliness based on a manually verified sample of grouped offer pairs. Finally, we assess the training quality of the corpus by evaluating symbolic and subsymbolic entity resolution models trained on training subsets derived from the corpus. Our evaluation results prove the utility of semantic annotations as a source of distant supervision for product-related entity resolution tasks.

1.3.3 Part III: Active Learning for Entity Resolution

In the third part of the thesis, we turn our focus on active learning as a means of reducing the labeling effort for entity resolution. More specifically, we address two specific research gaps: (i) the initialization of active learning and (ii) active learning for multi-source entity resolution. This part covers the contributions [C4], [C5], [C6], and [C7] of the thesis.

Chapter 6: Foundations of Active Learning In this chapter, we provide the foundations of active learning. We discuss the main scenarios in which the active learning query selection component can guide the labeling process. Next, we focus on the pool-based active learning setting and present its workflow. Finally, we discuss different families of query selection strategies and commonly-used metrics for evaluating the performance of active learning methods for entity resolution.

Chapter 7: Unsupervised Bootstrapping of Active Learning for Entity Resolution In this chapter, we deal with the cold start problem that frequently arises in active learning and refers to the lack of labeled data in the early iterations. To address this problem without increasing the labeling effort, we develop and present an unsupervised method for initializing active learning. In addition to the initialization step, our method contributes to the complete active learning workflow. To denote this combined assistance of different active learning components in an unsupervised fashion, we refer to our method as *unsupervised bootstrapping*. Furthermore, we compare our active learning method to both symbolic and subsymbolic active learning baselines that use random sampling or transfer learning for initialization.

Chapter 8: Active Learning for Multi-Source Entity Resolution In this chapter, we focus on applying active learning for multi-source entity resolution tasks. Towards this goal, we develop and present ALMSER with its two query selection strategies, ALMSERgraph and ALMSERgroup. Additionally, we evaluate ALMSER on five multi-source entity resolution tasks and compare to baseline active learning methods which do not use signals unique to the multi-source setting. Finally, we evaluate the distinct components of ALMSER, which utilize signals of the multi-source setting, by performing an ablation study.

Chapter 9: Impact of the Profile of Multi-Source Entity Resolution Tasks on Active Learning The results of the previous chapter indicate that exploiting multi-source-related signals for query selection has a varying effect on model performance across different tasks. In this chapter, we investigate the impact of the profile of multi-source entity resolution tasks on the performance of active learning methods exploiting different types of signals for query selection. We present a set of dimensions for profiling multi-source entity resolution tasks. Additionally, we develop ALMSERgen, a tool for generating multi-source entity resolution tasks along the suggested dimensions. Finally, we present and analyze the experimental results of 252 multi-source entity resolution tasks generated with ALMSERgen, as well as five benchmark tasks, and study the effect of their profiling characteristics on the performance of active learning methods using different query strategies.

Chapter 10: Conclusion This chapter summarizes the thesis and the main contributions of the individual parts. Additionally, we discuss open issues, possible relevant directions for future work, and the research impact of our work.

1.4 Published Work

Parts of the research presented in this thesis have been published previously in international journals, conferences, and workshops, which we list below. In each of the following chapters, we will explicitly mention the corresponding publications as well as the specific parts of the methodology and experiments resulting from collaborations with other researchers.

- On Part I: Entity Resolution
  - Primpeli, A. and Bizer, C. (2020). Profiling entity matching benchmark tasks. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management, CIKM '20, pages 3101–3108, New York, NY, United States. ACM.
- On Part II: The Semantic Web as Distant Supervision For Entity Resolution
  - Meusel, R., Primpeli, A., Meilicke, C., Paulheim, H., and Bizer, C. (2015). Exploiting microdata annotations to consistently categorize product offers at web scale. In Proceedings of the 16th International Conference on Electronic Commerce and Web Technologies, EC-Web '15, pages 83–99, Cham, Switzerland. Springer.
  - Primpeli, A., Meusel, R., Bizer, C., and Stuckenschmidt, H. (2017). The web data commons structured data extraction. In E-Science-Tage 2017: Forschungsdaten managen, page 1, Heidelberg. Heidelberg University.
  - Bizer, C., Primpeli, A., and Peeters, R. (2019). Using the Semantic Web as a source of training data. Datenbank-Spektrum, 19(2):127–135.
  - Primpeli, A., Peeters, R., and Bizer, C. (2019). The WDC training dataset and gold standard for large-scale product matching. In Companion Proceedings of the 2019 World Wide Web Conference, WWW '19 Companion, pages 381–386, New York, NY, United States. ACM.
  - Peeters, R., Primpeli, A., Wichtlhuber, B., and Bizer, C. (2020). Using schema.org annotations for training and maintaining product matchers. In Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics, WIMS '20, pages 195–204, New York, NY, United States. ACM.
  - Zhang, Z., Bizer, C., Peeters, R., and Primpeli, A. (2020). MWPD2020: Semantic Web challenge on mining the web of HTML-embedded product data. In Proceedings of the Semantic Web Challenge on Mining the Web of HTML-embedded Product Data co-located with the 19th International Semantic Web Conference, MWPD '20, pages 2–18, Aachen, Germany. CEUR-WS.org.

- On Part III: Active Learning for Entity Resolution
  - Primpeli, A., Bizer, C., and Keuper, M. (2020). Unsupervised bootstrapping of active learning for entity resolution. In Proceedings of the 17th Extended Semantic Web Conference, ESWC '20, pages 215–231, Cham, Switzerland. Springer.
  - Primpeli, A. and Bizer, C. (2021). Graph-boosted active learning for multi-source entity resolution. In Proceedings of the 25th International Semantic Web Conference, ISWC '21, pages 182–199, Cham, Switzerland. Springer.
  - Primpeli, A. and Bizer, C. (2022). Impact of the characteristics of multi-source entity matching tasks on the performance of active learning methods. In Proceedings of the 19th Extended Semantic Web Conference, ESWC '22, pages 113-129, Cham, Switzerland. Springer.

Part I

Entity Resolution

Chapter 2

Foundations of Entity Resolution

This chapter provides the theoretical and technical foundations of entity resolution in order to set the background of the research field in the focus of this work. Section 2.1 presents the entity resolution task and gives an overview of the evolution of entity resolution research trends. In Section 2.2, we formally define the entity resolution problem. Section 2.3 presents the general entity resolution workflow, discusses the details of each step in the workflow, and gives an overview of different families of methods for each step. Section 2.4 discusses standard metrics for evaluating the matching quality of entity resolution methods.

2.1 Background

Entity resolution (ER) is the task of identifying records from one or more data sources that refer to the same real-world object [Christen, 2012; Elmagarmid et al., 2007]. It is a vital step in data integration, which is the task of integrating different data sources, such as data repositories or databases, with different schemata and formats into one unified dataset [Christen, 2012]. Figure 2.1 presents a common view of the data integration workflow. Although the order of the data integration steps may vary across different approaches [Bilke and Naumann, 2005], entity resolution is usually preceded by schema matching [Christen, 2012; Papadakis et al., 2021], the task of identifying correspondences among the attributes of different sources and translating them to one integrated schema [Rahm and Do, 2000]. Entity resolution is followed by data fusion, which is the process of fusing multiple records that refer to the same real-world object into a single representation [Bleiholder and Naumann, 2009]. In the context of this thesis, we focus solely on the entity resolution step of the data integration workflow, as we deal with integrating data sources having a unified schema and do not proceed with fusing the matching records into a single, consolidated dataset.

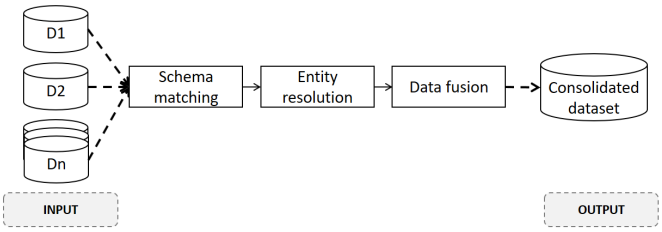


Figure 2.1: The data integration workflow.

ER across Research Communities Entity resolution is an essential component of many research areas and attracts the interest of different communities, such as health researchers, statisticians, business, and computer scientists [Christen, 2012]. Interestingly enough, every community uses disparate terms for describing the task of entity resolution: *merge/purge*, *data cleansing* or *field scrubbing* in the business community [Rahm and Do, 2000], *record matching* or *duplicate detection* in the database community, while the terms *data linkage* or *record linkage* were the first to be used for describing the entity resolution task [Dunn, 1946] by statisticians and health researchers even before entity resolution became an automated task to be solved by modern computers [Christen, 2012].

Early ER Research Trends Early works on entity resolution focus on probabilistic methods [Fellegi and Sunter, 1969; Newcombe and Kennedy, 1962]. These methods classify record pairs into matches, potential matches, and non-matches given the comparison of an aggregated similarity score, resulting from linearly combining attribute similarity values, to a pre-defined threshold value [Newcombe and Kennedy, 1962]. Porter and Winkler [1997] were the first to extend the basic probabilistic approaches by applying approximate string comparison functions to address veracity, i.e. achieving high accuracy despite the variations in the textual attribute values of the records referring to the same real-world object. Since the early work of Porter and Winkler [1997], approximate matching methods employing unsupervised and supervised classification techniques have been further developed and thoroughly studied [Dorneles et al., 2011; Elmagarmid et al., 2007; Köpcke et al., 2010; Koudas et al., 2006]. In their survey, Elmagarmid et al. [2007] study entity resolution methods that address solely the problem of lexical heterogeneity of records, while Dorneles et al. [2011] distinguish three categories of approximate matching methods addressing veracity: *content-based*, *structure-based*, which rely on the comparison of the values and structure of the records respectively, and *mixed*, which are a hybrid variant of the two categories above. Köpcke et al. [2010] compare eleven frameworks employing approximate supervised and unsupervised matching methods and denote significant matching quality differences with respect to different entity resolution tasks.

ER in the Era of Big Data The impressive increase of data creation, consumption, and storage worldwide in the last years calls for efficient and automated data cleansing methods in an effort to improve data quality [Kaisler et al., 2013]. The challenges in the era of Big Data are summarized by Laney [2001] with the 3V's model: *volume*, *variety*, and *velocity*. Identifying records referring to the same real-world objects is an integral part of the general data cleansing process. Consequently, research on entity resolution remains omnipresent, with a large body of work being published every year. As a reference, in 2010, the estimated amount of published works on entity resolution was 1.6 thousand, while in 2021, it was over 3.1 thousand.[1] Novel entity resolution methods adjust to the new challenges of the era of Big Data by employing machine learning, natural language processing, and graph-based techniques [Christen, 2012; Getoor and Machanavajjhala, 2012], while recently we observe a research trend shift towards deep learning [Barlaug and Gulla, 2021; Ebraheem et al., 2018; Mudgal et al., 2018].

Christophides et al. [2020] and Papadakis et al. [2021] present a comprehensive overview of the evolution of entity resolution methods addressing the challenges of the 3V's model of Laney in addition to the primary entity resolution challenge of veracity. In their work, they organize different entity resolution workflows and methods with respect to the challenges they address in four generations: (i) addressing only veracity, i.e. textual differences in the values of the records, (ii) addressing also volume, i.e. matching records of very large data sources, (iii) addressing also variety, i.e. unclear semantics and schema bindings, and (iv) addressing also velocity, i.e. data input as a stream. Although every generation inherits the challenges of the previous ones, none of them has either become irrelevant so far or can be considered the only focus of the research on entity resolution [Papadakis et al., 2021]. In contrast, even some novel deep learning-based entity resolution methods are regarded as 1st generation methods as they solely focus on the target of veracity [Barlaug and Gulla, 2021; Li et al., 2020; Mudgal et al., 2018; Wang et al., 2020]. Similarly, veracity is the main challenge addressed in the context of this work.

2.2 Problem Definition

We formally define the main components and the task of entity resolution using the notation and generic definitions of Christophides et al. [2015] and Papadakis et al. [2021]. The main components of the entity resolution task are the *records*, also referred to as profiles or entity descriptions, and the *entities*, which represent disparate real-world objects. A data source *record* is a structured representation of a real-world object, such as a product, a person, or a song. We consider entity resolution tasks in which records comprise a set of attribute name-value pairs, with

[1] Calculated with dimensions.ai and the following search query: *[YEAR] AND TITLE.contains("entity resolution" OR "entity linkage" OR "entity matching" OR "identity resolution" OR "record linkage" OR "data deduplication" OR "duplicate detection").*

the attribute names being shared across all data sources. An *entity* is represented by a set of records all referring to the same real-world object. More formally:

Definition 2.2.1 (Record) *A record r_i of a data source D, i.e. $r_i \in D$ is defined as $r_i = \{(a_{i_j}, v_{i_j}) : a_{i_j} \in N, v_{i_j} \in V\}$, with N, V being the set of attribute names and attribute values in D, respectively.*

Definition 2.2.2 (Entity) *An entity e_n is represented by a set of records, i.e. $e_n = \{r_1, ..., r_n\}$, with $\{r_1, ..., r_n\}$ all describing the same real-world object.*

The pairwise combinations of the records describing an entity $\{r_i, r_j\} \in e_n$, will be referred to as *matching pairs* or *matches* in this work. Any pair $\{r_i, r_j\} \in e_n$ is commutative, transitive, symmetric and reflexive [Papadakis et al., 2021]. The task of entity resolution is to discover all matches in a set of data sources S_D using a matching function M which determines whether two records r_i, r_j refer to the same entity, i.e. $M(r_i, r_j) = true$, or not, i.e. $M(r_i, r_j) = false$. More formally:

Definition 2.2.3 (Entity Resolution) *Let $R = \{r_1, ..., r_x\}$ be a set of records within S_D and $M : R \times R \rightarrow \{true, false\}$ be a boolean function. An entity resolution of R aims to create a partitioning $E = \{e_1, ...e_n\}$ of R where all $e \in E$ refer to distinct real-world objects, such that:*

- *matches are placed within the same partition,*
  i.e. $\forall r_i, r_j \in R : M(r_i, r_j) = true \Rightarrow \exists e_k \in E : r_i \in e_k \wedge r_j \in e_k$
- *each partition contains matches,*
  i.e. $\forall e_k \in E : r_i \in e_k \wedge r_j \in e_k \Rightarrow M(r_i, r_j) = true$

Both conditions of the Definition 2.2.3 are fully met in an ideal scenario in which the boolean matching function M resembles an oracle. In the context of this thesis, we consider entity resolution methods that aim to approximate the matching function M to fulfill the two conditions as closely as possible. Entity resolution is often called *data deduplication* or *dirty entity resolution* [Papadakis et al., 2021], when $|S_D| = 1$, i.e. the set of records R derives from one data source. In our work, we consider entity resolution tasks with $|S_D| = 2$, to which we refer as *two-source entity resolution tasks*, and $|S_D| > 2$, to which we refer as *multi-source entity resolution tasks*.

2.3 General Workflow

Entity resolution can be viewed as a sequence of four main steps presented in Figure 2.2 [Christen, 2012; Naumann and Herschel, 2010]. The input in the entity resolution workflow is a set of records originating from one or more data sources $D_1...D_n$, and the output is a set of matching and non-matching record pairs denoting the same or different real-world objects, respectively. In the first step, the input data sources are pre-processed so that the values of the corresponding attributes

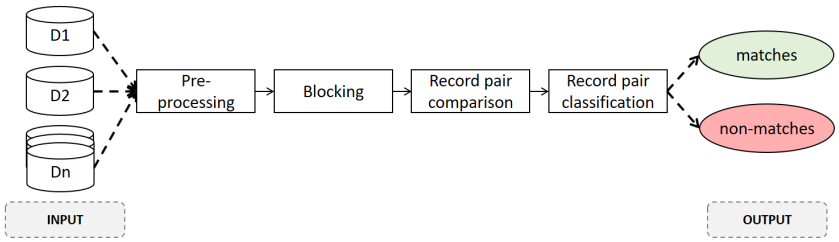


Figure 2.2: Entity resolution workflow (adapted from Christen [2012]).

have the same structure and format [Doan et al., 2012]. The second step is blocking, which minimizes the computational cost of entity resolution. Without this step, every record of each data source needs to be compared to every other record leading to a quadratic complexity, which is clearly prohibitive for large datasets. Blocking relies on an indexing function that is applied to every record and creates blocks of similar records. Only the pairwise record combinations of each block are used as input to the next step of record pair comparison. In this step, a numerical representation of the record pair similarity is calculated. Given the numeric representation of the record pairs, a binary matching decision is made whether or not the records refer to the same real-world object.

Variations of the ER Workflow The general entity resolution workflow can be found with slight variations in related works. Christophides et al. [2020] and Papadakis et al. [2021] merge the two steps of record pair comparison and classification under one step named *entity matching* and *matching*, respectively. Additionally, the pre-processing step often includes schema matching [Papadakis et al., 2021]. Barlaug and Gulla [2021] consider data pre-processing and schema matching as two different steps preceding blocking. In the case of deep learning-based entity resolution methods, the boundaries between the individual steps of the general entity resolution workflow become blurred [Barlaug and Gulla, 2021]. Barlaug and Gulla [2021] present the deep learning entity resolution workflow as a combination of four interwoven steps, which are distinguishable only up to a certain degree within the neural network architecture. Entity resolution methods that apply deep learning-based blocking methods rely on the indexable feature vectors produced by the network. This reduces the blocking step to an indexing task [Ebraheem et al., 2018; Zhang et al., 2020a]. Additionally, record pair comparison and classification are typically handled as a single step within the network [Ebraheem et al., 2018; Li et al., 2020; Mudgal et al., 2018]. Figure 2.3 presents an adapted illustration of the deep learning entity resolution workflow from the work of Barlaug and Gulla [2021].

In the following sections, we present each of the steps of the entity resolution pipeline in more detail and outline different techniques for each one of them. For each step, we distinguish between the variations of methods that employ deep

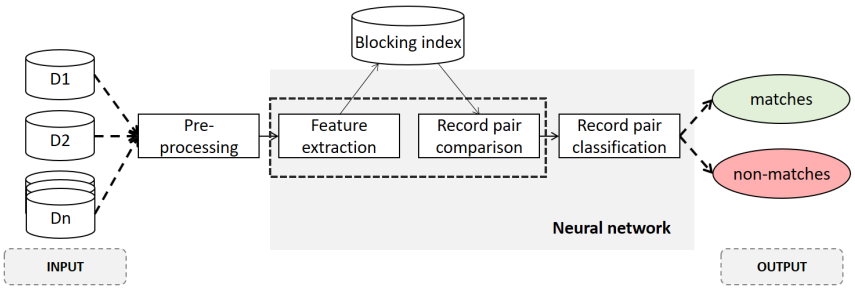


Figure 2.3: Deep learning-based entity resolution workflow (adapted from Barlaug and Gulla [2021]).

learning techniques and those that do not, for which we will use the naming conventions *subsymbolic* and *symbolic*, respectively. The naming conventions were established by Haugeland [1989] and Nilsson [1998] to describe methods that rely either on features generated within an artificial neural network architecture (subsymbolic) or explicit, interpretable features generated by humans (symbolic).

2.3.1 Pre-processing

During pre-processing, also known as data preparation [Elmagarmid et al., 2007], the structural and formatting variations of the input data sources are resolved. The pre-processing step is crucial in the entity resolution pipeline as it can facilitate the record pair comparison step and improve matching performance [Doan et al., 2012; Herzog et al., 2007]. It is necessary for symbolic and subsymbolic entity resolution methods, while for the latter, deep neural networks can effectively replace and automate some of the data pre-processing [Barlaug and Gulla, 2021]. An overview of common pre-processing techniques can be found in [Christen, 2012; Elmagarmid et al., 2007; Rahm and Do, 2000; Sarawagi, 2008]. Elmagarmid et al. [2007] define pre-processing as a sequence of the following steps: data parsing, data transformation, and data standardization.

Data Parsing During data parsing, the individual records and their attributes are located and extracted. Data parsing methods typically rely on hand-crafted pattern extractors, rule learning [Adelberg, 1998], machine learning techniques [Borkar et al., 2001; Collins and Singer, 1999; Etzioni et al., 2004] or wrapper induction methods [Crescenzi et al., 2001].

Data Transformation Data transformations are commonly performed on attribute value level with the goal to eliminate inconsistent single values or unnecessary text and ensure that the values conform with the data type of the corresponding domain [Christen, 2012; Elmagarmid et al., 2007]. Christen [2012] defines

three groups of data transformation techniques: (i) removal of unwanted characters and stopwords, (ii) abbreviation of expansions and misspelling correction, and (iii) segmentation of longer attributes to shorter ones, e.g. a date attribute of the format DD.MM.YY can be split into three attributes [day], [month] and [year]. Similarly, an address attribute can be split into [street], [number] and [postal code]. Other commonly used data transformations are lowercasing, tokenization, removal of punctuations, and common prefixes [Doan et al., 2012; Isele and Bizer, 2012].

Data Standardization Data standardization ensures that the record attribute values conform to a consistent content format [Elmagarmid et al., 2007]. For example, address standardization involves the identification of the different address elements and their combination according to a pre-defined pattern, such as "[street], [number], [postal code]".

Pre-processing for Subsymbolic ER In addition to data parsing, transformation, and standardization, Barlaug and Gulla [2021] consider the translation of records to embedding vectors and hierarchical representation learning as part of pre-processing for subsymbolic entity resolution methods. However, the distinction between these steps and record pair comparison is somewhat blurred as they are interwoven in the end-to-end neural network training process. Therefore, they are discussed as part of the record pair comparison step in Section 2.3.3.

2.3.2 Blocking

The goal of blocking is to achieve a trade-off between the reduction of unnecessary comparisons of non-matches and missing true matches [Christen, 2012; Christophides et al., 2020]. To achieve this, every blocking method typically consists of two main functions [Bilenko et al., 2006]: (i) The indexing function I, a unary function that is applied to every record r_i and returns a blocking key $b_{i_{key}}$. The blocking key defines in which blocks the record r_i will be indexed. (ii) An equality function, which is applied to a pair of indexed records and returns true if the records are indexed under at least one common block; otherwise, it returns false. The record pairs for which the equality function outputs true proceed to the next step of the entity resolution pipeline, while the rest are eliminated. More formally:

Definition 2.3.1 (Blocking) *Given a set of records $R = \{r_1, ..., r_x\}$, the index function I generates a blocking key for each $r_i \in R$, i.e. $I(r_i) = b_{i_{key}}$. Given an equality function $E_{key} \rightarrow \{true, false\}$ the records r_i, r_j are placed in the same block only if $E_{key}(b_{i_{key}}, b_{j_{key}}) = true$.*

An extensive overview and comparison of different blocking methods is provided in the surveys of O'Hare et al. [2019], Papadakis et al. [2019] and Steorts et al. [2014]. Considering the overlap of record pairs across the blocks and

the amount of blocking keys per record pair, Christen [2012] performs the following distinction of blocking method families: (i) standard blocking, (ii) sorted-neighborhood blocking, (iii) q-gram based indexing, and (iv) canopy clustering.

Blocking Methods with Single Blocking Keys Standard blocking methods assume a single blocking key per record and place each record in exactly one block, i.e. the equality function as defined in Definition 2.3.1, is strict equality [Fellegi and Sunter, 1969]. The record pairs across the blocks are not overlapping and during the matching step, only the pairwise combinations of the records within each block are considered. The sorted neighborhood approach sorts the records given their blocking key and uses a sliding window to define which records are placed in the same block [Hernández and Stolfo, 1998]. In contrast to standard blocking, which can only guarantee non-missing true matches if all matching records are assigned exactly the same blocking key, the sorted neighborhood blocking method allows for overlapping record pairs across the blocks. However, this comes with an increased comparison cost, as standard blocking always achieves better reduction than any sorted neighborhood blocking approach with a sliding window $\geqslant 2$.

Blocking Methods with Multiple or Relaxed Similarity of Blocking Keys Given the nature of the data to be matched, one blocking key per record might not be enough to capture all matching record pairs during blocking [Papadakis et al., 2021]. In such cases, blocking techniques that generate multiple blocking keys per record and thus index each record under multiple blocks are necessary. Q-gram indexing is such a technique as it generates different blocking key variations for each record by combining substrings of the blocking key of length q [Christen, 2012; Papadakis et al., 2015]. Canopy clustering is another family of blocking methods that relies on non-identical blocking keys [Christen, 2012; McCallum et al., 2000]. Such methods apply approximate string comparison functions on the blocking keys of the records. The record pairs with a blocking key similarity over a pre-defined threshold t are considered for the next matching step.

Learnable Blocking Keys All of the blocking methods mentioned above, require the selection of one or more suitable blocking keys. Depending on the entity resolution task at hand, these can be hard to determine. To circumvent the manual selection of blocking keys, blocking methods that rely on supervised learning aim to learn the optimal blocking key with respect to an objective function [Michelson and Knoblock, 2006; Shao et al., 2019], e.g. maximizing the ratio of previously uncovered matches over all non-matches [Bilenko et al., 2006].

Deep Learning-based Blocking Methods Deep learning-based blocking methods also bypass the need to manually determine blocking keys, as they reduce the complete blocking task to an indexing task [Barlaug and Gulla, 2021]. Every record is projected to the vector space based on pre-trained word-level or n-gram

level representations. Matching candidates are retrieved by selecting the nearest neighbors of each record based on a similarity function, e.g. cosine similarity [Papadakis et al., 2021]. Examples of deep learning-based blocking methods are DeepER [Ebraheem et al., 2018] and AutoBlock [Zhang et al., 2020a]. Thirumuruganathan et al. [2021] present DeepBlocker, a framework that offers multiple deep learning-based blocking solutions. DeepBlocker comes with different choices for calculating distributed representations on attribute value and record level, while it provides similarity-based, hash-based, and composite approaches for finding similar pairs.

2.3.3 Record Pair Comparison

The pairwise combinations of all records placed within the same block are considered for more detailed comparison and are often called *candidate pairs*. The record pair comparison step generates a similarity vector per candidate pair, consisting of one or more numerical values indicating how similar the records are [Christen, 2012]. The similarity vector of each candidate pair is the input to the next step of record pair classification.

Symbolic and subsymbolic entity resolution methods vary significantly on how they perform the record pair comparison step. Symbolic methods rely on handcrafted features extracted on record pair level. These features are typically extracted by applying different similarity metrics to the attribute values of the records [Christen, 2012]. Subsymbolic methods rely on neural networks, which can only handle numerical data, and therefore the input records of the candidate pairs need to be first transformed into a numerical format [Goodfellow et al., 2016]. Barlaug and Gulla [2021] refer to this step as *feature extraction* and it is highly interwoven with the record pair comparison and classification steps, as indicated in Figure 2.3. The power of subsymbolic methods lies in their ability to replace manual feature extraction by exploiting multi-layered network architectures, which can learn powerful distributed representations of the words or n-grams of the records of the candidate pairs [Papadakis et al., 2021].

In the following, we present some of the most widely used techniques and design decisions for record pair comparison for symbolic and subsymbolic entity resolution methods. A comprehensive overview of the two families of methods is provided in [Christen, 2012; Elmagarmid et al., 2007; Naumann and Herschel, 2010] and in [Barlaug and Gulla, 2021; Mudgal et al., 2018; Papadakis et al., 2021], respectively.

Record Pair Comparison for Symbolic ER

Symbolic entity resolution methods rely on data type-specific similarity metrics for the record pair comparison step [Christen, 2012; Elmagarmid et al., 2007]. Typically, multiple similarity metrics are applied on attribute level in order to deal with different types of value variations. Each metric takes as input two values

Record	Name	Artist
r1	let it be	Beatles
r2	let it bee	the Beatles

Record pair	Name-Levenshtein	Name-Jaccard	Artist-Levenshtein	Artist-Jaccard
r1-r2	0.90	0.50	0.63	0.50

Figure 2.4: Record pair comparison for symbolic ER methods - an example.

r_{i_a}, r_{j_a}, with a being the same attribute between the records r_i, r_j, and returns a score in the range $[0, 1]$. In this section, we first discuss the suitability of similarity metrics based on the application at hand. Next, we present different families of similarity metrics with a focus and more detailed explanation of the metrics used in this work for record pair comparison. All discussed similarity metrics can be formulated as distance metrics: a similarity of 1 between two values implies a distance of 0. To avoid confusion, we describe all metrics as similarity metrics. Figure 2.4 presents an example pair comparison using two of the similarity metrics presented in this section. A comprehensive overview of similarity metrics can be found in [Christen, 2012; Elmagarmid et al., 2007; Naumann and Herschel, 2010].

Suitability of Similarity Metrics The suitability of a similarity metric depends both on the underlying properties of the metric itself, i.e. what the metric is designed to capture, as well as on the nature of the values to be compared, e.g. their data type and length [Christophides et al., 2015]. For example, for the comparison of short values of a fixed set of attributes, as in the case of comparing the title and manufacturer of two product records, character-based similarity metrics are most suitable. However, this is not always the case for longer record descriptions with a loose structure, e.g. comparing two product descriptions. To compare long string values, more flexible similarity metrics operating on word-level are required.

Additionally, choosing appropriate similarity metrics depends on the underlying heterogeneity of records referring to the same real-world entity, which varies across different applications. For example, for comparing records in a data warehouse that might contain editing mistakes, such as the insertion of an extra character or accidental deletion of a character, we need metrics that compute the similarity of two values based on the edit operations required for transforming the one value into the other [Doan et al., 2012]. Such metrics are also suitable in case of OCR errors, e.g. an *O* character is recognized as zero by the OCR system. In other applications, lexical variations occur due to different spellings of words that sound identical, e.g. *Meyer* and *Meier*. For such types of errors, metrics that do not focus on the appearance of the values but are based on their sound are more suitable [Doan et al., 2012].

Character-based Similarity Metrics Character-based similarity metrics calculate the similarity of two string values on character-level and are designed to deal with typographical errors [Elmagarmid et al., 2007]. Commonly used character-

based similarity metrics are the *Jaro* similarity, originally developed for name comparison [Jaro, 1989], its extension *Jaro-Winkler*, which accounts for common string prefixes and assigns a larger similarity score to string values that match from the beginning [Winkler, 1990], as well as metrics based on edit distance. The latter calculate the distance between two string values as the minimum amount of edit operations that are required in order to convert the string value of r_{i_a} to the string value of r_{j_a}. The following three edit operations are applied: character removal, substitution, and insertion. For example, considering that each edit operation has a cost of one distance unit, a special case of edit distance known as *Levenshtein* [Levenshtein, 1966], the distance of the values "let it be" and "let it bee" is 1. This is the case since one character removal is required so that the second value is identical to the first. In order to transform the Levenshtein distance to a similarity score in the range $[0, 1]$, we apply Equation 2.1 on the string values of an attribute a of the records r_i and r_j. For the two string values "let it be" and "let it bee" the Levenshtein similarity is $1.0 - \frac{1}{10} = 0.9$.

$$S_{Levenshtein}(r_{i_a}, r_{j_a}) = 1.0 - \frac{dist_{Levenshtein}(r_{i_a}, r_{j_a})}{\max(|r_{i_a}|, |r_{j_a}|)} \tag{2.1}$$

Token-based Similarity Metrics Token-based similarity metrics calculate the similarity of two string values on token level, e.g. word or n-gram level, and are designed to deal with string values containing tokens in different order [Elmagarmid et al., 2007]. Examples of token-based similarity metrics are *Jaccard*, its variation *Jaccard containment*, also referred to as *Longest Common Substring Comparison* [Christen, 2012], and *CosineTFIDF* [Cohen et al., 2003].

The Jaccard similarity metric measures the similarity between two string values as the fraction of the number of the common tokens or n-grams to the number of the union of tokens or n-grams. Let $T_{r_{i_a}}$ and $T_{r_{j_a}}$ be the sets of tokens generated for the string values r_{i_a} and r_{j_a}, respectively. The Jaccard similarity is calculated using Equation 2.2. For example, the Jaccard similarity on word level of the values "Beatles" and "the Beatles" is $\frac{1}{2} = 0.5$.

$$S_{jaccard}(r_{i_a}, r_{j_a}) = \frac{|T_{r_{i_a}} \cap T_{r_{j_a}}|}{|T_{r_{i_a}} \cup T_{r_{j_a}}|} \tag{2.2}$$

The Jaccard containment similarity metric is calculated as shown in Equation 2.3. For example, the Jaccard containment similarity score on word-level of the values "the Beatles" and "Beatles" is 1.0.

$$S_{containment}(r_{i_a}, r_{j_a}) = \frac{|T_{r_{i_a}} \cap T_{r_{j_a}}|}{\min(|T_{r_{i_a}}|, |T_{r_{j_a}}|)} \tag{2.3}$$

The CosineTFIDF metric calculates the similarity between two string values as the cosine of the angle of their tf-idf weighting numerical vectors [Cohen et al., 2003]. The term tf-idf stands for *term frequency-inverse document frequency*. It

measures the importance of a token or n-gram as a combination of two parameters: first, its frequency in the record attribute value, and second, its frequency across all record values. Thus, the tf-idf of commonly shared tokens among all records receive lower weights than tokens that are unique to specific records. The tf-idf score of a token or n-gram t is calculated as shown in Equation 2.4. Once the values of the records r_{i_a}, r_{j_a} have been transformed to tf-idf vectors, we can calculate the CosineTFIDF as shown Equation 2.5.

$$\text{tf-idf}_t = \text{tf} \times \log \frac{R}{R_t} \tag{2.4}$$

where:

tf = the number of occurrences of t in the current record
R = the total amount of records
R_t = the number of records that contain t

$$S_{CosineTFIDF} = \frac{W_i \times W_j}{\|W_i\| \times \|W_j\|} = \frac{\sum_{t=1}^{n} w_{i,t} \times w_{j,t}}{\sqrt{\sum_{t=1}^{n} w_{i,t}^2} \times \sqrt{\sum_{t=1}^{n} w_{j,t}^2}} \tag{2.5}$$

where:

n = the unique terms over all records
$w_{i,t}$ = the tf-idf weight of term t for record r_i
$w_{j,t}$ = the tf-idf weight of term t for record r_j

Hybrid Similarity Metrics Hybrid similarity metrics combine the advantages of character-based and token-based similarity metrics with the aim to handle both typographical errors and different ordering of words [Doan et al., 2012]. Examples of hybrid similarity metrics are the *Monge Elkan* metric [Monge and Elkan, 1996], the extended Jaccard metric, and SoftTFIDF [Cohen et al., 2003]. The Monge Elkan metric computes the similarity of two values by averaging the character-based similarity of the individual tokens. The extended Jaccard metric, also referred to as relaxed Jaccard [Naumann and Herschel, 2010], combines a character-based similarity metric, such as Levenshtein similarity, with the Jaccard similarity. It calculates first the similarity of tokens, and then the similarity of the complete values considering the overlap of similar tokens. This allows similar tokens which exceed a pre-defined threshold θ, to be considered in the calculation of the intersection, as defined in Equation 2.2. As non-identical tokens can be potentially accepted as equal if they surpass the threshold θ, the set of shared tokens and the union of all tokens of the two values need to be redefined, as shown in the Equations 2.6 and 2.7. For example, given the two values ”let it bee“ and ”let it be“ the Jaccard similarity is 0.5 but the extended Jaccard similarity is $\frac{3}{3} = 1.0$, considering Levenshtein similarity as the inner edit-based similarity metric and a threshold

value $\theta = 0.6$. Similarly, SoftTFIDF is a relaxed version of the CosineTFIDF similarity metric as it uses a secondary similarity function to calculate the overlapping tokens.

$$|T_{r_{i_a}} \cap T_{r_{j_a}}|_{extended} = |\left\{(t_i, t_j : t_i \in T_{r_{i_a}} \wedge t_j \in T_{r_{j_a}} \wedge\ Sim(t_i, t_j) > \theta)\right\}| \tag{2.6}$$

$$|T_{r_{i_a}} \cup T_{r_{j_a}}|_{extended} = |\left\{(t_i, t_j : t_i \in T_{r_{i_a}} \wedge t_j \in T_{r_{i_a}}\right\}| - |T_{r_{i_a}} \cap T_{r_{j_a}}|_{extended} \tag{2.7}$$

Similarity Metrics for Non-String Values The similarity metrics presented so far can be applied to pairs of string values but cannot meaningfully assess the similarity of values of other data types, like numbers, dates, and geographical coordinates. Although one can consider the string representation of the values, applying string similarity metrics for their comparison would produce inaccurate results. For example, the numerical values "199.95" and "200" would have a string similarity of 0, using any of the presented string similarity metrics. Therefore, data type-specific metrics are needed. One way of calculating the numeric distance of values is by deriving the absolute difference of the two values normalized by a maximum absolute difference, which is either pre-defined or derived from all numerical values at hand [Christen, 2012]. The numeric distance of two values can be formulated as numeric similarity, as shown in Equation 2.8. Alternatively, the numeric similarity can be computed as the relative difference between the two values [Konda et al., 2016], as shown in Equation 2.9.

$$S_{numeric}(r_{i_a}, r_{j_a}) = 1.0 - \frac{|r_{i_a} - r_{j_a}|}{\max(r_{i_a}, r_{j_a})} \tag{2.8}$$

$$S_{numeric}(r_{i_a}, r_{j_a}) = 1.0 - \frac{2 \times |r_{i_a} - r_{j_a}|}{r_{i_a} + r_{j_a}} \tag{2.9}$$

The numeric similarity metric can be extended to cover time, date, and geographical coordinates similarity [Christen, 2012]. However, in this case, a translation of the specific data types to a numeric format is required. For example, time values can be translated into seconds before applying numeric similarity. Dates can be segmented into day, month, and year components, while geographic coordinates can be split into longitude and latitude values [Christen, 2012]. An overview of different methods for calculating the similarity of geographical locations is provided in [Koumarelas et al., 2018]. Finally, for comparing non-textual modalities such as visual, e.g. images and videos, or auditory records, a two-step approach is required: first, a transformation for representing the audio or visual record as a multi-dimensional feature vector and second, the comparison of the two vectors with a similarity metric, e.g. Euclidean distance. An overview of audio matching and image matching techniques is provided in the works of Berenzweig et al. [2004] and Mitchell [2010].

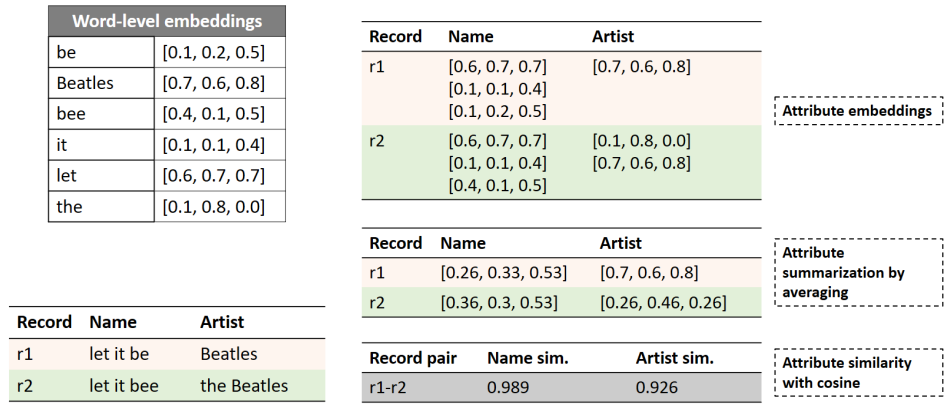
Record Pair Comparison for Subsymbolic ER

Subsymbolic entity resolution methods rely on deep neural networks, which are multi-layered neural networks [Goodfellow et al., 2016]. Multi-layered neural networks are able to automate the task of feature extraction by learning useful and highly abstract numerical representations from unstructured input records [Barlaug and Gulla, 2021]. To enable learning representations, neural networks require the input records to be transformed to a numerical format using an embedding model [Papadakis et al., 2021]. The initial numeric representations can be considered as parameters of the multi-layered network, which are learned throughout the complete training phase [Barlaug and Gulla, 2021]. The learned numeric representations of the records are then used as input for the record pair comparison. As already discussed, there is no clear line between the steps of feature extraction, i.e. initial embeddings and representation learning, and record pair comparison as both interplay within the deep neural network architecture [Barlaug and Gulla, 2021]. However, the design decisions of the initial embedding model, structure of representation learning, and how the record pair comparison is performed, come in different variations that are discussed below, while an extensive overview is provided in the work of Barlaug and Gulla [2021].

Embeddings Starting from the embedding vector representation of the record values, there are multiple embedding models to choose from. Considering their granularity, embeddings can be calculated on word- or on token-level [Barlaug and Gulla, 2021; Papadakis et al., 2021]. Word-level embeddings map every word of each attribute value of each record to a numeric vector of a pre-defined dimension using a lookup mapping dictionary. An example word embedding for the word ”Beatles“ may look like [0.7, 0.6, 0.8] considering a three-dimensional embedding space. The mappings are automatically learned with techniques like neural networks or dimensionality reduction, such that similar terms end up close in the resulting vector space [Bengio et al., 2003]. Token-level embeddings use a trained model, which produces word embeddings from the individual characters of the word, given that the characters are part of the model's vocabulary [Bojanowski et al., 2017]. Thus, token-based embeddings are less prone to out-of-vocabulary words in comparison to word-based embeddings. Given their origin, word embeddings can be distinguished into learned and pre-trained embeddings [Mudgal et al., 2018]. Learned embeddings are task-specific embeddings, trained on the available labeled data of the given task. Pre-trained embeddings are pre-trained on data from large corpora, such as Wikipedia, and can be directly reused for feature extraction [Papadakis et al., 2021]. Pre-trained embeddings can be further adjusted by fine-tuning them on the specific entity resolution task [Ebraheem et al., 2018]. Examples of word-level, pre-trained embeddings are *word2vec* [Mikolov et al., 2013] and *GloVe* [Pennington et al., 2014], while *fastText* [Bojanowski et al., 2017] is an example of a pre-trained character-based embedding model.

Representation Learning The initial mappings of the input data to numerical distributed representations are further evolved within the layers of the neural network in order to learn more accurate representations [Papadakis et al., 2021]. The learned representations can be either interdependent if they exploit information on record pair level to learn the highest representation on record level, or independent, in case they rely only on single records, while in both cases they can be designed to occur in different levels of granularity, i.e. character, word, attribute, record [Barlaug and Gulla, 2021]. Examples of works applying independent representation learning include the works of Kooli et al. [2018] and Kasai et al. [2019]. In the work of Kooli et al. [2018], simple concatenation of fastText embeddings is used for building attribute level representations. In the work of Kasai et al. [2019], an one-layer bidirectional GRU on word-level embeddings is used to retrieve attribute level representations. Examples of works applying interdependent representation learning include DeepMatcher and transformer-based methods [Li et al., 2020; Mudgal et al., 2018]. DeepMatcher is a deep learning-based framework for entity resolution, developed by Mudgal et al. [2018], which among other design decisions builds attribute representations from fastText word-level embeddings using a bidirectional GRU and cross-attention model. Transformer-based methods are considered extreme examples of interdependent representation learning as there are no boundaries between the different representation levels, while the learned representation of each token depends on every other token of the whole record pair sequence [Barlaug and Gulla, 2021].

Transformer-based Models and ER Methods Transformer-based language models were introduced in 2017 [Vaswani et al., 2017] with the aim to capture contextual information while generating word embeddings by being trained on predicting masked tokens within a sentence given its context [Papadakis et al., 2021]. Among the most commonly used transformer-based language models is BERT [Devlin et al., 2018], which stands for Bidirectional Encoder Representations from Transformers and indicates its pre-training on both right and left context, and its variants, RoBERTa [Liu et al., 2019] and DistilBERT [Sanh et al., 2019], among others. Typically all of those models are pre-trained on the masked token modeling objective. The pre-trained models can then be fine-tuned on a specific task, e.g. binary classification, using the labeled data at hand. Transformer-based language models have been applied and fine-tuned for entity resolution tasks [Brunner and Stockinger, 2020; Li et al., 2020; Peeters and Bizer, 2021]. An example of a transformer-based model for entity resolution is JointBERT [Peeters and Bizer, 2021]. In contrast to the BERT model and its pre-mentioned variants, JointBERT is a BERT-based multi-task learning variant [Caruana, 1997] specifically designed for entity resolution and pre-trained on a dual objective combining binary classification and multi-class classification.



(a) Pre-trained word embeddings and input record pair

(b) Calculation of attribute similarity

Figure 2.5: Record pair comparison for subsymbolic ER methods - an example.

Distributed Representation Comparison Subsymbolic entity resolution methods that learn distributed representations at either attribute [Kasai et al., 2019; Kooli et al., 2018] or record level [Ebraheem et al., 2018] proceed to the step of comparing the learned distributed representations and produce a similarity vector for each record pair [Barlaug and Gulla, 2021]. Some methods consider attribute-aligned comparison [Kasai et al., 2019; Mudgal et al., 2018], i.e. compare the attribute value representations of the two records, or disregard the schema and compare directly on record level [Ebraheem et al., 2018; Kooli et al., 2018].

The final comparison of the distributed representations of the candidate pair records results either in another distributed similarity representation [Ebraheem et al., 2018; Kooli et al., 2018; Mudgal et al., 2018] or a single numeric value, e.g. calculated using cosine similarity [Wolcott et al., 2018] or by applying learnable distance functions [Mudgal et al., 2018]. Figure 2.5 presents an example of comparing a record pair using sample pre-trained embeddings (Figure 2.5a) and attribute-aligned pair comparison resulting in a non-distributed similarity vector per attribute, calculated with cosine similarity (Figure 2.5b).

Methods that rely on transformers exploit self- and cross-attention mechanisms between all words and generate distributed representations for the complete sequence of record pairs [Brunner and Stockinger, 2020; Li et al., 2020]. Therefore for such methods, it is often hard to distinguish between representation learning of a single record and record pair comparison.

2.3.4 Record Pair Classification

In this step of the entity resolution pipeline, the matching decision *match* or *non-match* is made, given the similarity vector produced for the record pair candidate in the preceding step of record pair comparison [Christen, 2012]. The classification step is well distinguishable from the record pair comparison step in the general entity resolution workflow in the case of symbolic methods. Nonetheless, this is not the case for subsymbolic methods as they combine the feature extraction, record pair comparison, and classification steps in a neural network [Barlaug and Gulla, 2021]. However, for such methods the classification step is executed in distinct layers within the neural network architecture, often referred to as classification layers, which can be, to a certain extent, distinguished from the record pair comparison layers [Barlaug and Gulla, 2021]. In the following, we provide a short outlook on methods applied during the classification step for symbolic entity resolution methods and within the classification layers for subsymbolic entity resolution methods.

Record Pair Classification for Symbolic ER Several approaches have been proposed for tackling the classification task within the entity resolution workflow. Given the need for training data, the classification methods can be distinguished into unsupervised and supervised methods [Christen, 2012].

Early unsupervised approaches aggregate the similarity vector, produced in the record comparison step, into one similarity score and compare it against a predefined threshold value in order to obtain the matching decision [Bilenko and Mooney, 2003; Cohen et al., 2003; Monge and Elkan, 1996]. Alternatively, unsupervised approaches rely on hand-crafted matching rules [Hernández and Stolfo, 1998; Lim et al., 1996] or clustering approaches, such as k-means [Elfeky et al., 2002]. Saeedi et al. [2017] compare different unsupervised clustering-based techniques for entity resolution. In a later work, they develop CLIP [Saeedi et al., 2018], a clustering matching method that relies on hand-written domain-specific rules.

Supervised classification methods train machine learning classification models, using a set of labeled matching and non-matching record pairs. The trained models are then applied to unseen record pairs for which the matching decision is predicted [Christen, 2012]. Common classification models provided by entity resolution tools include decision trees [Elfeky et al., 2002], support vector machines [Christen, 2008], logistic and linear regression [Konda et al., 2016], genetic programming [Isele and Bizer, 2012] as well as ensemble classification models such as random forests and extreme gradient boosting (XGBoost) [Chen and Guestrin, 2016; Konda et al., 2016]. Köpcke et al. [2010] compare different supervised classification methods, while a comprehensive overview of both unsupervised and supervised methods is provided in [Christen, 2012; Doan et al., 2012; Elmagarmid et al., 2007].

Record Pair Classification for Subsymbolic ER As already discussed, the classification step of subsymbolic entity resolution methods is executed within the neural network either standalone or as part of the final layers of a larger network [Barlaug and Gulla, 2021]. In contrast to symbolic entity resolution methods, which can vary significantly on how the classification step is performed, there is less variety in the case of subsymbolic entity resolution methods. Commonly, a multi-layered perceptron with softmax as an activation function is applied for the classification step [Fu et al., 2019; Kasai et al., 2019; Mudgal et al., 2018]. Transformer-based entity resolution methods typically use a single dense classification layer with softmax [Brunner and Stockinger, 2020; Li et al., 2020]. An overview of the classification layer structure of different subsymbolic methods is provided by Barlaug and Gulla [2021].

2.4 Evaluation Metrics

The main goal of an entity resolution model addressing veracity is to be able to accurately assign matching or non-matching labels to record pairs describing the same or different real-world entities, respectively. Although other objectives like the interpretation of the entity resolution models might be relevant, they are not in the scope of this thesis.

In order to measure the prediction quality of an entity resolution model, a ground truth, often called *gold standard*, is required. The gold standard is a set of manually labeled and verified matching and non-matching record pairs that are similar to the data to be matched [Christen, 2012]. During evaluation, every record pair in the gold standard is assigned the label *matching* or *non-matching*, which is predicted by the matching algorithm, i.e. a function M' which approximates the boolean function M as per Definition 2.2.3. The predicted label is compared to the true match status and one of the following categories is assigned to each record pair [Christen and Goiser, 2007]:

- True Positives (TP): Matching record pairs that have been correctly classified as matches by the matching algorithm.
  More formally: $TP = |M'(r_i, r_j) = true \wedge \exists e_k \in E : (r_i, r_j) \in e_k|$
- False Positives (FP): Non-matching record pairs that have been wrongly classified as matches by the matching algorithm. Also known as Type I errors.
  More formally: $FP = |M'(r_i, r_j) = true \wedge \nexists e_k \in E : (r_i, r_j) \in e_k|$
- True Negatives (TN): Non-matching record pairs that have been correctly classified as non-matches by the matching algorithm.
  More formally: $TN = |M'(r_i, r_j) = false \wedge \nexists e_k \in E : (r_i, r_j) \in e_k|$
- False Negatives (FN): Matching record pairs that have been wrongly classified as non-matches by the matching algorithm. Also known as Type II errors.
  More formally: $FN = |M'(r_i, r_j) = false \wedge \exists e_k \in E : (r_i, r_j) \in e_k|$

Given the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), different quality measures can be calculated. The following quality measures are among the most frequently used for measuring the prediction quality of matching algorithms: *accuracy*, *precision*, *recall*, and *F1*. Accuracy measures the ratio of correctly classified record pairs to all record pairs and is calculated as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.10}$$

Precision measures how many of the classified matches are actually matches given the gold standard and is calculated as:

$$P = \frac{TP}{TP + FP} \tag{2.11}$$

Recall measures how many of the true matches have been classified correctly and is calculated as:

$$R = \frac{TP}{TP + FN} \tag{2.12}$$

The F1 score, also known as f-score or f-measure, is the harmonic mean of precision and recall and is calculated as:

$$F1 = \frac{2 \times P \times R}{P + R} \tag{2.13}$$

The harmonic mean is preferred against the arithmetic mean because in many applications, it is not sufficient if only one of the involved metrics scores high, which could potentially balance the arithmetic mean at a good level [Naumann and Herschel, 2010]. While accuracy is useful for tasks where the classes are balanced, i.e. the number of matching and non-matching pairs in the gold standard is similar, precision, recall, and F1 score do not include the number of true negatives in their calculations. Therefore they are suitable metrics in the case that the gold standard is imbalanced, which often occurs for entity resolution tasks as naturally there exist many more non-matching than matching record pairs.

Chapter 3

Profiling Entity Resolution Benchmark Tasks

Entity resolution has been studied over a long time [Fellegi and Sunter, 1969] and is the focus of many research works [Christen, 2012; Papadakis et al., 2021]. To evaluate and compare different entity resolution methods, a wide range of entity resolution benchmark tasks has been developed and made publicly available [Draisbach and Naumann, 2010; Konda et al., 2016; Köpcke et al., 2010]. Such tasks consist of the following artifacts: (i) one or more data sources with records describing real-world objects and (ii) a set of correspondences stating for all or for a subset of all record pairs whether they describe the same real-world object (matches) or different real-world objects (non-matches). Figure 3.1 illustrates the artifacts of an entity resolution task along with the example of two data sources containing phone product records. In our work, entity resolution is treated as a supervised binary classification problem in which a labeled set of matching and non-matching pairs is needed for training a machine learning model, similarly to many related works [Christen, 2012; Christophides et al., 2020; Halevy et al., 2006]. To enable the comparison and evaluation of supervised entity resolution methods, the correspondence set of the benchmark tasks is split into training and test sets [Han et al., 2011].

Despite the availability of a large number of benchmark tasks for entity resolution, there is no universal entity resolution method excelling in all of them. In contrast, different methods have been shown to perform better than others, given the characteristics of the task [Köpcke et al., 2010; Mudgal et al., 2018]. Therefore, understanding the difficulty and diversity of benchmark tasks is essential for selecting appropriate entity resolution methods, assessing their strengths and weaknesses as well as comparing their performance. The latter relies on exactly defined sets of matching and non-matching record pairs, as well as fixed splits of the correspondence sets into training and test sets when it comes to supervised entity resolution methods. However, these sets are not always provided for some widely used benchmark tasks.

Data source A			
ID	name	price	brand
A1	i-phone 4s	200€	apple
A2	htc one m9	220€	htc

Data source B			
ID	name	price	brand
B1	iphone 4	190€	apple
B2	one m9	210€	htc

Correspondences	
A1-B1	non-match
A2-B2	match
A1-B2	non-match

Figure 3.1: Artifacts of an example entity resolution task.

This chapter aims first to enable the comparison of supervised entity resolution methods on clearly defined grounds and second to systematically explore the challenges associated with entity resolution tasks having different profiling characteristics, covering the first contribution of this thesis [C1]. To achieve both goals, we complement, profile, and compare 21 entity resolution benchmark tasks. To enable the exact reproducibility of evaluation results and a clear comparison of entity resolution methods, we employ a heuristic and complement the tasks that do not have fixed sets of non-matching pairs, as well as fixed training and test splits. To better understand the specific challenges associated with different tasks, we define a set of profiling dimensions that capture central aspects of the entity resolution tasks. The profiling dimensions capture properties of the records to be matched as well as of the record pairs in the correspondence set and thus go beyond the dimensions used in related work for categorizing entity resolution tasks, which solely focus on the characteristics of the records of the data sources [Mudgal et al., 2018]. Using the proposed profiling dimensions, we create groups of benchmark tasks having similar characteristics. Additionally, we assess the difficulty of each group by computing baseline evaluation results with standard symbolic record pair comparison techniques and two common classification methods.

The contributions of this chapter are summarized as follows:

- We define a set of dimensions for profiling entity resolution tasks that captures characteristics of the correspondence set of record pairs in addition to the characteristics of the records of the data sources to be matched.
- We create groups of benchmark tasks having similar characteristics and associated challenges.
- We complement existing benchmark tasks by adding fixed splits of matching and non-matching record pairs in order to support the reproducibility and comparability of the results of different entity resolution methods.
- We evaluate the difficulty of 21 benchmark tasks by establishing baseline evaluation results.

This chapter is structured into six sections. Section 3.1 discusses the related work in the areas of profiling entity resolution tasks, as well as reproducing results of entity resolution methods. Section 3.2 introduces the 21 benchmark entity resolution tasks used in our analysis. In Section 3.3, we present the heuristic for complementing the incomplete benchmark tasks, while in Section 3.4, we present the proposed dimensions for profiling and grouping entity resolution tasks. In Section 3.5, we report the baseline results for each task. Finally, Section 3.6 summarizes the main findings and contributions of the chapter.

The methodology and results of this chapter have been published in the Proceedings of the 29th International Conference on Knowledge Management [Primpeli and Bizer, 2020]. All complemented benchmark tasks, together with the code for profiling and evaluating them, are available for public download.[1]

3.1 Related Work

Data Profiling Data profiling is the "activity of creating small but informative summaries of a database" [Johnson, 2009]. It is an essential task for many different use-cases, such as data exploration, data cleansing, and data integration [Abedjan et al., 2015], which goes hand in hand with the entity resolution task. Understanding the profile of the data to be integrated contributes to estimating the schematic fit, i.e. the overlap degree of schemata, and the data fit, i.e. the overlap degree of data objects among different data sources. Data profiling of relational data has moved from single column-related metadata, such as value counts, density, and value ranges, to detecting dependencies among different columns, such as foreign key discovery and functional dependencies [Naumann, 2014].

Profiling of ER Tasks There are many survey articles comparing entity resolution methods utilizing benchmark tasks [Christophides et al., 2020; Köpcke et al., 2010], but there are hardly any works comparing and categorizing entity resolution benchmark tasks. To the best of our knowledge, the work of Mudgal et al. [2018] is the only one that categorizes entity resolution tasks under the following three groups: structured, textual, and dirty. The three groups are loosely defined as follows: structured tasks are entity resolution tasks with short and atomic attribute values, textual tasks contain attributes with longer textual values, such as product descriptions, and dirty tasks are structured tasks with misplaced attribute values or missing values. Therefore, the categorization of an entity resolution task into one of those groups is conducted on the basis of the number of attributes, the length, and the presence or absence of artificial noise in the attribute values, without considering any properties of the correspondence set.

Reproducibility and Comparability of Evaluation Results The comparison and reproducibility of research results on clear grounds is an important issue across

[1] http://data.dws.informatik.uni-mannheim.de/benchmarkmatchingtasks/index.html

different research communities. Even in the case that the same benchmark tasks are used to evaluate different methods, the evaluation metrics or the train and test splits might differ [Barlaug and Gulla, 2021]. To circumvent this problem and reach safe comparison conclusions, researchers often need to re-implement other methods, which is a tedious task [Barlaug and Gulla, 2021]. Therefore, initiatives such as SIGMOD reproducibility[2] and Papers with Code[3] invite researchers to make their experiments repeatable by sharing all relevant artifacts. In addition, there exist multiple campaigns that contribute to the meaningful evaluation of different systems, such as the SemEval workshop[4] with a focus on NLP tasks, the SemWebEval, and the OAEI contest[5], both aiming at evaluating tasks relevant to the Semantic Web community. Such campaigns provide benchmarks and fixed evaluation procedures. The OAEI contest publishes, among others, matching tasks on both schema and instance level for evaluating ontology matching systems. These tasks use RDF data that include class and property hierarchies, which should be considered by successful matching systems. This differs substantially from the benchmark tasks that we consider in this chapter, which contain relational data sources with aligned schemata. For tasks that address both schema and instance matching, different profiling dimensions become relevant. These have been analyzed in the work of Daskalaki et al. [2016]. Examples of relevant profiling dimensions for such tasks are the schema similarity of the data sources and the data source creation method, i.e. whether the data are real or synthetic [Daskalaki et al., 2016].

3.2 Benchmark Tasks

Various repositories provide entity resolution benchmark tasks for public download in an effort to facilitate the evaluation, reproducibility, and comparability of different entity resolution methods. The Database Group of the University of Leipzig[6] published several benchmark tasks in 2010, which are widely used by the community since then. The Magellan repository[7] has been maintained since 2015 by the Data Management Research Group of the University of Wisconsin-Madison. It includes a large collection of benchmark tasks, a subset of which has been created by the same research group, while some tasks have been collected from other repositories. The DuDe repository of the Hasso Plattner Institute[8] provides three benchmark tasks, originally published by other sources, which have been modified to better serve the entity resolution setting [Draisbach and Naumann, 2010]. Fi-

[2] http://db-reproducibility.seas.harvard.edu/

[3] https://paperswithcode.com/

[4] http://alt.qcri.org/semeval2020/

[5] http://oaei.ontologymatching.org

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

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

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

nally, the Web Data Commons (WDC) project, which is maintained by the Data and Web Science Group of the University of Mannheim, has published several e-commerce-related benchmark tasks.[9,10] The WDC tasks include product-specific training sets, that are derived using distant supervision from the Web while the test sets are manually verified. More details on the curation of the WDC entity resolution benchmark tasks using distant supervision from the Web will be provided in Chapter 5.

We collect and profile 21 benchmark tasks from these four repositories. Table 3.1 provides information about the selected tasks. We report basic profiling information: (i) the number of data sources from which the records originate, (ii) the number of records in the data sources, indicated with DS1 and DS2 for two-source tasks otherwise we report the overall number of records, (iii) the number of matching and non-matching record pairs in the correspondence set, (iv) if the task is complete, i.e. contains fixed splits of matching and non-matching record pairs, (v) the number of attributes of specific data types, and (vi) the average density (ratio of non-null values to all values) of the attributes of all records appearing in the correspondence set. We consider an attribute to be of data type long string if the average length of its values exceeds six words, similar to the Magellan attribute data type detection system.[11] As shown in Table 3.1, the tasks are diverse and include data sources of different sizes, amounts of attributes, density as well as attribute data types. Most benchmark tasks contain records originating from two data sources. In contrast, the tasks provided by the Web Data Commons Product repositories include records from up to 269 data sources.

Six of the tasks provide the complete mapping (i.e. all matching record pairs) while no explicit non-matching pairs are offered. For the rest of the tasks, a subset of both matching and non-matching record pairs is included in the set of correspondences. For the cases where the complete mapping is provided, non-matching pairs can be generated by calculating the Cartesian product of all records and excluding the matching pairs. Given the size of the data sources, this often results in large amounts of non-matching pairs and thus motivates the usage of blocking techniques [Ebraheem et al., 2018; Papadakis et al., 2020] to remove obvious non-matches which are not helpful for training and are uninteresting for testing. As the benchmark tasks only define matches, different researchers who use these tasks generate different sets of non-matches. This influences the model training [Brunner and Stockinger, 2020; Li et al., 2020; Mudgal et al., 2018]. Given that not all of those works publish the complemented sets used for their experiments, it is not possible to exactly reproduce the evaluation of the methods.

The WDC tasks are published together with fixed training, validation and test splits of matching and non-matching record pairs. The repositories hosting the rest of the tasks do not provide fixed splits, while split sets for three of the eight pre-

[9] http://webdatacommons.org/productcorpus/index.html

[10] http://webdatacommons.org/largescaleproductcorpus/v2/index.html

[11] https://github.com/anhaidgroup/py_entitymatching In addition to the cut at six words, Magellan differentiates string values exceeding ten words.

sented Magellan benchmark tasks are provided in the DeepMatcher GitHub repository.[12] Researchers using the incomplete benchmark tasks thus split the correspondences into training and test sets by themselves, often without providing the details required for reproducing the splits in their papers, such as the sampling tool and random seeds. This hinders the reproducibility of the experimental results.

3.3 Task Completion

In order to exactly reproduce evaluation results, the correspondence set must be split into a fixed training and test set of matches and non-matches. This is especially necessary for tasks with small-sized correspondence sets. In this case, the model can overfit the small training set [Han et al., 2011]. Additionally, its evaluation can be misleading and not show the real prediction quality of the model, as it only depends on the few record pairs of the test set. However, as explained in the previous section and shown in Table 3.1, most benchmark tasks do not provide fixed splits.

We complement the correspondence sets of the benchmark tasks that provide a complete mapping but do not provide non-matches using the following heuristic which relies on canopy clustering [Christen, 2012] and random sampling. First, we construct all non-matching record pairs by calculating the Cartesian product of all records and excluding the pairs appearing in the complete mapping. We restrict the amount of non-matching pairs to allow for a reasonable duration of record pair comparison and training by blocking. We generate blocking keys using the domain-dependent label of the records, e.g. title for records describing books or product name for records describing products, and apply relaxed Jaccard with inner Levenshtein distance and a threshold of 0.2 as the equality function for generating the blocks. We select all non-matching pairs within each block. To avoid selection bias, we add randomly selected non-matching pairs, the amount of which equals 25% of the amount of non-matching pairs sampled within the blocks. To enhance the diversity of the correspondence set, we allow each record to appear in at most ten non-matching record pairs. Finally, we apply stratified sampling and split the correspondence sets of the tasks that do not provide fixed splits into test and training sets. The latter is further split to curate a validation set, often used for tuning the parameters of machine learning models [Han et al., 2011]. The splitting is performed on correspondence level rather than on entity level. This means that although the record pairs uniquely appear in the split sets, e.g. no record pair from the training set is in either the test or the validation set, this does not apply for single records. For example, the training set can include a record pair (r_1, r_2, match), while the record pair $(r_1, r_3, \text{non-match})$ might appear in the validation or test set. The record pairs of the correspondence set are distributed as follows: training set 70%, validation set 20%, and test set 10%.

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

Table 3.1: Overview of the 21 entity resolution benchmark tasks.

Task	#Data sources	#Records DS1	#Records DS2	#Matches	#Non-matches	Complete	Attributes #Short str.	Attributes #Long str.	Attributes #Num.	Attributes Density
Leipzig Database Group										
abt-buy	2	1,081	1,092	1,095	-		0	2	1	0.63
amazon-google	2	1,363	3,226	1,298	-		1	2	1	0.75
dblp-acm	2	2,614	2,294	2,223	-		1	2	1	1.00
dblp-scholar	2	2,616	64,263	5,346	-		1	2	1	0.81
DuDe										
$\text{restaurants}_{\text{Fodors-Zagats}}$	2	533	331	112	-		5	0	0	1.00
cora	1	1,879		64,578	268,082		11	6	1	0.31
Magellan										
$\text{products}_{\text{Walmart-Amazon}}$	2	2,554	22,074	1,154	-		6	3	1	0.84
baby products	2	5,085	10,718	108	292		7	2	7	0.42
beer	2	4,345	3,000	68	382		4	0	0	0.96
bikes	2	4,785	9,002	130	320		5	0	3	0.78
$\text{books}_{\text{Goodreads-Barnes}}$	2	3,967	3,700	92	305		6	1	3	0.53
cosmetics	2	6,443	11,026	128	280		2	1	0	0.94
$\text{music}_{\text{iTunes - Amazon}}$	2	6,906	55,932	132	407		7	0	0	0.99
$\text{restaurants}_{\text{Yellow - Yelp}}$	2	5,223	11,840	130	270		5	0	1	1.00
Web Data Commons										
phones	17	447	50	258	22,092		22	1	3	0.25
headphones	6	444	51	226	22,418		21	3	3	0.13
tvs	8	428	60	182	25,499		49	9	3	0.07
xlarge_cameras	269	3,665		7,478	35,899	✓	1	3	0	0.51
xlarge_watches	190	4,068		9,564	53,105	✓	1	3	0	0.43
xlarge_computers	235	4,676		9,991	59,571	✓	1	3	0	0.50
xlarge_shoes	120	2,808		4,440	39,088	✓	1	3	0	0.41

3.4 Task Profiling

This section introduces the profiling dimensions that we use to analyze the tasks and describes our strategy for selecting attributes relevant to solving the tasks. Afterward, we present the profiling results for the 21 tasks and create groups of tasks having similar characteristics.

3.4.1 Relevant Attributes

Attributes that do not contribute to the solution of an entity resolution task are not relevant for understanding the task-related challenges and, therefore, should be excluded from the profiling.

We identify relevant attributes by learning a random forest and selecting attributes based on the importance of their corresponding features resulting from the trained model, similar to a wrapper-based feature extraction method [Khalid et al., 2014]. In order to ensure that the different structural and formatting variations of the input data sources do not affect the selection of relevant attributes, we first normalize the attribute values following standard pre-processing steps as described in Section 2.3.1: the attributes of the different sources are aligned with respect to their schema, translated to a common language, lowercased and deprived of stopwords. To generate the features, we calculate various pairwise record similarity scores for each attribute using a set of data type specific similarity metrics, as introduced in Section 2.3.3. All details about the record pair comparison step will be provided in Section 3.5.1. The feature selection method starts by fitting a random forest classifier to the complete feature vector, i.e. all generated features N, and retrieving the weights of the features, i.e. the feature importances assigned by the trained classifier. The random forest classifier is evaluated using four-fold cross-validation. The score of this evaluation $F1_n$ is the F1 score that can be reached when training the model on the features generated by all attributes. Next, the N features are sorted in descending order given their weights. We iterate over the N sorted features and in each iteration $m \in [1, N]$, we reduce the feature vector to top m features. The reduced feature vector is used for training a new random forest classifier, which is evaluated with $F1_m$. We stop iterating once the model learned using a reduced feature vector reaches the quality of the model learned on the complete feature set, i.e. $F1_m >= F1_n$. Algorithm 1 gives the pseudo-code of our feature selection strategy.

We identify relevant attributes by projecting the relevant features to single record attributes, e.g. *title_Levenshtein* to *title*. In contrast to key detection [Naumann, 2014], which finds the combinations of attributes that uniquely identify records in a data source, our attribute selection strategy finds attributes whose values, when compared pairwise with one or more similarity metrics, help reveal if a pair of records is matching or non-matching.

The coverage of models, learned with different combinations of attributes, can significantly vary as not all attributes contribute equally to the solution of the en-

tity resolution task [Petrovski and Bizer, 2020]. Discovering the set of attributes that encode the most-identifying information is crucial for extracting more focused profiling meta-information. We define these attributes as *top relevant attributes* and approximate their calculation using the following simple rule: the *top relevant attributes* are the attributes that, when used for pairwise feature generation and model learning, make up for 95% of the maximum F1 score, $F1_n$. Therefore, we use the same feature selection strategy as described by Algorithm 1 with the only difference that the selection condition (line 8 in the pseudo-code) is now $F1_m >= 0.95 \times F1_n$.

Algorithm 1 Feature selection algorithm

Input: $D_n : R \times n$ matrix, $n > 0$
Output: $D' : R \times m$ matrix, $0 < m <= n$

1: $model_n = RandomForest(D_n)$
2: $F1_n = cross\_validate(model_n, D_n)$
3: $sorted\_features = sort\_desc(model_n.feature\_weights)$
4: **for** $m = 1$ *to* n **do**
5: $D' \leftarrow D_n[r, sorted\_features.top(m)]$
6: $model_m = RandomForest(D')$
7: $F1_m = cross\_validate(model_m, D')$
8: **if** $F1_m >= F1_n$ **then**
9: **break**
10: **end if**
11: **end for**
12: **return** D'

3.4.2 Profiling Dimensions

We define five profiling dimensions that evolve around specific matching-related challenges: schema complexity, textuality, sparsity, corner cases, and development set size. In the following, we present the motivation behind each profiling dimension and explain how it is calculated.

Schema Complexity (SC)

This dimension refers to the number of attributes that contribute to solving an entity resolution task. A high schema complexity suggests a larger amount of underlying matching patterns and thus might be harder for a learner to solve. We approximate schema complexity by the number of relevant attributes of an entity resolution task, calculated with the heuristic, introduced in Section 3.4.1.

Textuality (TX)

Attribute values that consist of long sequences of words, e.g. the title of a product offer in e-commerce, often contain information that could also have been represented using multiple attributes. Such attributes are challenging for entity resolution methods as they need to either separate the attributes in pre-processing or apply similarity metrics that can deal with below 1st normal form values, i.e. each attribute holds an atomic value that cannot be further split into smaller parts. We calculate the textuality of an entity resolution task as the average length value in words split by white space, of the top relevant attributes of the records appearing in the set of correspondences.

Sparsity (SP)

Having missing values in a supervised learning setting is a well-recognized challenge for machine learning. In a classification setting with a non-dense feature vector, the learning algorithm has to learn multiple classification functions, with each function covering a subset of the data points having the same missing features [Goodfellow et al., 2016]. We calculate sparsity as the ratio of missing attribute values to all attribute values of the relevant attributes of records that appear in the correspondence set of the entity resolution task.

Corner Cases (CC)

The dimension of corner cases aims to capture the number of record pairs that are non-matching, but their attribute values are similar and the ones that have attribute values of low similarity but are matching. This dimension is crucial for understanding the difficulty of an entity resolution task. In a trivial entity resolution setting with no corner cases, the matching record pairs lie far from the non-matching record pairs in the hypersurface of the vector space and can thus be perfectly separated with a decision boundary, e.g. defined by a linear function or a decision tree. A less trivial entity resolution task includes record pairs that are close to the decision boundary. For these corner cases, the learning algorithm needs to make further adjustments.

We use this observation and the simple case of a linear classifier for approximating the number of corner cases of an entity resolution task, which we compute with the following heuristic: for every record pair in the correspondence set, we calculate an aggregated similarity score by averaging the pairwise similarity score-based feature values, the computation of which is detailed in section 3.5.1, of the top relevant attributes. Feature values that cannot be computed because of missing corresponding attribute values of one or both records of the pair are excluded from this aggregation. With every record pair being represented by its aggregated similarity score, we iteratively search in steps of 0.01 for the threshold value that can best separate the matching from the non-matching pairs, i.e. maximizes the F1 score. Once the optimal threshold is found, we measure the number of match-

ing pairs lying in the non-matching zone (false negatives) and the amount of non-matching pairs lying in the matching zone (false positives). To account for the class imbalance problem in entity resolution, the corner cases dimension is calculated as the ratio of corner cases in relation to the amount of matching pairs: $\frac{\#false\_positives+\#false\_negatives}{\#matching\_pairs}$.

Development Set Size (DS)

The difficulty of an entity resolution task also depends on the number of correspondences that are available for learning and selecting matching models. This is an instantiation of the general observation that the performance of a classifier is affected by the size of the training set, while small sample sizes tend to cause classification models to overfit the data used for training and hyperparameter optimization [Anthony and Biggs, 1997]. We calculate the development set size as the amount of matching and non-matching record pairs in the training and validation sets of an entity resolution task.

3.4.3 Profiling and Grouping Benchmark Tasks

We identify the relevant attributes, calculate the values of the five profiling dimensions for each task, and create groups of tasks having similar characteristics. Table 3.2 presents the grouped benchmark tasks along with the attributes that are relevant for entity resolution, the ratio of relevant attributes to all attributes, as well as the values of the five profiling dimensions schema complexity (SC), textuality (TX), sparsity (SP), corner cases (CC) and development set size (DS). The relevant attributes are listed in descending order considering their corresponding feature importances. The inner brackets indicate the top relevant attributes.

Looking at the relevant attributes, we observe that in only 6 out of the 21 tasks, all attributes were selected as relevant, using the selection strategy described in Section 3.4.1. For the remaining 15 tasks, a subset of the attributes is sufficient for reaching the same F1 score, which would result from using all attributes.

Small development sets can lead to models that overfit the data. This is relevant for the bikes, $\text{books}_{\text{Goodreads-Barnes}}$, and cosmetics tasks. For these tasks, more identifying attributes, such as the book title and the bike model name, receive a lower weight compared to less identifying ones, such as the number of pages and the bike color.

The profiling results show patterns, which we use to create five groups of entity resolution tasks. The grouping can help researchers to select interesting tasks for evaluating entity resolution methods and to better understand and interpret their results, i.e. which challenges an entity resolution method can successfully deal with. In the following, we present the five groups.

Table 3.2: Relevant attributes and profiling results.

The inner brackets indicate in the *Relevant attributes* column indicate the top relevant attributes.
SC = schema complexity, TX = textuality, SP = sparsity, CC = corner cases, DS = development set size

Task	Relevant attributes		Profiling dimensions				
		% of all	SC	TX	SP	CC	DS
Group 1: Dense data, simple schema							
beer	[[beer_name], brew_factory, ABV, style]	1.00	4	5.32	0.06	0.32	405
bikes	[[color, bike_name], price, km_driven]	0.44	4	5.37	0.00	0.09	405
$\text{music}_{\text{iTunes-Amazon}}$	[[song_name, time], album_name]	0.42	3	6.6	0.00	0.08	485
$\text{restaurants}_{\text{Yellow-Yelp}}$	[[phone], name, address]	0.50	3	2	0.00	0.05	360
$\text{restaurants}_{\text{Fodors-Zagats}}$	[[phone], name, address]	0.60	3	1.02	0.00	0.03	600
Group 2: Sparse data, complex schema							
phones	[[phone_type,memory, display_size], color, mpn, ..]	0.38	10	4.8	0.39	0.42	20,137
headphones	[[model], mpn, impedance, sensitivity, accessories]	0.18	5	1.5	0.63	0.23	20,401
tvs	[[model], mpn, weight, height, width]	0.11	5	1.04	0.64	0.36	23,138
Group 3: Small and difficult							
baby products	[[title, ext_id, SKU],colors, category, ..]	0.50	8	9.2	0.32	0.81	360
$\text{books}_{\text{Goodreads-Barnes}}$	[[pagecount, title], publisher, ISBN13, format, ..]	0.70	7	7.86	0.24	0.53	357
cosmetics	[[color, description], price]	1.00	3	9.36	0.07	0.27	367
Group 4: Textual data, few corner cases							
dblp-acm	[[title], year, authors]	0.75	3	7.65	0.00	0.02	42,079
dblp-scholar	[[title], authors]	0.50	2	7.8	0.02	0.07	74,689
$\text{products}_{\text{Walmart-Amazon}}$	[[title, modelno], long_description, brand, price]	0.50	5	10.52	0.08	0.27	14,036
cora	[[authors, volume, pages, year, title] , ..]	0.38	7	15.97	0.26	0.23	299,726
Group 5: Textual data, many corner cases							
abt-buy	[[name], description, price]	1.00	3	8.69	0.23	0.69	6,452
amazon-google	[[name, price, description]]	0.75	3	130.22	0.03	0.68	7,604
xlarge_cameras	[[title, description], brand, specTable]	1.00	4	98.71	0.38	0.78	42,277
xlarge_watches	[[title, description], specTable, brand]	1.00	4	109.79	0.43	0.64	61,569
xlarge_computers	[[title, description], brand, specTable]	1.00	4	51.69	0.44	0.86	68,462
xlarge_shoes	[[title, description]]	0.50	2	80.85	0.17	0.98	42,429

Group 1: Dense Data, Simple Schema

This group comprises entity resolution tasks with low schema complexity ($\leqslant 4$ relevant attributes), high density (> 0.94), and short attribute values (< 7 words). The following benchmark tasks have dense data and simple schema: bikes, beer, $\text{restaurants}_{\text{Fodors-Zagats}}$, restaurants $_{\text{Yellow-Yelp}}$ and $\text{music}_{\text{iTunes-Amazon}}$. These tasks are expected to be easy to solve for symbolic entity resolution methods.

Group 2: Sparse Data, Complex Schema

In group 2 belong entity resolution tasks that have non-dense attributes (< 0.60) with short values (< 5 words) and high schema complexity ($\geqslant 5$ relevant attributes). Under this group fall the following tasks: phones, headphones, and tvs. The matching methods used for evaluating these tasks need to especially address the challenge of low data density [Petrovski and Bizer, 2020].

Group 3: Small and Difficult

In this group belong the entity resolution tasks that have challenging characteristics, like high schema complexity and textuality, but only provide a small number of matches and non-matches, which a classifier can use for learning and optimizing the hyperparameters. The small size of the training set makes it hard for classifiers to adapt to the different challenges and may lead to overfitting. The benchmark tasks falling into this group are $\text{books}_{\text{Goodreads-Barnes}}$, cosmetics, and baby products.

Group 4: Textual Data, Few Corner Cases

This group contains entity resolution tasks with textual relevant attributes (> 7 words) and a low to very low containment of corner cases ($\leqslant 0.27$). The challenge imposed by the textuality dimension becomes trivial in the absence of corner cases. The benchmark entity resolution tasks that fall under this category are $\text{products}_{\text{Walmart-Amazon}}$, dblp-acm, dblp-scholar, and cora. Subsymbolic entity resolution methods are known to outperform symbolic ones on highly textual tasks due to their ability to automatically generate highly dimensional features [Mudgal et al., 2018]. Nonetheless, it is hard for subsymbolic entity resolution methods to show their strengths, given textual tasks with a low containment in corner cases.

Group 5: Textual Data, Many Corner Cases

This group contains entity resolution tasks with high textuality and many corner cases. The benchmark tasks that can be categorized under this group are xlarge_cameras, xlarge_watches, xlarge_computers, xlarge_shoes, amazon-google, and abt-buy. For these tasks, subsymbolic entity resolution methods have been shown to achieve better results than methods relying on symbolic features [Mudgal et al., 2018; Peeters et al., 2020a,b].

Table 3.3: Parameter ranges of the grid search.

RF	estimators	max. depth	min. leaf size
	[10, 100, 500]	[5, 10, 50, None]	[1, 3, 5]
SVM	C	gamma	kernel
	log(-2, 5), $\alpha = 10$	$[1^{-6}, 1^{-4}, 1^{-2}, 1, 10]$	[rbf, linear*]

3.5 Baseline Evaluation

Considering baseline results is crucial for judging the difficulty of benchmark tasks. Baseline results set the lower limit of the predictive quality that a new entity resolution method needs to achieve on a specific task and indicates if there is room for improvement for more sophisticated methods. In this section, we describe the entity resolution methods that we employ for calculating baseline results. Afterward, we apply the methods to the 21 benchmark tasks, present the baseline results, and discuss the difficulty of the tasks in relation to the task groups.

3.5.1 Record Pair Comparison

We perform record pair comparison using symbolic features, which we calculate with data type specific similarity metrics, as presented in Section 2.3.3 and similarly to the Magellan matching system [Konda et al., 2016]. As a first step, we need to detect the data type of each attribute. Our data type detection heuristic distinguishes between three data types: short string, long string, and numeric. The long string data type is assigned to those attributes whose values have an average length larger than six words. If more than one data type is detected for the same attribute, we assign long string, if long string appears in the list of detected data types. Otherwise, we assign short string.

Given the detected data type, multiple data type specific similarity metrics are used for constructing the feature vector. Feature values of data type short string are compared with the following similarity metrics: Levenshtein, Jaccard on the token level, Jaccard with inner Levenshtein, exact similarity, and containment similarity. The containment similarity is calculated on word-level for any string value with more than one word. Otherwise, it is calculated on the token level. For long strings, the similarity metrics used for short strings are applied while Jaccard is calculated on the word level and additionally the cosine similarity with tf-idf weighting is computed. For numeric attributes, the absolute difference is computed. In addition to the attribute-specific similarity scores, we also concatenate all attribute values of a record and calculate an overall similarity score using cosine similarity with tf-idf weighting over the concatenated values.

All similarity scores are scaled to the range of [0,1]. In the case that the value of an attribute is missing for one or for both records of the record pair, we assign the out-of-range score -1. This allows any classifier to consider all record pairs without dropping or replacing the missing values.

3.5.2 Record Pair Classification

We employ two widely-used supervised classification models for learning baseline matchers: support vector machines (SVM) and random forests (RF). We optimize the hyperparameters shown in Table 3.3 using grid search. The asterisk indicates that the linear kernel has not been used in combination with all other parameter values in the grid search like the rbf kernel, but only in one setting with default C and gamma values. For the non-optimized parameters, we use the default values of the python scikit-learn library, version 0.22.1. Finally, we use fixed random seeds for allowing the reproducibility of the baseline results by setting the random_state parameter of the scikit-learn classification models to 1.

3.5.3 Baseline Results

We evaluate the baseline entity resolution methods with split validation and calculate precision, recall, and F1 score on the positive class (matching). We present the baseline results in Table 3.4. In addition, we show the best reported result found in related work from supervised entity resolution methods. Entity resolution methods applying other types of learning, e.g. active learning or semi-supervised learning, are excluded from this comparison [Kasai et al., 2019; Papadakis et al., 2020; Primpeli and Bizer, 2019]. The interpretation of the comparison to the results reported in related work should be made with attention to the differing, unfixed train, validation, and test sets.

Comparing the results of the two classifiers, we see that the SVM classifier performs similar (6 tasks) or worse (11 tasks) than the random forest classifier for 17 out of 21 tasks. However, this is not the case for the xlarge_shoes, baby products, cosmetics, and $\text{books}_{\text{Goodreads-Barnes}}$ tasks, for which the SVM outperforms the random forest by 0.04 to 0.09 in F1 score. In addition, six benchmark tasks are perfectly solved by the baseline method (F1 score=1.00), indicating that there is no room for improvement for more advanced methods.

Considering the grouping of the tasks, we can see that each profiling group imposes varying difficulty levels on our baseline method. More concretely, F1 scores between 0.92 and 1.00 are achieved for the tasks belonging to *Group 1: Dense Data, Simple Schema*. For solving the tasks of this group, only a small amount of attributes needs to be considered by the learning algorithm. The matching patterns are simple, which we verify by looking at the average depth of the trees of the best random forest estimator per task, ranging between 4.5 and 4.98. On the other hand, the tasks of *Group 2: Sparse Data, Complex Schema* are more difficult to solve, with the F1 scores being lower than the F1 scores reported for the tasks of Group 1, as a result of the higher sparsity and schema complexity. These characteristics require the machine learning model to learn complex matching patterns. The average depth of the trees of the best random forest estimators for the tasks of Group 2 is 10.

Table 3.4: Baseline results and comparison to related work.

Profiling group	Task	SVM			Random forest			Δ_{RF-SVM}	Best F1 score result in related work
		P	R	F1	P	R	F1		
Group 1: Dense data, simple schema	beer	1.00	0.86	0.92	1.00	1.00	1.00	+0.08	0.78 [Mudgal et al., 2018]
	bikes	0.92	0.92	0.92	0.92	0.92	0.92	0.00	-
	$\text{music}_{\text{iTunes-Amazon}}$	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.94 [Li et al., 2020]
	$\text{restaurants}_{\text{Yellow -Yelp}}$	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00 [Mudgal et al., 2018]
	$\text{restaurants}_{\text{Fodors-Zagats}}$	1.00	0.91	0.95	1.00	1.00	1.00	+0.05	0.97 [Li et al., 2020]
Group 2: Sparse data, complex schema	phones	0.85	0.88	0.86	0.85	0.88	0.86	0.00	0.84 [Petrovski and Bizer, 2020]
	headphones	0.98	0.93	0.95	0.98	0.96	0.97	+0.02	0.94 [Petrovski and Bizer, 2020]
	tvs	0.93	0.78	0.85	0.94	0.89	0.91	+0.06	0.83 [Petrovski and Bizer, 2020]
Group 3: Small and difficult	baby products	0.70	0.64	0.67	0.68	0.55	0.63	-0.04	-
	$\text{books}_{\text{Goodreads-Barnes}}$	0.80	0.89	0.84	0.73	0.89	0.80	-0.04	-
	cosmetics	1.00	0.77	0.87	0.90	0.69	0.78	-0.09	-
Group 4: Textual data, few corner cases	dblp-acm	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.98 [Mudgal et al., 2018]
	dblp-scholar	0.99	0.99	0.99	0.99	0.99	0.99	0.00	0.94 [Mudgal et al., 2018]
	$\text{products}_{\text{Walmart-Amazon}}$	0.97	0.87	0.92	0.93	0.89	0.93	+0.01	0.89 [Gokhale et al., 2014]
	cora	0.99	0.98	0.99	1.00	1.00	1.00	+0.01	1.00 [Wang et al., 2011]
Group 5: Textual data, many corner cases	abt-buy	0.96	0.71	0.81	0.95	0.77	0.85	+0.04	0.91 [Liu et al., 2019] reported in [Peeters and Bizer, 2021]
	amazon-google	0.83	0.68	0.74	0.80	0.69	0.75	+0.01	0.71 [Li et al., 2020]
	xlarge_cameras	0.71	0.61	0.65	0.75	0.67	0.71	+0.06	0.98 [Peeters and Bizer, 2021]
	xlarge_watches	0.86	0.71	0.78	0.82	0.73	0.81	+0.03	0.97 [Peeters and Bizer, 2021]
	xlarge_computers	0.74	0.67	0.70	0.78	0.78	0.78	+0.08	0.97 [Peeters and Bizer, 2021]
	xlarge_shoes	0.83	0.43	0.57	0.82	0.38	0.52	-0.05	0.97 [Peeters and Bizer, 2021]

Our baseline methods perform poorly for the tasks of *Group 3: Small and Difficult*, with F1 scores ranging between 0.67 and 0.87. We assume the reason for the poor performance to be, on the one hand, the inherent challenging characteristics of the tasks, such as high textuality, a significant amount of corner cases, and high schema complexity, and on the other hand, the small amount of training data. These factors likely cause the model to overfit the training data and not generalize well on the test set.

The F1 scores achieved using the random forest classifier for the tasks of *Group 4: Textual Data, Few Corner Cases* are above 0.93. This indicates that they have a low level of difficulty despite their high textuality. The small amount of corner cases makes the tasks trivial. For each of the tasks dblp-acm and dbpl-scholar, which contain almost no corner cases, there is only one top relevant feature: *title_cosine_tfidf* for the dblp-acm task and *title_relaxed_jaccard* for the dblp-scholar. This indicates that condensing the title attribute into a single similarity score is enough for solving both tasks. The other tasks of Group 4 contain a few corner cases but have the following characteristics, which we hypothesize help the model to generalize well and achieve good performance: the cora task has a very large development set and the $\text{products}_{\text{Walmart-Amazon}}$ task has a model number attribute, which is highly identifying for resolving product records.

The performance of our baseline method drops for the entity resolution tasks of *Group 5: Textual Data, Many Corner Cases* and ranges between 0.57 and 0.85. These tasks have many corner cases and highly textual relevant attributes, such as product titles and descriptions. The weak performance can be explained by the inability of the similarity-based feature vector to adequately summarize the textual data. In contrast to our symbolic, similarity-based features, the related works, reporting significantly higher F1 scores on these tasks, use subsymbolic entity resolution methods, which have been shown to perform better on textual data [Li et al., 2020; Mudgal et al., 2018; Peeters and Bizer, 2021].

3.6 Discussion and Conclusion

In this chapter, we selected 21 benchmark entity resolution tasks and identified that 17 of them were incomplete with regard to fixed sets of training, validation, and test record pairs. We complemented the incomplete tasks in order to facilitate the reproducibility and comparability of entity resolution methods.

Additionally, we proposed a heuristic for extracting attributes relevant for solving entity resolution tasks as well as five dimensions for profiling the tasks. The profiling dimensions evolve around specific challenges of entity resolution tasks and go beyond the dimensions of textuality and structuredness, which have been used so far in related work [Mudgal et al., 2018]. Compared to the profiling dimensions proposed by Naumann [2014], which focus on profiling single data sources, our suggested profiling dimensions capture properties of both the data sources and the correspondence set of the entity resolution task. We used these dimensions to

profile and group the collection of 21 benchmark tasks into five groups entailing similar challenges. By calculating baseline results, we identified the difficulty level of the tasks of each group.

Our results indicate that focusing only on the characteristics of the records to be matched is not sufficient for understanding the challenges associated with the entity resolution tasks and properties of the correspondence set need to be additionally considered. This finding complements the categorization of entity resolution tasks proposed by Mudgal et al. [2018] into structured, textual, and dirty tasks. We could identify that textual tasks, for which subsymbolic methods are known to perform well [Mudgal et al., 2018], can be equally well solved using symbolic entity resolution methods if a small number of corner cases exists in the correspondence set. With regard to structured tasks, we find that more fine-grained categorizations are needed, which involve additional properties such as schema complexity and density.

Part II

The Semantic Web as Distant Supervision for Entity Resolution

Chapter 4

Semantic Annotations on the Web

The Semantic Web emerged from the vision for a Web in which *"information is given a well-defined meaning, better enabling computers and people to work in cooperation"* [Berners-Lee et al., 2001]. This vision motivated a large body of research work to focus on developing and extending various technologies, which are commonly known as part of the *Semantic Web Stack* [Berners-Lee, 2009]. For example, data formats, such as the Resource Description Framework (RDF) [Klyne and Carroll, 2004], ontologies, and knowledge representation languages, such as the Web Ontology Language (OWL), were established to provide a universal syntactic and structural representation of information published on the Web. In addition, tools, like the SPARQL query language [Prud'hommeaux and Seaborne, 2008], were developed to facilitate the consumption of such information. These technologies and tools have enabled the realization of the vision of the Semantic Web in different ways. The *Linked Open Data Cloud* (LOD) is one form of this realization and refers to a collection of publicly available datasets containing interlinked entities of different topic areas. As of May 2020, the LOD Cloud contains 1,301 datasets with 16,283 links.[1]

Semantic annotations on the Web are another form of realization of the Semantic Web vision enabled by domain-independent markup formats as well as annotation vocabularies. These standards allow data publishers to add semantic information within the HTML pages, thus enhancing plain HTML documents with structured data and fulfilling a vital Web design aspect by separating content from presentation [Guha et al., 2015].

The usage of markup formats and vocabularies for semantically annotating data on the Web makes the content of webpages more easily consumable, crawlable and searchable [Mika, 2015]. This facilitates the design and execution of multiple web applications such as web search, price comparison, and reservation engines [Guha et al., 2015]. The consumption of semantically annotated data by more and more web applications has made the exchange of structured data important for web data publishers [Guha et al., 2015]. To ease the adoption of a consistent semantic ter-

[1] https://lod-cloud.net/#about

minology, the major search engines Bing, Google, Yahoo, and Yandex joint their efforts in 2011 and developed the schema.org vocabulary. According to the Web Data Commons project, which monitors the adoption of semantic annotations on the Web yearly, there has been significant growth in the adoption of semantic annotations in the last decade. In 2010, 5.7% (147 million) of the examined webpages contained semantically annotated data, which increased to 49.9% (1.7 billion) in 2020.[2]

Being able to extract easily and understand web content not only serves the primary goal of semantic annotations, that of making web data more easily consumable for web applications, but also makes the Web a rich source of structured data that can be used as supervision for multiple downstream tasks. For example, semantically annotated reviews with properties defining the review text and the review sentiment can be used as training data for a sentiment classification model. The trained model can then be applied to unseen and non-annotated textual reviews in order to predict their sentiment. Similarly, semantically annotated product descriptions with product identifiers such as GTIN numbers can be used as training data for entity resolution models, given that multiple product descriptions share the same product identifier. The trained model can then predict matching or non-matching relations between product descriptions that do not contain identifiers.

In this part of the thesis, we explore the potential of using semantic annotations as a source of distant supervision for entity resolution tasks. Towards this goal, in this chapter, we first give an overview of how semantic annotations can be realized and consumed. Second, we analyze the adoption trends of semantic annotations and explore the potential of using semantic annotations for generating training data for entity resolution tasks, thus covering the second contribution of the thesis [C2]. For our analysis, we set a specific focus on two widely used classes with different annotation policy recommendations by the main search engines: Product and Local Business. We monitor the adoption of semantically annotated identifying properties for these two classes, which can be used as distant supervision for different entity resolution tasks, such as matching product or local business records.

The contributions of this chapter are summarized as follows:

- We provide an overview of the adoption of semantic annotations for the period 2012 to 2020.
- We explore the utility of semantically annotated identifiers of two different schema.org classes as a form of distant supervision for building large training sets for entity resolution with no labeling effort.

This chapter is structured into seven sections. In Sections 4.1 and 4.2, we give an overview of the four main markup formats and the schema.org vocabulary used for defining structural elements and a common terminology for semantic annotations. In Section 4.3, we showcase three web applications that consume

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

semantically annotated data. Section 4.4 presents the adoption trends of the four main markup formats and schema.org vocabulary from 2012 to 2020. Section 4.5 presents the use-case analysis of exploiting semantically annotated data describing products and local businesses as source of distant supervision for entity resolution tasks. Section 4.6 discusses related work on using content of the Semantic Web as a source of supervision for different tasks other than entity resolution. Finally, Section 4.7 concludes the chapter and summarizes our main findings.

The profiling of the growth of semantic annotations is joint work with Robert Meusel and has been reported yearly since 2012 on the *Web Data Commons - Microdata, RDFa, JSON-LD, and Microformat Data Sets* page.[3] The code for profiling the growth of semantic annotations can be found on the download instructions page of each release.[4] Parts of the profiling analysis have been published in the Proceedings of the International Conference on Electronic Commerce and Web Technologies [Meusel et al., 2015] and in E-Science-Tage 2017: Forschungsdaten managen [Primpeli et al., 2017]. Exploring the Semantic Web as a source of distant supervision for different tasks, including entity resolution, has been published in the Datenbank-Spektrum Journal [Bizer et al., 2019].

4.1 Markup Formats

Semantic markup formats provide standards or recommendations for extending data objects described in HTML pages with HTML elements that can be easily parsed, consumed, and generated by machines. Four markup formats are most commonly used for adding semantic annotations to HTML pages: Microdata, RDFa, Microformats, and JSON-LD. In the following, we present each of the four markup formats and the attributes used to denote an object's metadata semantically. Next, we discuss the differences among the four markup formats along with an example code snippet.

4.1.1 Microdata

Microdata defines a set of machine-readable features which is used in combination with and as an extension of standard attributes defined within HTML. The Microdata format was first envisioned as part of the 5th major revision of the HTML language, the core language of the World Wide Web. Its purpose was to give semantic meaning to objects described in HTML pages and enhance the interoperability among user agents.[5] Microdata attributes can be used to denote the following:

1. An item, i.e. a data object within the HTML code that we want to annotate semantically. To create an item the `itemscope` attribute is used.

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

[4] http://webdatacommons.org/structureddata/2020-12/stats/how_to_get_the_data.html

[5] https://www.w3.org/standards/history/microdata

2. A property of an item. To create a property of an item, the `itemprop` attribute is used in one of the children elements of the item element.
3. The type of an item defined as a URL. To add a type to an item, the attribute `itemtype` is used.
4. The global identifier of the annotated item. To add a global identifier value, the attribute `itemid` is used.

4.1.2 RDFa

RDFa is short for *Resource Description Framework in Attributes* and, similarly to Microdata, defines a set of extensions for HTML-based languages for semantically annotating data objects. The RDFa markup format was first proposed in 2004[6] as a means to combine the standard document markup language HTML with the RDF metadata standard and facilitate RDF-related parsers to consume data published on the Web. RDFa reached W3C recommendation status in 2008[7] and comes in three main variants: RDFa 1.0, used solely in combination with XHTML documents, RDFa 1.1, the first generic RDFa standard for both HTML and XML documents, and finally RDFa Lite, a light version of RDFa containing a subset of its attributes. Below we list and explain the attributes of the RDFa markup format, which are also part of RDFa Lite:

1. `resource` defines the identifier of a data object described on a page.
2. `vocab` specifies the vocabulary used to markup a resource.
3. `prefix` allows to define and create abbreviations of more vocabularies that can be used in addition to the one defined in the `vocab` attribute.
4. `property` specifies the relationship between the subject resource and a literal value or the object resource.
5. `typeof` specifies the RDF type of the subject or the object resource.

Additionally RDFa defines the following attributes: `about`, `rel`, `rev`, `content`, and `datatype`.

4.1.3 Microformats

Microformats define a set of conventions for semantically annotating HTML documents about certain pre-defined types of entities, such as people, companies, events, and recipes. The markup format was first launched in 2005, envisioning a decentralized development of resources through machine-readable data.[8] Although Microformats is still consumed by big search engines, such as Google, for

[6] https://www.w3.org/MarkUp/2004/02/xhtml-rdf.html
[7] https://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/
[8] https://microformats.org/2005/06/20/welcome

providing rich search results, it is not a W3C recommendation. There exist several microformats with an intended use given their topical domain. In the following, we enumerate some of the most commonly used microformats together with their defined root class name, which indicates the topical domain of the annotated data object:

1. `hCard` for publishing people, companies and organizations. The root class name is *vcard*.
2. `hCalendar` for publishing events. The root class name is *vcalendar* or *vevent*.
3. `hProduct` for publishing products. The root class name is *hproduct*.
4. `hReview` for publishing reviews. The root class name is *hreview*.
5. `hRecipe` for publishing recipes. The root class name is *h-recipe*.

4.1.4 JSON-LD

JSON-LD is short for *Javascript Object Notation for Linked Data*. Its primary goal is to provide a Javascript notation for combining Linked Data in Web-based environments. It was first introduced in 2010 and became a W3C recommendation in January 2014. It is an open format, maintained and developed by a publicly open community, the JSON-LD Community Group.[9] JSON-LD provides a semantic context around data objects published in HTML pages in the form of a JSON snippet. The JSON snippet is added either in the `<head>` or in the `<body>` section of the HTML document. JSON-LD specifies several syntax tokens and keywords necessary for building the JSON metadata objects. Below we list some of the most useful JSON-LD keywords and outline their functionality:

1. `@context` defines a mapping dictionary between vocabulary terms or JSON-LD keywords and their values.
2. `@type` specifies the data type of a node or a typed value.
3. `@id` specifies an IRI or a blank node identifier for uniquely identifying data objects within the document.
4. `@vocab` specifies the vocabulary from which the properties and the `@type` keyword value originate.
5. `@value` indicates a typed value, i.e. a combination of a value and its standardized type indicated with an IRI.

[9] https://www.w3.org/community/json-ld/

SIGNATURE INSTANT DOME 7 WITH INTEGRATED FLY

Item# : 2000015676

Rated 5 out of 5 (a year ago)

Review: *Great waterproof tent!*

```
<div>
   <h1>Signature Instant Dome 7 with integrated fly</h1>
   Item# <span>2000015676</span>
   <img src="../p_2000015676.jpg">
   <div>
       <span>
       Rated 5 out of 5
       </span> (a year ago)
       Review: <span>Great waterproof tent!</span>
   </div>
</div>
```

(a) No semantic annotations

```
<div itemscope itemtype="http://schema.org/Product">
   <h1 itemprop="name">Signature Instant Dome 7 with integrated fly</h1>
   Item# <span itemprop="productID">2000015676</span>
   <img itemprop="image" src="../p_2000015676.jpg">
   <div itemprop="review" itemscope itemtype="http://schema.org/Review">
       <span itemprop="reviewRating" itemscope itemtype="http://schema.org/Rating">
       Rated <span itemprop="ratingValue"/>5 out of
       <span itemprop="bestRating"/>5</span> (a year ago)
       Review: <span itemprop="reviewBody">Great waterproof tent!</span>
   </div>
</div>
```

(b) Microdata

```
<div vocab="http://schema.org/" typeof="Product">
   <h1 property="name">Signature Instant Dome 7 with integrated fly</h1>
   Item# <span property="productID">2000015676</span>
   <img property="image" src="../p_2000015676.jpg">
   <div property="review" typeof="Review">
       <span property="reviewRating" typeof="Rating">
       Rated <span property="ratingValue"/>5 out of
       <span property="bestRating"/>5</span> (a year ago)
       Review: <span property="reviewBody">Great waterproof tent!</span>
   </div>
</div>
```

(c) RDFa

```
<div class="hproduct">
   <h1 class="fn">Signature Instant Dome 7 with integrated fly</h1>
   Item# <span class="identifier">2000015676</span>
   <img class="photo" src="../p_2000015676.jpg">
   <div class="review">
       <span class="hreview">
       Rated <span class="rating"/>5 out of
       <span class="best"/>5</span> (a year ago)
       Review: <span class="description">Great waterproof tent!</span>
   </div>
</div>
```

(d) Microformats

```
<script type="application/ld+json">
{
    "@context" : "http://schema.org",
    "@type" : "Product",
    "name" : "Signature Instant Dome 7 with integrated fly" ,
    "productID" : "2000015676" ,
    "image" : "../p_2000015676.jpg" ,
    "review": {
        "@type" : "Review",
        "reviewBody": "Great waterproof tent!",
        "reviewRating": {
            "@type": "Rating",
            "ratingValue": "5",
            "bestRating": "5"
    }
}
```

(e) JSON-LD

Figure 4.1: HTML snippets with and without semantic annotations.

4.1.5 Markup Formats Comparison

Figure 4.1a presents an example HTML snippet of a product offer describing an outdoor tent. The individual HTML elements contain specific properties of the product offer easily understood by a human reader, such as its name, product identifier, and review rating. However, these properties cannot be easily parsed by machines, e.g. search engines, to optimize their search results. Adding semantic annotations using one of the presented markup formats can help a search engine to understand the content of the page.

Figures 4.1 (b-e) present the same HTML example snippet annotated with the four markup formats. Microdata (Figure 4.1b) and RDFa (Figure 4.1c) exploit different terms for differentiating between classes and properties of data objects. In contrast, the Microformats (Figure 4.1d) markup format only uses the keyword *class* for both. Additionally, Microformats is rather strict in its usage as it specifies attribute terms and cannot be combined with external vocabularies. On the other hand, RDFa, Microdata, and JSON-LD rely on external vocabularies. For example, in order to signify the name of the product in the presented example, the Microformats format uses the *fn* property defined within the *hProduct* class. In contrast, the other three markup formats exploit the Product/name property defined by the schema.org vocabulary. JSON-LD can be considered easier to implement than the other three formats, as webmasters need to add the JSON snippet either in the head or the body of the HTML document, without wrapping the semantic notations around the HTML elements. Due to its simple maintenance and parsing, JSON-LD is the recommended format by Google Search for enhancing the search results.[10]

4.2 Schema.org Vocabulary

Vocabularies Prior to Schema.org Initial attempts to develop a common vocabulary for semantically annotating webpages targeted specific applications [Guha et al., 2015]. For example, the RSS vocabulary (Rich Site Summary) released in 1999 allowed users to add news feeds from different sources to their webpages. Microformats like hCalendar and hCard were designed to be used with specific terms serving calendar and contact information exchange, respectively, while the FOAF vocabulary (Friend of a Friend) was used to annotate social network data. A historical overview of different vocabularies and their target applications is provided by Guha et al. [2015]. All those vocabularies contained a limited number of terms covering the needs of the specific applications they were addressing.

Web search was the first application that sought to consume semantic annotations of a broader coverage with the aim to enhance search results [Guha et al., 2015]. Early attempts of web search applications to establish guidelines for semantically annotating web content, such as Yahoo's SearchMonkey in 2008 and

[10] https://developers.google.com/search/docs/advanced/structured-data/intro-structured-data

Google Rich Snippets in 2009, led to data silos [Mika, 2015]. Each application expected different semantic annotation mechanisms and terms, i.e. different attribute names were used to denote the same type of data objects. This was burdensome for web content publishers. Therefore the need to simplify the effort of web content publishers, by providing one common terminology understood and consumed by all main search engines, emerged.

Initial Development and Extensions This need was addressed in 2011 with the development of schema.org, a joint initiative by Google, Microsoft, Yandex, and Yahoo. The schema.org vocabulary is a collection of terms that can be used to describe data objects of different types, such as products, persons, job postings, and recipes. Schema.org terms can be combined with different markup formats, such as RDFa, Microdata, and JSON-LD.

In its initial version, schema.org contained 297 classes/types and 187 relations/properties [Guha et al., 2015]. Until October 2021, the schema.org vocabulary has been extended through 52 releases.[11] At its launch, schema.org was designed to be extensible in two ways, which were more formally defined in 2015 as *hosted* and *external* extensions [Guha et al., 2015]. Hosted extensions are discussed with the broader community in collaboration with experts and integrated within the core of the schema.org vocabulary. External extensions are coordinated independently and are not integrated into schema.org's core. An example of an external extension of schema.org is the GS1 web vocabulary, which supports fine-grained product data descriptions according to product regulation standards, e.g. EU No1169.[12]

The discussions regarding the vocabulary extensions initially relied on project-based collaborations, but soon evolved to occur within a public and open community, taking place over the W3C public-vocabs mailing list [Guha et al., 2015]. In March 2015, the community discussions concerning maintenance and extensions were moved to a W3C Community Group [Mika, 2015].[13] The latest release (October 2021) 13.0 contains 792 classes, 1447 relations/properties, 15 datatypes, 83 enumerations and 445 enumeration members.[14]

Hierarchy of Terms The classes of the schema.org vocabulary are arranged in a hierarchy, with *Thing* being the most generic class. In the 13.0 release, there exist the following ten classes under Thing: Action, BioChemEntity, CreativeWork, Event, Intangible, MedicalEntity, Organization, Person, Place, and Product. The properties of the vocabulary can be used across different classes but have an indented usage with regard to the classes and values with which they are compatible. Figure 4.2 shows part of the documentation of the Product schema.org class. Each property of the class product is accompanied by its expected type and descrip-

[11] https://schema.org/docs/releases.html
[12] https://www.gs1.org/voc/
[13] https://www.w3.org/community/schemaorg/
[14] https://schema.org/docs/schemas.html

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 characteristics of the entitity, 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.
`audience`	Audience	An intended audience, i.e. a group for whom something was created. Supersedes serviceAudience.
`award`	Text	An award won by or for this item. Supersedes awards.
`brand`	Brand or Organization	The brand(s) associated with a product or service, or the brand(s) maintained by an organization or business person.
`category`	PhysicalActivityCategory or Text or Thing or URL	A category for the item. Greater signs or slashes can be used to informally indicate a category hierarchy.
`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.
`gtin14`	Text	The GTIN-14 code of the product, or the product to which the offer refers. See GS1 GTIN Summary for more details.
`manufacturer`	Organization	The manufacturer of the product.
`material`	Product or Text or URL	A material that something is made from, e.g. leather, wool, cotton, paper.

Figure 4.2: Part of the documentation of the schema.org Product class.

tion. For example, the property *manufacturer* can denote the manufacturer of the product and expects a value of type *Organization*. Additionally, schema.org offers properties for indicating product identifiers, such as *gtin13* and *gtin14*, which both expect textual values.

4.3 Applications using Semantic Annotations

In the following, we present example applications that consume semantic annotations embedded in HTML pages: web search, intelligent personal assistants, product catalogs, and comparison portals. An extensive list of applications consuming semantic annotations on the Web can be found in the work of Guha et al. [2015].

4.3.1 Web Search

One of the main contributions of semantic markup annotations is that they can be used to enhance the search results in the form of additional information snippets, i.e. compact summaries of the content of the page. As each search engine may implement a different crawling strategy for retrieving search results, guidelines for best annotation practices are published in the webmasters' documentation of each web search engine. In the following, we give an overview of the markup annotation

guidelines of Yahoo and Bing[15] as well as Google[16] and Yandex[17] as reported in their webmasters' documentation as of October 2021.

Annotation Guidelines of Bing and Yahoo The common crawling strategy of the Bing and Yahoo[18] search engines supports the following vocabularies and markup formats: schema.org with Microdata or JSON-LD, Microformats, RDFa, and Open Graph. Nine types of entities are supported in total, such as People, Recipes, and Products. However, except for the Microdata format, which can be used with any of the nine types, some types are not supported in combination with specific formats, e.g. Breadcrumbs with Microformats and Products with RDFa. Additionally, an extension of the defined nine types is allowed through the use of the schema.org classes, e.g. a web content publisher can annotate an element with the schema.org class Place and its properties, even though Place is not one of the nine defined classes which Bing and Yahoo support.

Annotation Guidelines of Google Google Search supports structured data in Microformats, RDFa, Microdata, and JSON-LD format, while the latter is recommended. Schema.org is the only supported vocabulary after the announcement in April 2020 that data-vocabulary.org, which was the predecessor of schema.org, will no longer be eligible for Google rich results features.[19] Depending on the annotation type, Google distinguishes between required and optional properties. In order to check the validity of the embedded structured data against the crawling strategy of Google, webmasters can use publicly available structured data validation tools such as the Rich Results Test tool[20] or the schema.org Validator.[21] Validating the code of Figure 4.1b against the Rich Results Test Tool outputs one error due to the missing author property of the *Review* item, as well as some warnings due to missing optional product properties. These properties are recommended for optimized search results by Google, such as product description and brand.

Annotation Guidelines of Yandex Yandex supports the enrichment of search snippets with the schema.org vocabulary in combination with the RDFa, Microdata, or JSON-LD markup formats. Additionally, Microformats are supported. Although all schema.org classes can be used for annotating data objects in HTML pages, only two Microformats are supported: *hCard* for marking up contact information such as addresses or phone numbers and *hRecipe* for marking up recipes. Similar to the Google validation tool, Yandex specifies required properties per type

[15] https://www.bing.com/webmasters/help/marking-up-your-site-with-structured-data-3a93e731
[16] https://developers.google.com/search/docs/advanced/structured-data/sd-policies
[17] https://yandex.com/support/webmaster/
[18] https://en-global.help.yahoo.com/kb/SLN2213.html
[19] https://developers.google.com/search/blog/2020/01/data-vocabulary
[20] https://search.google.com/test/rich-results
[21] https://validator.schema.org/

and offers its own structured data validator[22] whose usage is restricted to registered Yandex users. For example, the properties name, description, price, and priceCurrency are required to form a snippet describing a product.

4.3.2 Intelligent Personal Assistants

Structured data assist automated agents such as chatbots, voice assistants, and e-mail assistant applications with the completion of web service tasks, such as question answering and scheduling of appointments. The Google Voice Assistant exploits the schema.org property *speakable* within an HTML page for identifying text snippets that are suitable for audio playback using text-to-speech.[23] Similarly, Cortana, the virtual assistant developed by Microsoft uses markup annotations to synchronize different applications, such as the e-mail and calendar applications [Guha et al., 2015]. For example, an e-mail containing structured data concerning a flight reservation would assist Cortana in creating the corresponding event in the calendar application of the user. Neumaier et al. [2017] developed a prototype for a chatbot that accumulates Open Data, translates them to a unified schema.org template, and exploits schema.org annotations for question answering. Such usage of structured data for assisting chatbots has attracted the attention of providers that support conversational AI providers, such as Schema App[24] and Onlim[25].

4.3.3 Product Catalogs and Comparison Portals

Annotated data can be exploited for building catalogs and comparison portals, with the product domain being well-suited for such downstream applications. Google products rebranded as Google Shopping in 2012, exploits structured data to create comprehensive lists of products and compare the prices of product offers from different merchants. To support merchants in publishing their offers, Google established the Merchant Center, which provides guidelines on adding semantic annotations on product offers.

4.4 Adoption of Semantic Annotations

In this section, we analyze the yearly adoption of web semantic annotations in the period 2012-2020 and explore which markup formats are the most widely used, both in general and in combination with the schema.org vocabulary. This exploration can provide a first estimate of the potential of exploiting semantic annotations, complying with specific markup formats and using schema.org terms, as a source of distant supervision. For our analysis, we compile the profiling results of the overall adoption of semantic annotations as well as of the adoption of the four

[22] https://webmaster.yandex.com/microtest.xml
[23] https://developers.google.com/search/docs/advanced/structured-data/speakable
[24] https://www.schemaapp.com/schema-markup/schema-markup-make-chatbots-intelligent/
[25] https://onlim.com/en/conversational-ai-and-schema-org/

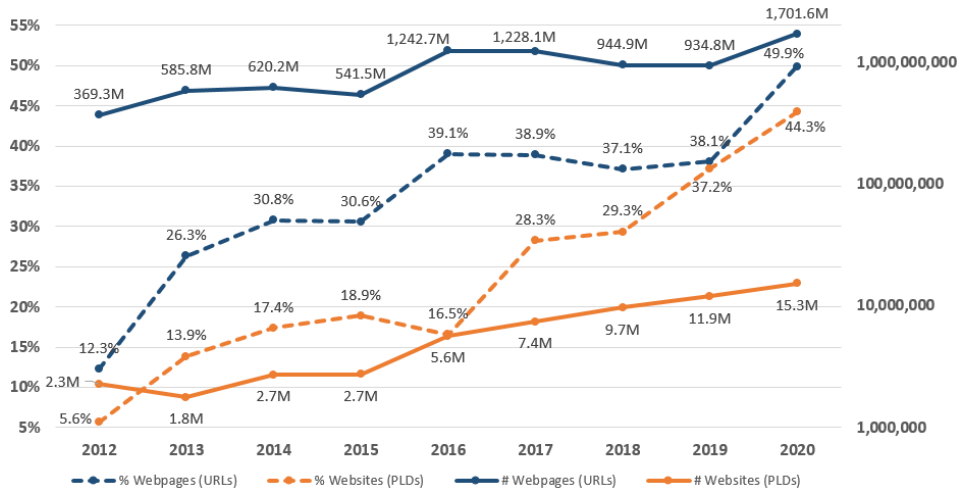


Figure 4.3: Relative and absolute number of webpages (blue) and websites (orange) per WDC release with semantic annotations.

markup formats, introduced in Section 4.1 and schema.org vocabulary, introduced in Section 4.2, as published by the Web Data Commons (WDC) project,[26] which was maintained since 2016 as part of the work of this thesis. The WDC project extracts yearly all Microdata, JSON-LD, RDFa, and Microformats data from the Common Crawl web corpus,[27] the largest and most up-to-date web corpus that is currently available to the public. The extracted data are available for download in the form of RDF-quads. In addition, yearly statistics on the deployment of the different markup formats and vocabularies are calculated and published. Considering that JSON-LD became a W3C recommendation in 2014, the first reported statistics on the adoption of this format appeared in the WDC 2015 release. It is important to mention that the corresponding Common Crawl corpora, used to extract the yearly WDC structured data releases, differ in size, number of webpages (URLs), number of pay-level domains (PLDs), i.e. websites, as well as crawl depth, i.e. the amount of crawled pages within the same website. Although this is important for interpreting the overall trends, a deeper analysis of the effect of the yearly crawling strategy of Common Crawl on the resulting WDC corpora is beyond the scope of this work.

4.4.1 Overall Adoption Trends

The line chart of Figure 4.3 presents the relative (primary y ax - left) and absolute (secondary y ax, log scale - right) number of webpages (URLs) and websites (PLDs) with semantic annotations per year. The 2012 WDC release revealed that only 5.6% of the PLDs in the crawled corpus contained semantic annotations. To put things into context, we remind the reader that the joint effort of the four main

[26] http://webdatacommons.org/
[27] http://commoncrawl.org/

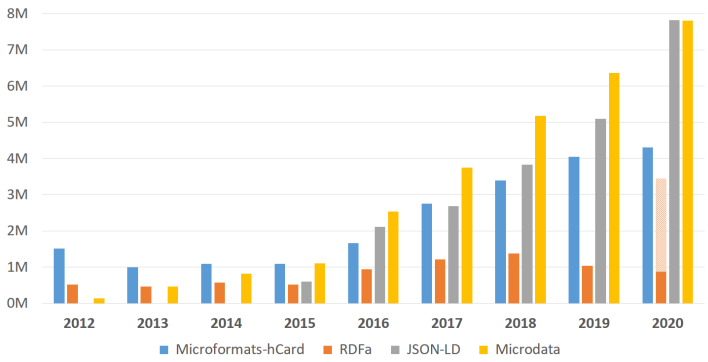


Figure 4.4: Websites in millions deploying the major markup formats per year.

search engines Google, Bing, Yahoo, and Yandex to create the schema.org vocabulary and motivate the web content publishers to add semantic annotations for improving search results was introduced in 2011. Although we will look in more detail at the adoption growth of the schema.org vocabulary in the following paragraphs, we can already attribute the large relative jump in the years after 2012 to the improvement of the collaboration between publishers and consumers of web content through the common ground offered by the schema.org vocabulary.

The relative growth of URLs with semantic annotations is mostly positively correlated with the yearly growth of PLDs with semantic annotations. A noticeable exception occurred in 2016, for which we see a high jump in terms of PLD growth but a decrease in the relative URL growth. The reason for this is the considerable difference in the depth of the October 2016 Common Crawl corpus compared to the rest of the crawls. More concretely, the average number of URLs per PLD was larger than 500 in the 2016 Common Crawl corpus, which is 5-7 times larger than the average number of URLs per PLD for the rest of the crawled corpora used for analysis. Within eight years, the structured data adoption ratio increased to 44.3% and 49.9% in terms of PLDs and URLs, respectively, indicating that almost half of the websites and webpages in the crawl use markup annotations to semantically annotate their web content. This corresponds to 15.3 million websites and 1.7 billion pages, which is more than four times larger than the adoption reported in 2012.

4.4.2 Markup Format Adoption Trends

Despite the overall continuous increase in the deployment of structured data, not all markup formats follow the same growth rate. We demonstrate this with the barplot of Figure 4.4, which presents the adoption of the four major markup formats in 2012-2020 in terms of the absolute number of websites (PLDs). We can observe that after 2016, the Microdata and JSON-LD formats dominate over the hCard and RDFa formats. Through the years, JSON-LD has become as popular as Microdata

and has grown over 50% from 2019 to 2020. Microformat hCard is also adopted by a steadily increasing number of websites. However, considering that Microformats are used together with a specific set of inextensible terms of a flat hierarchy, their expressivity is rather limited in comparison to extensible vocabularies, such as schema.org. We verify this observation by looking at the top classes by the number of PLDs for the different markup formats for the 2020 WDC release: for the Microformats hCard, the top classes linked to the root class vcard2006 are *Name, VCard*, and *Organization*, which apart from the latter do not signify the specific type of the annotated data object, such as WebSite, Person, Product, SearchAction, Breadcrumb, Article and Document, which are some of the top classes for the other three markup formats. It is worth noting that the large increase in the number of PLDs using RDFa annotations in 2020 in comparison to the previous years is due to a different extraction strategy. Between 2012 and 2019, only RDFa annotations found in the `body` element of the HTML page were extracted, while in 2020 we additionally extracted RDFa annotations found in the `<header>` element of the HTML page. For a fair comparison to the previous years, we represent the number of PLDs annotating RDFa data as a stacked bar in Figure 4.4: the upper light-colored part indicates the number of PLDs that include RDFa annotations only in the `<header>` elements while the lower part indicates the number of PLDs that include RDFa annotations in the `<body>` element of their HTML pages.

4.4.3 Schema.org Adoption Trends

As introduced in Section 4.2, the schema.org vocabulary offers a set of terms arranged in a hierarchy for annotating different types of entities. Being promoted by the main search engines and used in different downstream applications such as web search, intelligent personal assistants, and online advertising, schema.org has seen noteworthy growth over the last few years. We demonstrate the growth of the schema.org vocabulary with the line chart of Figure 4.5, which shows the number of websites using the schema.org vocabulary in combination with either the Microdata or the JSON-LD format since 2012 and 2015, respectively. Additionally, we depict with dotted lines the relative number of websites using schema.org with Microdata or JSON-LD with respect to the overall number of websites with Microdata and JSON-LD, respectively.

Schema.org migrated its website from http to https and changed the provided annotation examples to "https://schema.org"-based URLs during 2020. Although this action is not clearly documented, it can be verified by comparing the different archived versions of the schema.org website. For example, the documentation of the schema.org/Product class in August 2019 provides an "**http**://schema.org"-based example,[28] while the version of November 2021[29] provides an "**https**://schema.org"-based example. Additionally, the usage of both http and https in schema.org annotations is confirmed as valid in the Frequently Asked Questions section of the

[28] https://web.archive.org/web/20190809175648/http://schema.org/Product

[29] https://web.archive.org/web/20201102143606/https://schema.org/Product

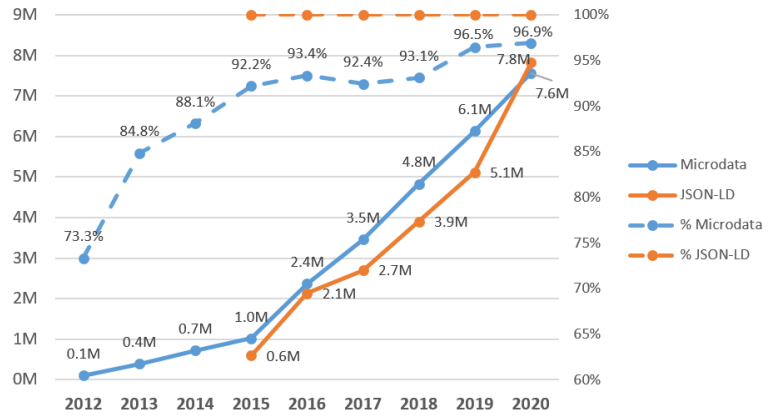


Figure 4.5: Absolute (in millions) and relative number of websites using schema.org with either Microdata (blue) or JSON-LD (orange).

schema.org website.[30] Therefore, in Figure 4.5, we present the number of websites that use either suffixes for annotating their structured data.

Given the yearly PLD adoption of the JSON-LD and Microdata markup formats, we can observe that the two formats are highly interlinked with the schema.org vocabulary. More specifically, all webmasters that use the JSON-LD format to mark up their data include at least one schema.org annotation on their websites. This trend has been observed yearly since the first reported adoption statistics of JSON-LD in 2015. The same is not applied to the Microdata markup format, for which we see a continuously growing trend of its usage in combination with the schema.org vocabulary. In 2012, 73.3% of the websites using Microdata were found to include at least one schema.org annotation in their websites. The respective ratio for 2016 is 93.4%, while for 2020 it is 96.9%. With respect to the ratio of schema.org entities to all Microdata and JSON-LD entities per year, we can observe a similar trend to the one of the PLDs. All entities (99.9%) marked up using the JSON-LD format found in the WDC releases from 2015 to 2020 are annotated with a schema.org class. The corresponding ratio for the entities marked up with the Microdata format has grown from 59.6% to 94.4%. The continuously increasing adoption of the schema.org vocabulary in combination with the two most widely used markup formats, i.e. Microdata and JSON-LD, indicates that semantically annotated web content with this vocabulary and markup formats may have a promising potential as distant supervision for different tasks. In order to verify this assumption for the task of entity resolution, we will look in the next section at specific schema.org classes and the adoption of their identifying properties.

[30] https://schema.org/docs/faq.html#19

4.5 Use-Case Analysis: Semantic Annotations as Distant Supervision for Domain-Specific Entity Resolution Tasks

This section explores the potential of using semantically annotated data of the Product and LocalBusiness, with its subclasses Hotel and Restaurant, schema.org classes as a source of training data for entity resolution tasks of the same topical domain. Figure 4.6 presents the growth of the selected classes in terms of the number of websites annotating web content with the respective classes using either the Microdata or the JSON-LD markup formats for the period 2013-2020. In general, we observe a growing adoption trend throughout the years for all classes. While the number of parsed websites of the Common Crawl corpora used for the 2013 and 2020 WDC extraction has almost tripled from 12.8 million to 34.5 million, respectively, the number of websites using the selected schema.org classes in 2020 is at least $9\times$ larger in comparison to 2013.

Nevertheless, some drops between the years can be noticed. For example, the number of websites using schema.org/Restaurant annotations decreased from 5,079 to 3,901 between 2014 and 2015. Such drops are primarily attributed to the different crawling strategies employed by Common Crawl throughout the years and secondarily to the changes in the adoption practices followed by the webmasters. Following the example of the drop in schema.org/Restaurant annotations from 2014 to 2015 and comparing the Common Crawl releases used for the WDC 2014 and 2015 versions, we see that the 2014 version contains 1.2 million more websites in comparison to the 2015 version. By analyzing the websites employing schema.org/Restaurant annotations, we find that 70% of the websites appearing in the WDC 2014 version but not in the WDC 2015 version are not found in the respective 2015 Common Crawl release. Therefore these websites were excluded by the crawler. Examples of such websites are *tapasitas.com*, *californiapizza.ca*, and *online-sushi.com*. For the rest 30% of the websites, which could be found in the respective 2015 Common Crawl release, we observe a shift in the annotation practices of the webmasters: schema.org/Restaurant annotations were included in 2014 but then removed in 2015.[31] Examples of such websites are *giantrusticpizza.com*, *jimmysbarandgrill.com*, and *wingatehotelnyc.com*.

In the following, we profile the adoption of specific schema.org properties, which when used alone or in combination, can uniquely identify a real-world object of a specific class, e.g. a GTIN number for products or a tax identification number for businesses. The profiling analysis of the identifying schema.org properties is done on website (PLD) and markedup entity level. To avoid confusion with the entity definition presented in Section 2.2, we clarify that a markedup entity refers to the set of semantic annotations describing an object or a concept within a single HTML page. Figure 4.1e illustrates an example of a JSON-LD markedup entity describing an outdoor tent product offer.

The existence and consistent usage of such identifying schema.org properties

[31] Verified using the Wayback Machine: https://web.archive.org/

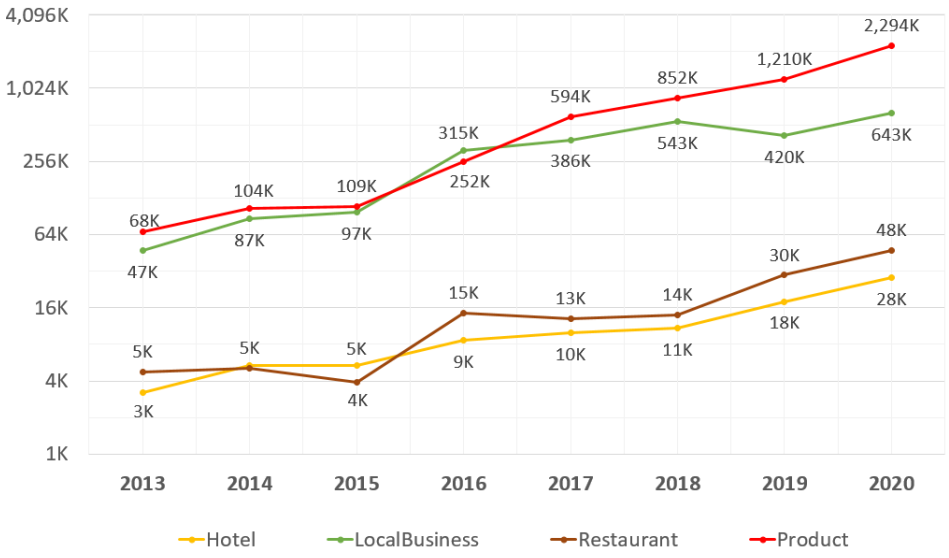


Figure 4.6: Websites in thousands with selected schema.org classes.

can be a valuable source of distant supervision. By comparing the values of the identifying schema.org properties, we can group the records, i.e. markedup entities, into clusters representing unique real-world objects from which matching (intra-cluster) and non-matching (inter-cluster) record pairs can be retrieved without the need of manual annotation. The generated correspondence sets can be exploited for training entity resolution models. The distantly trained models can then be applied to new record pairs describing real-world objects of the same class and predict matching or non-matching relations without requiring explicit identifier information.

4.5.1 Use-case 1: schema.org Product Annotations

The Product class is one of the most widely used schema.org classes in terms of the number of annotated entities and websites, according to the yearly WDC schema.org subset statistics.[32] As shown in Figure 4.6, the adoption of schema.org/Product annotations is increasing significantly per year. In 2013, 68 thousand websites were found to annotate product-related data, while in 2017, the number of websites was $8.7\times$ larger. In the 2020 version of the WDC structured data analysis, 2.2 million websites were found to use the schema.org/Product class in combination with either the Microdata or the JSON-LD markup format.

Identifying Schema.org Properties According to the documentation of the schema.org/Product class, the Product type can be used for annotating an offered

[32] http://webdatacommons.org/structureddata/2020-12/stats/schema_org_subsets.html

Table 4.1: Absolute (in thousands) and relative number of schema.org/Product entities and websites with identifier properties.

schema.org/ Product property	2013				2017				2020			
	Markedup entities		PLDs		Markedup entities		PLDs		Markedup entities		PLDs	
	# (in K)	%	# (in K)	%	# (in K)	%	# (in K)	%	# (in K)	%	# (in K)	%
gtin8	0.3	0.0	0.0	0.0	540.9	0.1	0.3	0.0	3,661.9	0.5	22.8	1.0
gtin12	0.3	0.0	0.0	0.0	507.6	0.1	0.6	0.1	4,052.6	0.5	24.5	1.0
gtin13	177.5	0.1	0.3	0.4	5,737.9	1.2	6.6	1.0	29,590.2	3.7	68.0	2.9
gtin14	10.9	0.0	0.0	0.0	578.1	0.1	0.8	0.1	2,784.0	0.3	18.0	0.8
identifier	273.4	0.2	0.2	0.2	425.3	0.1	0.6	0.1	4,197.5	0.5	14.1	0.6
productID	28,427.0	16.0	7.4	10.8	54,787.4	11.0	38.0	6.3	51,663.9	6.5	109.3	4.7
mpn	1,561.4	0.9	0.5	0.7	15,678.3	3.2	10.1	1.7	69,860.7	8.8	148.0	6.3
sku	14,513.1	8.2	1.3	1.9	49,732.8	10.0	150.4	25.3	241,700.5	30.4	1,291.1	56.2

product item or service, such as a pair of shoes or a concert ticket.[33] The schema.org vocabulary offers 57 properties. Among those, the following eight properties describe product identifiers: *gtin8*, *gtin12*, *gtin13*, *gtin14*, *mpn*, *productID*, *identifier*, and *sku*. The gtin-based properties are indented for describing global trade item numbers, i.e. identifier values for distinguishing trade items and services developed by GS1.[34] The mpn and sku vendor-specific properties are assigned by the products' manufacturers and describe the manufacturer part number and stock keeping unit, respectively. Finally, the productID and identifier properties can be used to describe any product identifier value, such as an ISBN number. The mentioned identifying properties, with the exception of the more general productID and identifier properties, are listed as recommended for rich results eligibility when marking up product-related content by Google.[35] Additionally, Google made an official announcement in February 2021 through the Google Search Central Blog, motivating and providing guidelines to the webmasters for semantically annotating unique product identifiers.[36] If used consistently, such global identifiers allow offers for the same product from multiple e-shops to be grouped into clusters and are therefore a valuable source of distant supervision.

Adoption Statistics In order to estimate the potential of the annotated product identifiers as a source of distant supervision, we measure their adoption in different WDC releases and analyze the overlap of the identifier property values. Table 4.1 presents the absolute and relative number of websites using the respective identifier properties for three WDC Microdata and JSON-LD corpora in 2013, 2017, and 2020. Additionally, we report the absolute and relative number of markedup entities of type schema.org/Product containing the respective properties. We observe that there has been a significant increase in the adoption of identifying properties of product markedup entities over the years. For example, the vendor-specific sku

[33] https://schema.org/Product

[34] https://www.gs1.org/

[35] https://developers.google.com/search/docs/advanced/structured-data/product

[36] https://developers.google.com/search/blog/2021/02/product-information

Table 4.2: Group size distribution of product identifiers overlap.

Group size	# Occurrences		
	2013	2017	2020
1	4,492,158	41,538,284	100,464,424
[2-5]	2,084,117	10,676,183	36,033,295
[6-10]	409,380	1,234,244	5,254,722
[11-20]	285,232	634,551	2,576,139
[21-50]	180,727	310,188	1,182,615
[51-100]	54,480	95,664	295,704
[101-200]	23,863	40,512	123,900
[201-500]	17,408	16,411	65,008
[501-800]	4,082	3,049	16,205
[801-1000]	1,208	917	5,057
[1001-5000]	2,436	2,127	11,623
[5001-10000]	125	208	852
>10000	161	497	465

identifying property was used by 1.9% of the PLDs in 2013, while the respective relative adoption for 2017 and 2020 is 25.3% and 56.2%. Interestingly, we see that e-shops tend to move from general identifying properties to more specific ones. In 2013, 10.8% of the PLDs annotating products used the general productID property, while its relative adoption dropped to 4.7% in 2020. Looking at the gtin-based global identifiers, we observe an overall increasing trend from less than 1% relative PLD adoption in 2013 to more than 5% relative PLD adoption in 2020. This suggests that e-shops adapt to the main search engine guidelines to provide unique global identifiers for their offered products.

Overlap Estimation The pre-requisite for generating training sets from semantically annotated product offers for entity resolution is the overlap of their identifying property values. Product offers having the same annotated product identifiers can be used to build up matching record pairs. In contrast, product offers with different semantically annotated identifiers can be exploited for building non-matching record pairs. We estimate the product identifiers overlap by calculating the number of occurrences of the identifier values regardless of the identifying schema.org property in the three Product WDC corpora. Table 4.2 presents the frequency per group size range per year, e.g. in 2017, there exist 1.23 million distinct identifier values that appear in 6 to 10 markedup entities each. Despite the large majority of identifying values appearing only once (group size 1) for all three yearly snapshots, we observe that there exists a large and continuously increasing amount of identifying values that appear in multiple markedup entities. Exploiting the offers of groups with size [2-10] in the WDC 2017 Product corpus, we can produce 55.6 million pairs of matching offers without investing any labeling effort. Although more

Table 4.3: Absolute and relative number of schema.org/LocalBusiness entities and websites with identifier properties.

schema.org/ LocalBusiness property	2013				2017				2020			
	Markedup entities		PLDs		Markedup entities		PLDs		Markedup entities		PLDs	
	#	%	#	%	#	%	#	%	#	%	#	%
duns	9	0.0	7	0.0	18,897	0.0	124	0.0	25,916	0.0	179	0.0
GLN	0.0	0.0	0.0	0.0	18,820	0.0	6	0.0	1,000	0.0	10	0.0
leiCode	0.0	0.0	0.0	0.0	133	0.0	5	0.0	2,368	0.0	32	0.0
taxID	1,226	0.0	23	0.0	262,770	0.3	396	0.1	586,371	0.8	901	0.1
vatID	1,567	0.0	27	0.0	259,451	0.3	2,864	0.7	956,118	1.3	12,890	2.0
identifier	0.0	0.0	0.0	0.0	43	0.0	1	0.0	37,326	0.1	138	0.0
telephone	9,005,683	11.8	21,491	45.7	36,049,923	44.6	169,375	43.8	48,001,143	64.5	509,804	79.2
geo	1,900,621[39]	83.3	1,754	3.7	13,929,608	17.2	21,046	5.4	9,253,675	12.4	128,631	20.0

elaborate cleansing and grouping steps are required considering the noisy nature of web data which will be further investigated in Chapter 5, we can conclude that there exists a considerable potential of generating training data for product-related entity resolution tasks by exploiting the semantically annotated product markedup entities.

4.5.2 Use-case 2: schema.org Local Business Annotations

According to the schema.org documentation, the schema.org type LocalBusiness describes a physical business or branch of an organization.[37] It is a subclass of schema.org/Organization or schema.org/Place classes and a parent class of specific types of businesses such as schema.org/Hotel and schema.org/Restaurant.

Identifying Schema.org Properties As subclasses of schema.org/Organization, the classes LocalBusiness, Hotel, and Restaurant inherit the properties of the Organization type, among which the following five properties can be used for denoting global identifiers: *duns*, *globalLocationNumber*, *leiCode*, *taxID*, and *vatID*. The *duns* property refers to the Dun & Bradstreet DUNS (Data Universal Numbering System) number, a unique nine-digit identifier for businesses.[38] The numbering system was established in 1963 and assigns a numerical identifier at the lowest organizational level, i.e. a business location with a distinct operation. The *globalLocationNumber* property (GLN) is a 13-digit number for identifying parties or physical locations and is part of the GS1 systems of standards. The *leiCode* property refers to the Legal Identifier Code, a 20-character alpha-numeric code based on the ISO 17442 standard. The property *vatID* refers to the value-added tax identification number, while the *taxID* is indented for covering country-specific tax identifiers.

[37] https://schema.org/LocalBusiness

[38] https://www.dnb.com/duns-number.html

[39] For the calculation of #Entities the domain citysearch.com has been excluded as it accounted for more than 97% of the entities. This is due to a focus of the crawl in that domain which did not occur in the next years.

Table 4.4: Group size distribution of local business, hotel, and restaurants identifiers overlap.

Group size	# Occurrences								
	Local business			Hotel			Restaurant		
	2013	2017	2020	2013	2017	2020	2013	2017	2020
1	1,773	184,843	59,723	5	76	607	3	166	30769
[2-5]	64	7,134	10,932	2	28	237	1	15	6816
[6-10]	7	5,422	3,395	0	25	117	0	6	191
[11-20]	5	1,486	2,192	0	25	81	0	4	73
[21-50]	3	743	1,863	0	20	43	0	1	63
[51-100]	-	145	1,007	0	5	23	0	3	10
[101-200]	-	169	738	0	1	5	0	0	6
[201-500]	-	197	865	0	0	13	0	0	0
[501-800]	1	56	394	0	0	1	0	0	0
[801-1000]	-	15	144	0	0	0	0	0	0
[1001-5000]	-	16	173	0	0	1	0	0	0
[5001-10000]	-	2	4	0	0	0	0	0	0
>10000	-	2	6	0	0	0	0	0	0

Adoption Statistics The Google documentation on annotating data content with the schema.org type LocalBusiness does not include any of the above-mentioned identifying properties as recommended. Table 4.3 presents the adoption of schema.org identifier properties in 2013, 2017, and 2020 in terms of absolute and relative numbers of websites and markedup entities. We observe that there is overall a slowly increasing trend in annotating global identifiers for local businesses, while the taxID and vatID properties are more frequently used in comparison to the rest of the identifying properties. Despite the increasing adoption, the ratio of PLDs using either taxID or vatID for annotating global identifiers of businesses to all PLDs annotating local businesses remains lower than 2.5% even in 2020.

Overlap Estimation Table 4.4 presents the group size distribution given the overlap of the identifier values of all identifying properties for the three years 2013, 2017, and 2020 for the parent class LocalBusiness and two of its subclasses, Hotel and Restaurant. We observe that in contrast to 2013, where we have no sufficient overlap for generating large amounts of matching LocalBusiness record pairs, in 2017 and 2020, there exist more than 15 thousand and 21 thousand identifier values respectively that occur in more than one LocalBusiness markedup entity and can therefore be exploited for generating correspondence sets with considerable amounts of matching record pairs. Even though the identifier value overlap for the two subclasses Hotel and Restaurant is 2 to 30 times smaller in terms of occurrences per group size in the WDC 2020 corpus, the estimated amount of matching record pairs that can be generated from the groups of size smaller than 50, is 35 thousand and 54 thousand respectively. This indicates that despite the limited adoption of identifying properties for these two classes, a training set of considerable

Table 4.5: Group size distribution of local business identifiers overlap. Grouping based on exact match of the telephone and on approximate match (< 300 meters distance) of the geo property values.

Group size	[2-5]	[6-10]	[11-20]	[21-50]	[51-100]	>100
# Occurrences	96,773	7,134	1,998	563	115	105

size can still be generated and used as distant supervision for entity resolution tasks containing hotel or restaurant records.

Alternative Identifying Properties Besides the properties for annotating global identifiers of businesses, the schema.org/LocalBusiness class offers the *geo* and *telephone* properties, which, when used in combination, can help disambiguate local businesses. Using both properties has the following two advantages over using only one of them: First, it enables distinguishing different local businesses which belong to the same business chain and thus often share the same telephone number. Therefore, combining geographical information with the telephone number can uniquely identify the individual establishment of a business chain. Second, it enables distinguishing different local businesses located in the same building and thus have the same address.

In contrast to the schema.org properties describing global identifiers, the geo and telephone properties are recommended by Google for annotating businesses.[40] This leads to broader adoption in comparison to the adoption of global identifier properties, as presented above. Indeed, in 2013, 45.7% of the websites annotating businesses (9 million entities) included telephone number annotations, while in 2020, the corresponding adoption increased to 79.2% (48 million entities). The property schema.org/geo was used by 3.7% of the websites in 2013 (1.9 thousand entities), while in 2020, the adoption increased to 20% (9.2 million entities).

Although combining the values of the telephone and geo schema.org properties can be exploited for grouping the LocalBusiness markedup entities, normalizing the geo values to the same level of precision and the telephone number to a consistent standard, e.g. [+][country code][area code][number], requires extensive pre-processing. We apply the following normalization pipeline[41] and group the markedup entities of the WDC 2020 release: First, non-numerical characters are removed from the telephone values. Next, the telephone values are converted to the international telephone number standard E.164.[42] Finally, the geo-coordinates are converted to float values. The markedup entities with exactly the same telephone numbers and a geo distance of no more than 300 meters are grouped together and

[40] https://developers.google.com/search/docs/advanced/structured-data/local-business#structured-data-type-definitions

[41] The normalization pipeline is part of the yet unpublished work of Alexander Brinkmann on *Unsupervised and Supervised Blocking using Contrastive Learning for Entity Resolution*.

[42] https://www.itu.int/rec/T-REC-E.164/en

therefore considered to describe the same real-world business.

Table 4.5 presents the occurrences per group size with the grouping being generated using the telephone and geo property values after applying the previously described cleansing process on the WDC 2020 LocalBusiness corpus. Considering the groups of size smaller than 50, more than 700 thousand matching record pairs can be retrieved. This further indicates that despite the more elaborate cleansing process, exploiting the geo and telephone number properties for grouping can also produce a large amount of potentially matching record pairs that can be used as distant supervision.

4.6 The Semantic Web as Distant Supervision for Other Tasks

Using content from the Semantic Web as distant supervision has been explored in different tasks, such as relation extraction, product categorization, and product feature extraction. Table 4.6 presents a summarized overview of works using semantic annotations or knowledge bases as a source of distant supervision for different tasks. The column *Source* refers to the data source that was used for extracting labeled data, the column *Usage of Semantic Annotations* indicates if semantically annotated data in HTML pages were exploited for constructing the labeled set and the *Training Size* column shows how many labeled entities were extracted and used for training machine learning models.

Information Extraction Semantic annotations about types, e.g. product, event, hotel, local business, cooking recipe, and properties, e.g. name, address, opening hours, ingredient, together with the structure of the HTML code around the annotations can be used to train information extraction methods to recognize the same type of information in webpages that do not contain such annotations. For instance, the annotation of the product price *69,99 Euro* within an HTML page provides the learning algorithm with an example of the structure and unit of measurement of price values as well as an example of the HTML structures that are used around price values on this page.

A successful example of an information extraction system that employs schema .org annotations as training data is the work of Foley et al. [2015]. The purpose of their system is to discover data about local events, such as small venue concerts, theatre performances, garage sales, and movie screenings on webpages. To train their system, they use event data from webpages which are annotated using the schema.org event properties *name*, *date*, *time*, and *location*. They evaluate their method on 700 million webpages from the ClueWeb12 corpus. Using 217,000 semantically annotated events as supervision, they are able to double recall at a precision level of 85%.

In the work of Ristoski and Mika [2016], the authors propose an approach for enriching product ads with product features extracted from semantically annotated

Task	Topical domain	Source	Usage of semantic annotations	Training size	Appears in
Relation extraction	Multiple	Freebase relations	NO	1.8M	[Mintz et al., 2009]
Relation extraction	Multiple	Webpages from ClueWeb	NO	120M (by 2015)	[Mitchell et al., 2018]
Information extraction	Events	Schema.org event data	YES	217K	[Foley et al., 2015]
Product categorization	Products	Schema.org product data	YES	9.4K	[Meusel et al., 2015]
Feature extraction	Products	HTML product annotations	YES	∼30K	[Ristoski and Mika, 2016]
Product categorization	Products	Schema.org product data	YES	765K	[Brinkmann and Bizer, 2021]

Table 4.6: Related works using distant supervision from the Semantic Web.

product offers on the web. The structured products are used as distant supervision for training feature extraction models, which can then extract attribute-value pairs from unstructured product descriptions. The extracted key-value pairs are used in a two-fold manner: first for identifying matching products to specific product ads and second for enriching the ads with the extracted data.

A series of publicly available information extraction evaluation datasets that were built using schema.org annotations was compiled by Meusel and Paulheim [2014] for the information extraction challenge of the Linked Data for Information Extraction (LD4IE) workshops in 2014 and 2015. The dataset of the LD4IE Challenge 2014 consists of webpages containing Microformats-hCard annotations describing contact information of persons and organizations. The goal of the challenge was to extract such contact information from pages without annotations. The dataset of the LD4IE Challenge 2015 [Meusel and Paulheim, 2015a] consists of HTML pages that contain schema.org annotations describing music recordings, persons, cooking recipes, restaurants, and sports events. This dataset was extracted from the December 2014 version of the Common Crawl. Altogether, the pages originate from 7,300 different websites. The goal of the challenge was to extract information on the above-mentioned domains from pages without annotations.

Relation Extraction Relation extraction is a specific information extraction task that aims at finding relations between entities and has been often tackled with distant supervised learning [Mintz et al., 2009; Mitchell et al., 2018]. Mintz et al. [2009] use entity pairs that are linked with some relation in the Freebase knowledge base as distant supervision for training a relation extraction machine learning model. They are able to extract 1.8 million relation instances, which they use as training data for a relation extraction machine learning model. The trained model is able to extract 10 thousand instances of 102 distinct relations with a precision of 67.2%. For the same application domain, Mitchell et al. [2018] developed NELL, a *Never Ending Language Learner model*. NELL automatically and continuously extracts relations from ClueWeb, a collection of billions of webpages, and adds them to the NELL knowledge base. The relations of the curated knowledge base

are exploited as training examples for extracting additional relations in a cyclic manner. By 2015, the NELL system was able to extract 120 million relations.

Product Categorization Schema.org product annotations have also been exploited for the product categorization task [Meusel et al., 2015]. More specifically, the annotated schema.org/Product/category values are used as features for training a classifier to predict product category labels given product descriptions with no category information. The model reaches an accuracy of 56%, given that the category information is combined with other properties such as the name of the product.

Brinkmann and Bizer [2021] tackle the task of hierarchical product categorization by exploiting schema.org/Product entities to pre-train a ROBERTa transformer model. In the pre-training phase, the values of the properties title, description, and category of schema.org/Product offers are used in order to inject domain-specific knowledge into the model. Next, the model is fine-tuned on the hierarchical product categorization task by exploiting manually labeled product offers. The authors show that using schema.org product annotations in the pre-training phase leads to an improved performance in terms of weighted F1 score by 1.22%.

4.7 Discussion and Conclusion

In this chapter, we gave an overview of different markup formats and the schema.org vocabulary used for semantically annotating web content. Next, we explored the adoption trends of semantic annotations in the period 2012 to 2020 and their potential as a source of distant supervision for entity resolution tasks. We could identify that although there is a constantly growing number of websites using semantic annotations, not all markup formats share the same growth trends. We showed that Microdata and JSON-LD are the most popular in terms of adoption markup formats and are widely used together with the schema.org vocabulary.

We conducted a use-case-specific analysis by presenting an overview of the vocabulary, search engine annotation recommendations, and adoption growth of the schema.org Product and LocalBusiness classes. We observed that although schema.org provides terms for each one of those classes which can be used for annotating unique entity identifiers, there exist large differences concerning the adoption of such properties among the analyzed classes. Those differences can be mainly attributed to the annotation recommendations from large search engines such as Google and Yahoo. For example, we observed that more than 55% (1.2 million) of the websites including product annotations in 2020, used at least one schema.org identifier property such as *sku*. In contrast less than 2.5% (14 thousand) of the websites with local business annotations were found to also include identifier properties, such as *taxID* and *vatID*.

By grouping the identifier values using simple overlap, we can obtain thousands of matching record pairs for both of the analyzed classes, which in terms of

size can be considered a rich source of supervision. However, given the limited website adoption of identifying properties for the local business-related data, we expect the grouped records to have a lower level of heterogeneity in comparison to the product-related grouped records, which derive from many more websites. Additionally, we showed that a combination of alternative identifying properties, such as address and telephone number, can be used to group local business markedup entities. However, exploiting such properties comes at a higher pre-processing cost. In summary, we conclude that for both use-cases, semantic annotations show a considerable potential for generating training sets for entity resolution tasks in terms of the amount of distantly labeled record pairs, while there exist significant differences among the classes in terms of expected data heterogeneity and pre-processing effort.

Chapter 5

The WDC Product Corpus for Entity Resolution

Product-related semantic annotations have a good potential to be used as a source of distant supervision for training entity resolution models targeting product matching tasks, as we saw in the previous chapter. This is due to the following two reasons: First, products are naturally accompanied by identifiers, while more and more webmasters semantically annotate those, following the recommendations of the main search engines. Second, there exists a large number of groups of product offers sharing the same annotated identifiers. These groups can be used to derive matching and non-matching pairs of offers from multiple websites, which can serve as labeled data for training entity resolution models. The trained models can, in turn, be applied to unseen product-related record pairs without any identifiers and predict matching relations.

The noisy nature of the Web, in combination with the different levels of knowledge and understanding of semantic vocabularies from the side of the webmasters, lead to errors in the annotated data or misuse of the vocabulary terms. An overview of frequent semantic annotation errors is presented by Meusel and Paulheim [2015b]. Thus, semantic annotations need to be cleaned before they can be used for generating training data. For example, looking at the 20 most common schema.org/productID, schema.org/gtin, and schema.org/sku values of the WDC 2017 schema.org/Product corpus, we can detect the following noisy identifiers: "stock:" with 195 thousand occurrences, "as_shown" with 63 thousand occurrences and "00000000000000" with 7.5 thousand occurrences. Such frequently occurring identifier values can lead to erroneously grouping offers referring to different real-world products. Additionally, webmasters do not always consistently use the schema.org vocabulary and therefore provide product identifiers for entities of types other than Product or Offer. For example, in the WDC 2017 schema.org/Product corpus, we find 612 and 129 thousand occurrences of the productID property assigned to markedup entities of types *IndividualProduct* and *SomeProducts*, respectively. Both of those classes are valid schema.org subclasses

of the class schema.org/Product, and can therefore contribute to the grouping of markedup entities and the curation of training data.

In this chapter, we consider different annotation strategies for product-related data and develop a pipeline for cleansing and grouping semantically annotated product offers describing the same real-world product, which relies on the overlap of their product identifiers. The output of this pipeline is the *WDC Product Corpus* for entity resolution, which is the third contribution of this work [C3]. The WDC Product Corpus comprises 26.5 million records of product offers described with multiple attributes such as name, description, price, and product category. The product offers are grouped in 16.3 million clusters. Combining pairwise the product offer records of the same clusters, we can generate the largest training set in terms of size and heterogeneity for entity resolution so far, with more than 40 million matching record pairs deriving from more than 79 thousand websites. In this chapter, we provide profiling information on the WDC Product Corpus. Additionally, we evaluate its quality in a twofold manner: First, by manually validating for a subset of the grouped product offers whether they refer to the same real-world product or not. Second, by using record pairs of the same clusters (matching) and of different clusters (non-matching) to train traditional symbolic as well as subsymbolic entity resolution models. For evaluating the performance of the entity resolution models, we build a gold standard with manually verified matching and non-matching product offers.

The contributions of this chapter are summarized as follows:

- We exploit semantically annotated product identifiers for grouping matching records of product offers and curate the WDC Product Corpus. By combining pairwise the grouped product offers, we generate the largest and most heterogeneous, in terms of the number of data sources, training dataset for entity resolution produced so far.
- We identify different types of errors appearing in product-related semantic annotations and build a cleansing pipeline for reducing the noise of the WDC Product Corpus.
- We evaluate the cleanliness of the WDC Product Corpus by manually verifying for a subset of the grouped offers whether they refer to the same real-world product.
- We evaluate the training quality of the WDC Product Corpus using symbolic and subsymbolic entity resolution methods and a manually curated gold standard.

This chapter is structured into five sections. In Section 5.1, we present the pipeline which we execute for curating the WDC Product Corpus for entity resolution and discuss in detail the cleansing and feature extraction steps. Section 5.2 presents the profiling statistics of the WDC Product Corpus. In Section 5.3, we

evaluate the cleanliness and training quality of the corpus and present the results of baseline experiments using a subset of the matching and non-matching record pairs retrieved from the corpus as training data. In Section 5.4, we compare the WDC Product Corpus to other benchmark entity resolution tasks and discuss related works that have used the corpus for training entity resolution models. Finally, Section 5.5 summarizes the main findings of the chapter.

The methodology for curating the WDC Product Corpus and its profiling statistics have been published in the Companion Proceedings of the 2019 World Wide Web Conference [Primpeli et al., 2019]. The evaluation of the quality of the WDC Product Corpus is joint work with Ralph Peeters and has been partially published in the Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics [Peeters et al., 2020b]. The categorization of the corpus is joint work with Helene Bechtold and has been published in her Master's thesis [Bechtold, 2019]. The WDC Product Corpus is available for public download.[1]

5.1 Curation Pipeline

In this section, we outline the details of the pipeline which we execute to curate the WDC Product Corpus. The curation workflow consists of a cleansing and an attribute extraction pipeline. The cleansing pipeline allows to gather, cleanse and group records of product offers with semantically annotated identifiers into clusters representing the same real-world products. Hereon, we will refer to the semantically annotated product offers as *offers*. The attribute extraction pipeline assigns to each offer of the corpus a set of product-related attributes, such as product name and product category.

5.1.1 Cleansing Pipeline

We use the Web Data Commons schema.org/Product data corpus November 2017,[2] containing 809 million schema:Product and schema:Offer markedup entities, as a starting point, which is extracted from the October 2016 Common Crawl web corpus.[3] In order to overcome syntactic errors in annotations and the inconsistency of annotation practices of different webmasters, we develop a cleansing pipeline. Figure 5.1 gives an overview of the cleansing pipeline and the amount of resulting offers and clusters in the WDC Product Corpus after each cleansing step. In the following, we provide the methodological details of each cleansing step.

Filtering of Offers with Annotated Identifiers The schema.org vocabulary provides multiple properties for annotating product identifiers, which were presented and compared in Section 4.5.1. According to the schema.org documen-

[1] http://webdatacommons.org/largescaleproductcorpus/index.html

[2] http://www.webdatacommons.org/structureddata/2017-12/stats/schema_org_subsets.html

[3] https://commoncrawl.org/2016/11/october-2016-crawl-archive-now-available/

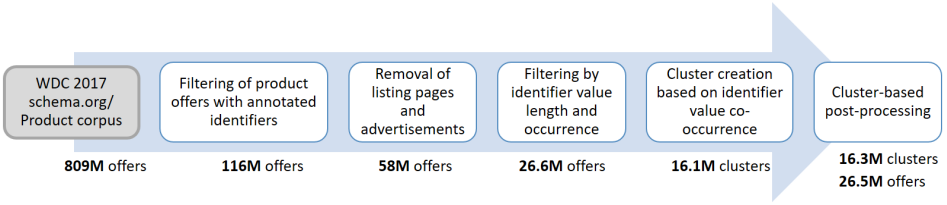


Figure 5.1: Cleansing pipeline for curating the WDC Product Corpus.

Table 5.1: Overlap of global and vendor-specific schema.org product identifiers.

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%

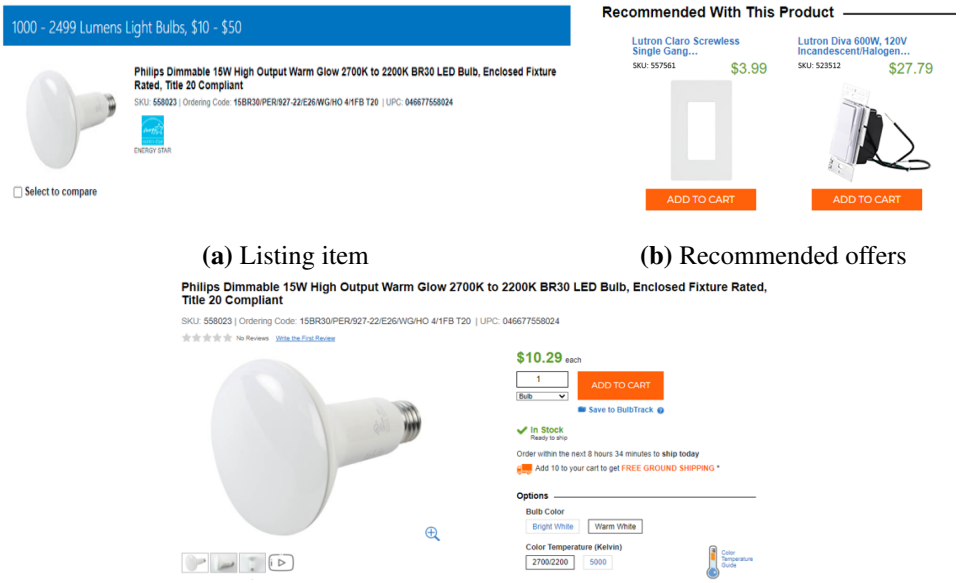
Table 5.2: Most frequent alternative identifier-related schema.org terms.

Schema.org term	#Entities
IndividualProduct/productID	612,260
IndividualProduct/sku	308,096
ProductModel/sku	236,415
SomeProducts/mpn	136,058
SomeProducts/productID	129,608
SomeProducts/gtin13	127,360

tation for the Product[4] and the Offer[5] classes, the *gtin*-based and *mpn* properties should be used to annotate global-scoped identifiers. Vendor-specific identifiers should be marked-up as *sku*, while *productID* and *identifier* can be used either to markup vendor-specific or global identifiers. However, we observe that the identifier-related terms are used inconsistently in many cases, as vendor-scoped terms are often used to annotate global-scoped identifier values. Table 5.1 presents the number of overlapping values marked with vendor-specific and global-scoped schema.org identifier properties. For example, 2,065 product identifier values are found to be marked with both the sku and the gtin8 properties. In total, more than 19% of the distinct global identifier values are annotated using the property *sku*, while the properties *identifier* and *productID* are also often used for the same purpose. Based on this observation, we consider the values of all of the following properties for the grouping step: *gtin8, gtin12, gtin13, gtin14, mpn, sku, identifier,* and *productID*.

Additionally, we notice that 6% of the websites annotating offers with identifier values use schema.org terms that are either invalid or do not conform with the search engines' recommendations for annotating products, as described in Section 4.5.1. This is similar to the observations of Meusel and Paulheim [2015b]. For example, a frequent pattern is the usage of alternative schema.org types denoting product offer identifiers, such as *IndividualProduct/productID* or *ProductModel/sku*. We present such identifier-related terms and their frequency in Table 5.2. Such terms reveal identifying information for an offer and therefore, we

[4]https://schema.org/Product
[5]https://schema.org/Offer



(a) Listing item **(b)** Recommended offers

(c) Detail page

Figure 5.2: Example of a product offer listing and advertisement.

do not want to ignore them during grouping. We capture such offers by applying the following regular expression pattern on the schema.org terms used to describe them: *.*/(gtin8|gtin12|gtin13|gtin14|sku|mpn|identifier|productID)*. This results in 116 million offers being selected from the WDC 2017 schema.org/Product corpus.

Removal of Listing Pages and Advertisements Figure 5.2 shows an example of a listing item referring to a light bulb product offer (Figure 5.2a), a part of its corresponding detail page (Figure 5.2c), as well as two recommended offers appearing on the detail page (Figure 5.2b). Product summaries included in listings and advertisements are often repetitive and provide information that is either fully contained in the product's detail page (Figure 5.2a) or incomplete and thus not enough to disambiguate a product (Figure 5.2b). We consider that the limited and repetitive texts of listing and advertisement offers do not contribute to learning powerful classification models due to their triviality or even mislead the learning process due to their ambiguity. Therefore, we want to include only the comprehensive information about a product from its detail page in the WDC Product Corpus and not the summaries of this information found on listing pages or as advertisements on other detail pages.

To detect listing pages and advertisements, the following heuristic is applied: First, we group the offers that appear in the same webpage. For each group that contains more than one offer, we calculate the standard deviation $\sigma(O_{length})$ and the median $median(O_{length})$ of the length of the concatenated title and descriptions of its offers. If we find that the standard deviation of a group is small, i.e.

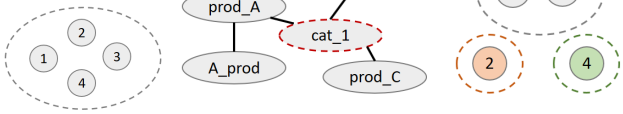
the descriptions of all offers of the same webpage have little variation, we assume that the webpage is a listing one and exclude all offers appearing on it. The standard deviation boundary for the classification decision is empirically set as $0.2 \times median(O_{length})$. If the standard deviation $\sigma(O_{length})$ is smaller than the defined boundary, we classify as listings and filter out all offers of the group. Otherwise, we proceed to a second step and check for every offer if there exists a semantic connection to other offers of the same group using the terms schema:RelatedTo and schema:SimilarTo. If such connection exists, we classify the offer as an advertisement and filter it out. We measure the quality of our heuristic using a manually annotated test set of 80 offers from different webpages with a ratio 70%:30% of listingsORads and non-listingsORads, respectively. We calculate the F1 score on the positive class listingsORads to be 94.8%. This cleansing step removes 49% of the offers, leaving 58 million non-listing or advertisement offers in the WDC Product Corpus.

Filtering by Identifier Length and Occurrence In the next pre-processing step, the annotated 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*, and the fact that short identifiers are more likely to introduce noise in the grouping step, we filter out all offers having identifiers that are shorter than eight characters. Additionally, offers whose identifier values completely consist of alphabetical characters are removed. Finally, we observe that a considerable amount of websites use the same identifier value to annotate all their offers, likely due to an error in the script generating the pages. We detect these websites and remove their offers from the corpus. After these filtering steps, 26.6 million offers remain in the WDC Product Corpus.

Grouping and Post-processing We group the remaining 26.6 million offers into 18 million clusters using their identifier values. It happens that single offers contain multiple alternative identifiers referring to the same product, e.g. gtin8 and gtin12, or gtin12 and mpn. We use this information transitively to merge clusters referring to the same product, which results in a reduction of the number of clusters to 16 million.

We also note that some websites annotate offers with identifiers referring to product categories, such as UNSPSC numbers, in addition to product-specific identifiers. For detecting such cases, we examine the structure of the identifier co-occurrence graph within each cluster. Figure 5.3 presents how such an identifier co-occurrence graph can be derived: Given the overlap of the offers' identifier values in Figure 5.3a and considering graph transitivity, all offers are grouped in one cluster, as shown in Figure 5.3b. Representing every distinct identifier value appearing in the clustered offers as a vertex and denoting co-occurrence relations with edges results in the graph of Figure 5.3c. For example, there is an edge between

offer_id	schema.org identifiers	
1	prod_A	cat_1
2	prod_B	cat_1
3	prod_A	A_prod
4	prod_C	cat_1



(a) Offers with identifiers **(b)** Clustered offers **(c)** Identifier co-occurrence graph **(d)** Post-processed clustered offers

Figure 5.3: Example of deriving the identifier co-occurrence graph and post-processing the clustered offers.

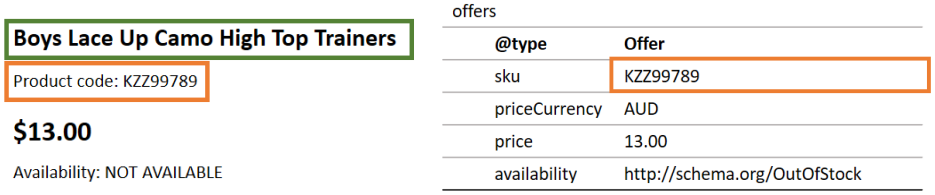
the vertices representing the values *prod_A* and *cat_1*, as they co-occur in the offer with id 1, considering Figure 5.3a.

We detect identifier values that refer to product categories rather than single products by measuring the degree and the clustering coefficient for each vertex in the identifier co-occurrence graph. The degree of a vertex is calculated as the number of edges adjacent to it. The clustering coefficient of a vertex is calculated as the fraction of pairs of the vertex's neighbors which are adjacent to each other. A high degree of a vertex in combination with a small clustering coefficient in the identifier co-occurrence graph denotes that an identifier value often co-occurs with other values that, however, never appear together. In the example of Figure 5.3, the vertex representing the value *cat_1* occurs together with the values *prod_A*, *prod_B*, and *prod_C*, among which, however, there is no edge. This indicates that the vertex *cat_1* likely represents a product category identifier rather than a single product identifier and should be removed. With the removal of this vertex, the clustered offers of Figure 5.3b are split into three clusters, as shown in Figure 5.3d.

We discover that vertices having a degree larger than ten and a clustering coefficient smaller than 0.2, tend to represent product categories rather than single products. Therefore, we remove the identifier values corresponding to those vertices and split the clustered offers, similar to the example above. This removes 90,073 offers and results in the creation of 199,139 additional clusters. Therefore the final WDC Product Corpus contains 26,507,033 offers deriving from 79,126 websites and grouped into 16,391,439 clusters.

5.1.2 Attribute Extraction Pipeline

Product identifiers allow offers to be grouped into clusters representing the same real-world product. Deriving pairs of offers within and across clusters allows us to build a training set of distantly labeled matching and non-matching pairs of offers. In order to be able to train entity resolution models that can predict if two product offers without identifiers are matching or non-matching, the offers of the training set need to be accompanied by product-related attributes. We obtain such attributes using the schema.org property values of the markedup offer entities. Additionally,



(a) Product offer on a webpage. **(b)** Corresponding schema.org annotations.

Figure 5.4: Example of a product offer annotation with two schema.org entities.

we further enrich the offers by exploiting the content of the specification tables found in their corresponding webpages as well as by assigning product category information. Below, we present the steps for extracting product-related attributes.

Leveraging Entity Relations

For the majority of the clustered offers, we directly extract the most frequently appearing schema.org properties, which reveal part of the product's identity and can be therefore used for matching: *name, title, description, brand, price, priceCurrency,* and *manufacturer*. We merge the *name* and *title* attributes into a single *title* attribute. In addition, we extract the values of the properties *image* and *availability*. The image information can be used as an additional matching signal in general but will not be exploited in our work, as we solely focus on textual attributes. The availability information can be used as distant supervision in different applications other than entity resolution, such as feature extraction.

We find that 20% of the offers in the corpus do not contain highly relevant to the entity resolution task properties, such as *title* and *description*. This originates in the annotation practice of describing a product offer using two separate markedup entities. An example of this annotation practice is shown in Figure 5.4. The example product comprises two markedup entities which are connected with the Product/offers property: the parent markedup entity, which is of type schema.org/Product and is annotated with the name property, and the child markedup entity of type schema.org/Offer, which contains the sku identifier, the price, and the availability information. Given our cleansing pipeline described in Section 5.1.1, the first markedup entity would be filtered out in the first cleansing step as it does not contain any identifier information, while the second markedup entity would end up in the corpus without, however, having any title information.

We identify such cases by detecting offers in the WDC Product corpus connected with the Product/offers property to any offers of the input corpus, i.e. the WDC 2017 schema.org/Product corpus, and merge them. By merging the two offers of Figure 5.4, we enrich the offer with "sku:KZZ99789" with the title value "Boys Lace Up Camo High Top Trainers". This step reduces the number of offers in the WDC Product Corpus having no title or description to less than 3%.

Brand	Philips Lighting
Product Line	Warm Glow
Model Number	15BR30/PER/950/E26/DIM/HO 4/1FB T20
Energy Used	15 Watts
Incandescent/Halogen Equivalent	100 Watts
Volts	120 Watts
Base	Medium (E26)

```
<html>
 <table cellspacing="0" cellpadding="0" border="0" style="border-collapse:collapse;">
    <tr>
        <td class="specAttributeName"><p class="attributeName">Brand</p></td>
        <td itemprop="brand">Philips Lighting</td>
    </tr><tr>
        <td class="specAttributeName"><p class="attributeName">Product Line</p></td>
        <td>Warm Glow</td>
    </tr><tr>
        <td class="specAttributeName"><p class="attributeName">Model Number</p></td>
        <td>15BR30/PER/950/E26/DIM/HO 4/1FB T20</td>
    </tr><tr>
        <td class="specAttributeName"><p class="attributeName">Energy Used</p></td>
        <td>15 Watts</td>
    </tr><tr>
        <td class="specAttributeName"><p class="attributeName">Incandescent/Halogen Equivalent</p></td>
        <td>100 Watts</td>
    </tr><tr>
        <td class="specAttributeName"><p class="attributeName">Volts</p></td>
        <td>120 Watts</td>
    </tr><tr>
        <td class="specAttributeName"><p class="attributeName">Base</p></td>
        <td>Medium (E26)</td>
        </tr>
    </table>
</html>
```

Figure 5.5: An example specification table and its HTML content.

Detection and Extraction of Specification Tables

Product pages often contain specification tables describing the product in the form of key/value pairs. Figure 5.5 presents an example specification table along with its HTML content. The example specification table corresponds to the product offer of Figure 5.2c. The structured data provided within the specification table are beneficial for matching product-related records [Kannan et al., 2011; Petrovski et al., 2014]. In order to differentiate between specification and non-specification HTML tables, we consider the findings of Qiu et al. [2015] and Petrovski and Bizer [2017] and apply a table detection heuristic in an unsupervised fashion. The heuristic extracts certain HTML table attributes, calculates their values, and compares the calculated values to pre-defined upper and lower thresholds. If the value of at least one of the attributes exceeds the upper threshold or is below the lower threshold, then it is classified as a non-specification table. If the values of all attributes are within the threshold ranges, the HTML table is classified as a specification table. The following HTML table attributes and threshold ranges are used for the heuristic: ratio of numerical over alphabetical characters (0.08-2.0), ratio of nested elements that are row or columns elements, i.e. *<td>, <tr>, <th>* (0.8-1.0), number of columns (1-6), number of rows (3-50), and average length of text per row (15-100). The HTML table attributes values of the table in Figure 5.5 fall within the specified ranges for all attributes and is therefore classified as a specification table: ratio of numerical over alphabetical characters =0.2, ratio of nested elements that are row

or columns elements =1.0, number of columns =2, number of rows =7, and average length of text per row =26.4. We evaluate the specification table detection heuristic based on a manually annotated gold standard of 455 HTML tables found in 27 webpages. The annotation process is conducted by a single annotator who receives the complete HTML page and annotates every HTML <*table*> element found on the page as *specification* or *non-specification*. Based on our gold standard, the applied heuristic reaches an F1 of 78% on the positive class *specification*.

After an HTML table has been classified as a specification table, we extract all its textual elements, concatenate them into one string value and add it as an attribute of its corresponding offer. For the table of Figure 5.5, the extracted specification table attribute is *specTableContent: {Brand Philips Lighting Product Line Warm Glow Model Number 15BR30/PER/950 /E26/DIM/HO 4/1FB T20 Energy Used 15 Watts Incandescent/Halogen Equivalent 100 Watts Volts 120 Watts Base Medium (E26)}*. Additionally, to enable the corpus usage as distant supervision for feature extraction applications, we use the two column-heuristic from [Qiu et al., 2015] only in the case of two-column specification tables. The heuristic extracts the values of the left column as attribute names and the values of the right column as attribute values. If applicable, the attribute name/value pairs are organized as a dictionary and added as an attribute to the corresponding offer. For the table of Figure 5.5, the extracted name/value pairs attribute is *keyValuePairs:{Brand: Philips Lighting, Product Line: Warm Glow, Model Number: 15BR30/PER/950 /E26/DIM/HO 4/1FB T20, Energy Used: 15 Watts, Incandescent/Halogen Equivalent: 100 Watts, Volts: 120 Watts, Base: Medium (E26)}*.

Categorization of Offers

E-shops use a wide range of different categorization schemata to present their offers. Product category information is crucial to the entity resolution task. It can be used as part of the blocking strategy to reduce the candidate record pairs to the ones belonging to the same category. This requires that the offers are mapped to a pre-defined, unique categorization taxonomy, a task faced by many aggregators in e-commerce. The schema.org vocabulary contains the *category* property, which is indented for capturing the product category of a markedup product entity. However, only 2% of the offers in the WDC Product Corpus contain category information, while the category values are not consistent among the different webmasters. In this section, we aim to assign consistent product category labels to the offers of the WDC Product Corpus. We treat the problem of product categorization as a supervised multi-class classification task that comes with the following requirements: (i) a product category taxonomy, (ii) a gold standard of annotated offers with their respective product category labels, and (iii) a multi-class classification model. In the following, we present the methodological details and evaluation results of the product categorization task.

Product Category Taxonomy We create a flat product category taxonomy by comparing and merging the first-level categories of three existing product category taxonomies: the Amazon taxonomy [McAuley et al., 2015],[6] the Google product taxonomy[7] and the UNSPSC taxonomy.[8] While the Amazon product taxonomy is a flat set of 24 product categories, the Google and UNSPSC taxonomies are hierarchical and comprise 156,478 and 6,000 categories, respectively.

Table 5.3 presents the categories of our defined taxonomy and their mappings to the Amazon, Google, and UNSPSC categories, including the depth of the corresponding category path for the case of the hierarchical taxonomies. We split the broad category of Clothing, Shoes & Jewelry into four categories: Clothing, Shoes, Jewelry, and Luggage & Travel Gear. Similarly, we split the Electronics category into three categories: Camera & Photo, Computers & Accessories, and Other Electronics. Comparing the flat Amazon taxonomy to the more fine-grained Google and UNSPSC taxonomies, we create a product category in our taxonomy if the mapped Google and UNSPSC categories are first-level categories. For example, we directly add in our taxonomy the Automotive category of the Amazon taxonomy as it corresponds to the Vehicles & Parts first-level category of the Google taxonomy and the Commercial, Military, Private Vehicles first-level category of the UNSPSC taxonomy. For the Amazon categories mapped to deeper levels in the Google and UNSPSC taxonomies, the respective first-level categories are compared and we manually assess whether the sub-categories can be merged. For example, the Health & Beauty category of our taxonomy considers cosmetics, health, beauty, and personal care products which follow under the merged sub-categories of the Google and UNSPSC taxonomies. This results in 23 product categories as listed in the right column of Table 5.3. Finally, we add two more categories to our taxonomy: *Others*, containing products that could not be assigned to any of the defined categories, and *Not Found*, which is assigned to product offers with unclear descriptions.

Gold Standard for Product Categorization In order to enable the training and evaluation of a multi-class classifier for the task of offer categorization, we create a gold standard from a subset of the offers in the WDC Product Corpus. The annotation is conducted by one annotator who manually inspects the schema.org properties and specification table content extracted in the previous steps for a subset of the offers in the WDC Product Corpus. After inspection, she assigns one of the category labels of the defined taxonomy to each offer. Offers that do not fit into any of the defined categories are assigned the category label *Others*. In addition, offers whose description was unclear are assigned the label *Not Found*. In total, 24,689 offers deriving from 2,115 clusters are annotated. The distribution of the number of offers, clusters, and attribute density in the categorization gold standard is presented in Table 5.4.

[6] http://jmcauley.ucsd.edu/data/amazon/

[7] https://www.google.com/basepages/producttype/taxonomy.en-US.txt

[8] https://wwwcfprd.doa.louisiana.gov/osp/lapac/vendor/commodityTree.cfm

Table 5.3: Mapping of different product category taxonomies to the final taxonomy.

Amazon taxonomy	Google taxonomy	UNSPSC taxonomy	Final taxonomy
Automotive	Vehicles & Parts (1)	Commercial, Military, Private Vehicles (1)	Automotive
Baby	Baby & Toddler (1)	Baby & Toddler Furniture & Accessories (3)	Baby
Beauty Health & Personal Care	Cosmetics (3) Health & Beauty (1)	Cosmetics (4) Personal Care Products (2)	Health & Beauty
Books	Books (2)	Printed Media (2)	Books
CDs & Vinyl	Music & Sound Recordings (2)	Music on Tape or CD (4)	CDs & Vinyl
Cellphones & Accessories	Mobile Phone Accessories (4)	Mobile Phones (4)	Cellphones & Accessories
Clothing, Shoes & Jewelry	Clothing (2) Jewelry & Watches (2) Shoes (2) Luggage & Bags (1)	Clothing (2) Jewelry (2) Footwear (2) Luggage & Handbags and Packs & Cases (2)	Clothing Jewelry Shoes Luggage & Travel Gear
Digital Music	-	-	-
Electronics	Cameras & Optics (1) Computers (2) Electronics (1)	Photographic, Filming or Video Equipment (2) Computer Equipment & Accessories (2) Consumer Electronics (2)	Camera & Photo Computers & Accessories Other Electronics
Grocery & Gourmet Food	Food, Beverages & Tobacco (1)	Food, Beverages & Tobacco Products (1)	Grocery & Gourmet Food
Home & Kitchen Patio, Lawn & Garden	Home & Garden (1)	Furniture & Furnishings (1) Agriculture, Forestry & Garden Handtools (3)	Home & Garden
Movies & TV	DVDs & Videos (2)	Motion Pictures on DVD (4)	Movies & TV
Musical Instruments	Musical Instruments (3)	Musical Instruments & Parts & Accessories (2)	Musical Instruments
Office Products	Office Supplies (1)	Office Equipment, Accessories & Supplies (1)	Office Products
Pet Supplies	Animal & Pet Supplies (1)	Domestic Pet Products (2)	Pet Supplies
Sports & Outdoors	Sporting Goods (1)	Sports & Recreational Equipment (1)	Sports & Outdoors
Tools & Home Improvement	Hardware (1)	Tools & General Machinery (1)	Tools & Home Improvement
Toys & Games	Toys & Games (1)	Toys & Games (2)	Toys & Games
Video Games	Video Games (2)	Computer Game or Entertainment Software (3)	Video Games

Table 5.4: Statistics of the product categorization gold standard.

Category	# Offers	# Clusters	Title	Description	Brand	Manufacturer
Automotive	1,446	78	100%	66%	63%	5%
Baby	918	89	100%	69%	47%	12%
Books	656	89	100%	35%	62%	7%
CDs_and_Vinyl	604	90	95%	36%	19%	5%
Camera_and_Photo	968	91	100%	86%	35%	19%
Cellphones_and_Accessories	1,377	90	100%	67%	70%	2%
Clothing	3,242	232	100%	14%	1%	0%
Computers_and_Accessories	4,753	162	100%	97%	94%	1%
Grocery_and_Gourmet_Food	561	76	100%	80%	20%	7%
Health_and_Beauty	506	73	99%	52%	23%	10%
Home_and_Garden	554	78	100%	69%	25%	4%
Jewelry	767	56	100%	79%	3%	1%
Luggage_and_Travel_Gear	812	72	99%	68%	31%	6%
Movies_and_TV	643	75	98%	91%	11%	9%
Musical_Instruments	570	83	99%	94%	35%	11%
Office_Products	659	57	100%	51%	6%	7%
Other_Electronics	687	87	100%	81%	33%	8%
Others	10	7	100%	70%	40%	10%
Pet_Supplies	610	77	100%	97%	3%	6%
Shoes	555	68	100%	48%	23%	2%
Sports_and_Outdoors	818	71	100%	85%	86%	1%
Tools_and_Home_Improvement	783	85	100%	68%	57%	17%
Toys_and_Games	586	89	100%	34%	21%	9%
Video_Games	584	82	100%	86%	48%	7%
not found	1,020	58	97%	26%	3%	3%
Total/ Average*	**24,689**	**2,115**	**99.58%***	**65.48%***	**42.78%***	**4.87%***

Product Categorization Model We split the categorization gold standard into train and test with a ratio of 80%:20% and compare different classification models. We build different product categorization models by executing a workflow of preprocessing, feature extraction, and model selection steps, which we explain below.

For each offer in the gold standard the following attributes are considered: title, description, brand, manufacturer, HTML page content, and specification table, if existing. For each attribute, the contents are filtered by stopwords and punctuation characters, lowercased, and tokenized.

We curate the feature vector of each offer by applying a bag-of-words feature creation model and compute tf-idf weights for different combinations of attributes, e.g. only schema.org property values or schema.org property values and specification tables. The hyperparameters for the feature vector creation, such as the ngram level of splitting the tokens and pruning of tokens given their frequency, are optimized using grid search with 5-fold cross-validation. Finally, the following attribute combinations are considered for feature creation and evaluated: (i) *schema.org*, which only uses the values of the existing schema.org properties, (ii) *schema.org+spec.tables*, which additionally exploits the specification table content if existing and (iii) *schema.org+html content*, which additionally exploits the content of the webpage in which the offer was found.

We evaluate the following classification algorithms: SVM, logistic regression,

Table 5.5: Product categorization results.

Algorithm	Attributes	Micro-F1
SVM	schema.org	0.86
	schema.org + spec. tables	0.85
	schema.org + html content	0.81
Logistic regression	schema.org	0.85
	schema.org + spec. tables	**0.87**
	schema.org + html content	0.81
Naive Bayes	schema.org	0.84
	schema.org + spec. tables	0.84
	schema.org + html content	0.73
Decision tree	schema.org	0.66
	schema.org + spec. tables	0.66
	schema.org + html content	0.67

Naive Bayes, and decision tree. Table 5.5 shows the F1 scores on the test set for each algorithm and attribute combination. We observe that the logistic regression model trained on the features generated from the schema.org properties and specification table content achieves the best micro-averaged F1 score of 0.87. Therefore, we select this model for profiling the offers of the WDC Product Corpus on product category level.

5.2 Profiling

In this section, we present the profiling statistics of the WDC Product Corpus and its English language subset, which is created by selecting all offers from pages having the suffixes: com, net, co.uk, and org. Thereafter, we will refer to the English subset as *WDC English Product Corpus*. Figure 5.6 shows an example offer of the WDC Product Corpus.


```
{
	"id":4615205,
	"cluster_id":1072350,
	"category":"Computers_and_Accessories",
	"title":"418367-B21 HP 146-GB 3G 10K 2.5 DP SAS HDD",
	"description":"Description:146GB 2.5-inch Serial Attached
		SCSI (SAS)SFF 3G Dual Port Hot-Plug 10K Hard Drive",
	"brand":"HP Enterprise",
	"price":"$29.50",
	"keyValuePairs":{
		"Manufacturer":"BL465c",
		"Category":"Hard Drive",
		"Type":"Hard Drive - Hot-Swap",
		"Capacity":"146GB"
	},
	"specTableContent":" Specifications:  Manufacturer BL465c
		Category Hard Drive Type Hard Drive - Hot-Swap
		Capacity 146GB"
}
```


Figure 5.6: An example offer of the WDC Product Corpus.

Table 5.6: Distribution of offers and matching offer pairs per cluster size.

	# Clusters		# Matching pairs	
Cluster size	WDC Product Corpus	WDC English Product Corpus	WDC Product Corpus	WDC English Product Corpus
1	13,301,842	8,434,389	0	0
2	1,915,909	1,012,220	1,915,909	1,012,220
[3-4]	760,360	400,522	3,026,997	1,552,680
[5-10]	304,379	163,356	5,677,852	3,064,532
[11-20]	63,981	37,562	6,752,150	3,751,124
[21-30]	17,710	10,567	5,374,523	3,185,935
[31-40]	10,863	4,461	6,666,546	2,646,306
[41-50]	6,318	2,504	6,281,387	2,502,691
[51-60]	2,663	1,300	3,978,483	1,972,822
[61-70]	1,378	832	2,891,198	1,750,014
[71-80]	1,058	682	2,960,532	1,899,880
[>80]	4,978	3,999	137,488,518	113,935,797

The WDC Product Corpus contains 26 million offers deriving from 79 thousand websites, grouped into 16 million clusters. The WDC English Product Corpus contains 16 million offers deriving from 43 thousand websites, grouped into 10 million clusters. Table 5.6 shows the distribution of offers per cluster in the two corpora as well as the amounts of matching pairs that can be derived by combining pairwise the offers of each cluster. We observe that small clusters (sizes one and two) account for 92% of the clusters in the WDC Product Corpus. The reasons for the large fraction of small clusters are twofold: First, the long tail distribution of products on the Web, i.e. only a few products are offered by many e-shops, while a large number of products are offered by few e-shops. Second, the limited depth of the CommonCrawl, as only a fraction of the pages of each website is crawled. The WDC English Product Corpus contains over 600 thousand clusters of size three or larger. Despite the large number of small clusters, and disregarding the clusters of size larger than 80, which we expect to be noisy, more than 20 million matching pairs can be derived from the WDC English Product Corpus.

Next, we profile the density of the offer attributes, the extraction of which was analyzed in Section 5.1.2. Table 5.7 shows the distribution of the schema.org properties that are used to describe the offers in the two corpora. Our specification table detection method finds at least one specification table in 24% of the HTML pages contained in the WDC Product Corpus and 17% of the pages of the WDC English Product Corpus. Using the key/value pair extraction heuristic described in Section 5.1.2, we are able to extract ten or more key/value pairs from 73% of the specification tables.

Finally, we profile the distribution of offers and clusters per product category. Considering that the multi-class classifier for the product categorization task is trained on offers with English attribute values, we apply the best performing model

Table 5.7: Density of offer attributes in the WDC Product Corpus.

Attributes	Offers in			
	WDC Product Corpus		WDC English Product Corpus	
	# (in K)	%	# (in K)	%
title	25,281	95.3	15,653	95.1
description	17,215	64.9	11,352	69.0
brand	9,313	35.1	5,645	34.3
image	5,785	21.8	4,348	26.4
price	3,335	12.5	1,977	12.0
price currency	2,971	11.2	1,873	11.3
availability	1,180	4.4	716	4.3
manufacturer	2,024	7.6	1,254	7.6

only to the offers of the WDC English Product Corpus. The profiling results in terms of the relative and absolute amount of offers and clusters per category are presented in the barplot of Figure 5.7. We observe that for most categories the number of offers (orange bars) and clusters (blue bars) are in a similar range. However, this is not the case for the office products category which is assigned to more than 2 million offers belonging to 403.8K clusters. This indicates that the large clusters of the corpus refer to office products. By manually inspecting the classified office product offers, we indeed verify that 64% (1.4 million offers) belong to only 100 clusters with size larger than 1000 offers and derive from websites that offer vector images, such as *www.vectorstock.com*. The fact that these websites typically offer millions of vector images, explains the large number of offers being annotated as office products.

5.3 Quality Evaluation

In the previous section, we profiled the WDC Product Corpus and showed that we can derive a large amount of matching offer pairs. Additionally, we showed that the majority of the offers are accompanied by product-related attributes, such as title and description. In this section, we analyze the quality of the WDC Product Corpus in terms of overall cluster quality as well as training quality. To assess the overall cluster quality, we select a sample of offer pairs from clusters of different sizes as well as categories and manually inspect how many of those refer to different products and are therefore wrong. In order to assess the training quality of the WDC Product Corpus, we filter category-specific clusters and derive training sets from inter-cluster (matching) and intra-cluster (non-matching) offer pairs. We use the derived sets to train baseline entity resolution models. We assess the prediction quality of the trained models using the WDC Product Gold Standard, a manually labeled set of matching and non-matching offer pairs. In the following, we present the details of our two-fold evaluation of the WDC Product Corpus.

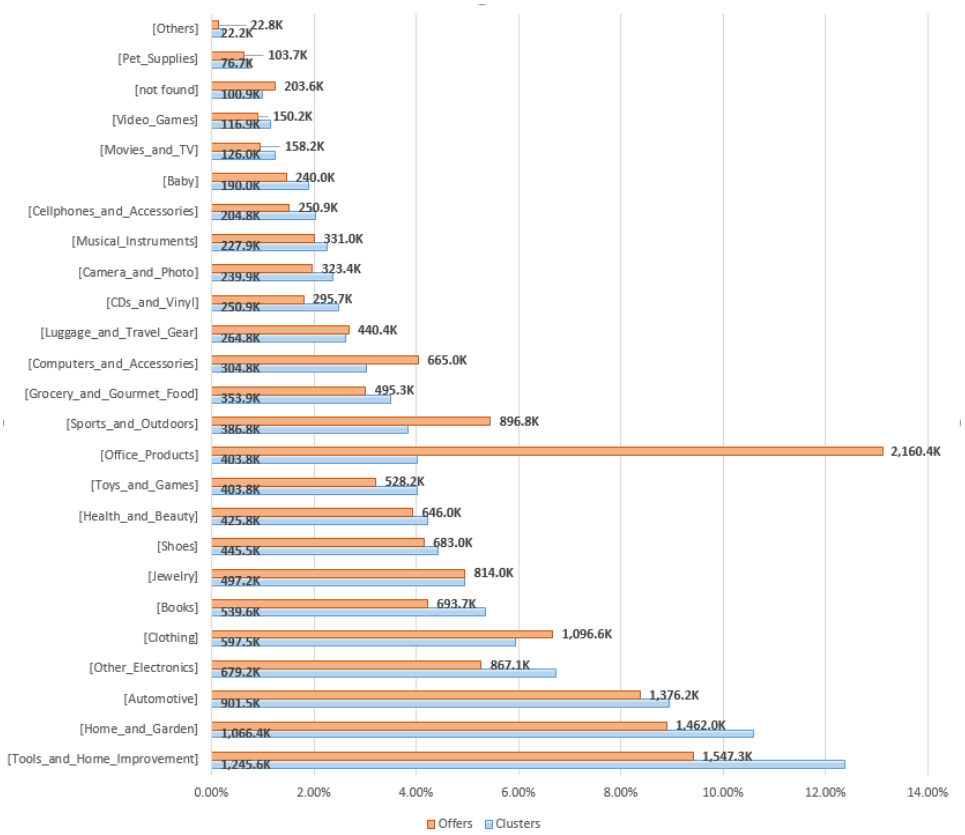


Figure 5.7: Distribution of offers and clusters per category in the WDC English Product Corpus.

5.3.1 Clusters Quality

We assess the level of noise in the clusters of the WDC Product Corpus based on a sample of 900 offer pairs from clusters of different size ranges and product categories. We sample clusters from the following six categories: Cellphones and Accessories, Health and Beauty, Clothing, Home and Garden, Jewelry, and Office Products, and the following cluster size ranges: small (3-5 offers), medium (6-80 offers) and large (>80 offers). From each group size and product category, we randomly sample 50 offer pairs, which we manually annotate as matching or non-matching. The annotation process is conducted by two annotators. Each annotator inspects half of the sampled offer pairs along with their extracted attributes, i.e. schema.org properties, specification tables, and category labels, and assigns matching or non-matching labels. The ambiguous offer pairs are inspected by both annotators. Considering that every sampled offer pair derives from the same cluster, the offer pairs that are annotated as non-matching constitute noise.

Our evaluation shows that 846 offer pairs (94% of the complete sample) refer to the same real-world product and are therefore grouped correctly, while 29 offer

```
{
"id":"t4hg010001",
"cluster_id":16251955,
"title":"signature fragrance beauty tomford"
}

{
"id":"t4hg010001",
"cluster_id":"16251955",
"title":"gift set beauty tomford"
}
```


(a) Error type: wrong identifiers.


```
{
"id":"721405206254",
"cluster_id":700061,
"title":"mazzer super jolly automatic coffee grinder commercial model",
"brand":"mazzer
}

{
"id":"892199001223",
"cluster_id":700061,
"title":"bamix professional gastro 200 immersion hand blender",
"brand":"bamix"
}
```


(b) Error type: wrong grouping.

Figure 5.8: Examples of erroneously clustered offers in the WDC Product Corpus.

pairs (3% of the complete sample) refer to different products and are therefore wrong. For the remaining 25 offer pairs, the annotators are unsure if the offer descriptions refer to the same or different real-world products. We find that the attribute values of 70% of these offer pairs contain less than four tokens, which is the reason for their ambiguity.

Analyzing the origin of the errors, we find two main reasons for erroneously grouping offers that refer to different products: (i) the offers are assigned wrong identifiers, and (ii) our grouping strategy based on identifier value co-occurrence leads to wrongly grouping offers referring to different products. Figure 5.8 presents two examples of erroneously clustered offer pairs. The error shown in Figure 5.8a belongs to the first error type, as the two offers are assigned the same identifier but refer to different real-world products considering their description. Thus, we can assume that either one or both identifier values are wrong. Figure 5.8b belongs to the second error type, as the two offers are grouped in the same cluster even though they have different identifiers. The wrong grouping is due to the indirect co-occurrence of the two identifier values via transitivity, e.g. another offer is assigned both identifier values *721405206254* and *892199001223*, which causes the grouping of offers having these two identifier values under the same cluster.

We observe that 90% of the grouping errors are found in medium and large clusters, while 50% of the wrong identifier errors are found in large clusters. Finally, the errors are almost evenly distributed among the different product categories, which signifies that no product category can be considered noisier than any other.

5.3.2 Training Quality

In this section, we estimate the usefulness of the WDC Product Corpus for training entity resolution models for product matching. We do so by measuring the prediction quality of baseline symbolic and subsymbolic entity resolution methods that can be achieved using matching and non-matching offer pairs derived from the corpus as training data. In order to allow entity resolution methods to be evaluated on completely clean data, we create the WDC Product Gold Standard by manually verifying for a subset of pairs of offers if they refer to the same product or not. Considering that for the subsymbolic methods we use English language embeddings, all offer pairs used for training as well as for building the WDC Product Gold Standard originate from the WDC English Product Corpus. In the following, we describe the curation of the WDC Product Gold Standard and the extraction of training sets from the WDC English Product Corpus. Additionally, we present the experimental setup and evaluation results of different baseline symbolic and subsymbolic entity resolution methods.

WDC Product Gold Standard

We consider offer pairs from the WDC English Product Corpus of four different product categories for building the WDC Product Gold Standard: Watches (a subset of the Jewelry category), Shoes, Computers & Accessories, and Camera & Photo. First, we identify the clusters belonging to the selected product categories. Second, we select 150 related clusters from each product category, preferring clusters having a large diversity among the offers' textual content and a minimum size of seven offers. The large diversity, in this context, refers to offers describing the same product while the Jaccard string similarity of their titles and descriptions varies. This leads to the selection of clusters that contain textually similar as well as less similar offers.

In order to select challenging pairs of offers for the manual verification, we apply the following procedure: From every selected cluster, we pick one offer and concatenate its textual content given by the values of the attributes title, description, brand, and specification table. Similarly to Köpcke et al. [2010], we use the Jaccard similarity metric and the offers' textual content to calculate the similarity scores between the picked offer and the offers of the same cluster (intra-cluster similarity scores) as well as the offers of different clusters (inter-cluster similarity scores). To build matching offer pairs, we select two intra-cluster offer pairs having the highest and lowest similarity scores and add them to the gold standard. To build

Table 5.8: WDC Product Gold Standard profiling.

T=title, D=description, B=brand, S=specification table

Category	# Pairs		% Density			
	Matching	Non-matching	T	D	B	S
computers	300	800	100	82	42	22
cameras	300	800	100	73	25	7
watches	300	800	100	72	15	7
shoes	300	800	100	70	8	2

non-matching offer pairs, we add two to three inter-cluster offer pairs with the highest similarity score and three randomly chosen inter-cluster pairs in the gold standard. We manually verify for all selected pairs if they are really matching or non-matching by comparing the textual content of the offers. If we discover an incorrectly labeled pair, we correct the label.

The resulting gold standard consists of 300 matching and 800 non-matching offer pairs for each product category. Table 5.8 presents the profiling statistics of the WDC Product Gold Standard. As *density*, we denote the ratio of matching and non-matching pairs in the WDC Product Gold Standard for which both offers contain the corresponding attribute.

Training Subsets

We generate category-specific subsets of different sizes, i.e. small, medium, large and xlarge, from the WDC English Product Corpus for training baseline entity resolution models. To build matching pairs, we iterate over all clusters which have offers in the gold standard and retrieve a varying amount of offer pairs, depending on the size of the training set. For building the small training sets we retrieve one pair of offers. For medium, we retrieve three pairs, while for large and xlarge, 15 and 50 intra-cluster pairs are selected. To build non-matching pairs, in a first step, the title similarity of each pair of clusters is calculated using Jaccard similarity over the concatenated titles of the offers. In a second step, for each cluster, the top ten most similar clusters based on this similarity are chosen. For each of the resulting most similar cluster pairs, a varying amount of offer pairs is sampled, depending on the size of the training set. We select 3 pairs for the small training sets, 9 for the medium, 45 for the large, and 150 for the xlarge. This procedure for sampling non-matching pairs ensures that the pairs in the training subsets are similar and allow the classifiers to learn useful patterns for differentiating difficult non-matching offers. During the training subsets creation, we ignore any offer pair that is already in the WDC Product Gold Standard and do not manually verify the correctness of the labels of the offer pairs. The resulting amounts of training examples in the four training subsets as well as their attribute density, can be seen in Table 5.9.

Table 5.9: Training subsets profiling.

T=title, D=description, B=brand, S=specification table

Category	Size	# Pairs		% Density			
		Matching	Non-matching	T	D	B	S
computers	small	722	2,112	100	51	34	21
	medium	1,762	6,332	100	51	34	20
	large	6,146	27,213	100	51	31	18
	xlarge	9,690	58,771	100	50	30	16
cameras	small	486	1,400	100	53	21	4
	medium	1,108	4,147	100	57	22	4
	large	3,843	16,193	100	60	25	3
	xlarge	7,178	35,099	100	66	29	3
watches	small	580	1,675	100	43	15	7
	medium	1,418	4,995	100	44	14	6
	large	5,163	21,864	100	45	14	6
	xlarge	9,264	52,305	100	50	11	5
shoes	small	530	1,533	100	49	8	0
	medium	1,214	4,591	100	49	7	0
	large	3,482	19,507	100	51	6	0
	xlarge	4,141	38,288	100	53	5	0

Baseline Experiments

We execute a set of baseline experiments using different supervised symbolic and subsymbolic entity resolution methods. For all experiments, the training subsets are used for training while the trained model is evaluated against the manually verified WDC Product Gold Standard. Additionally, we pre-process the attribute values by lowercasing and removing non-alphanumeric characters and stopwords. We apply a combination of different record pair comparison and classification configurations for the symbolic and subsymbolic methods. For the record pair comparison step of symbolic methods, we use (i) binary word co-occurrence, which assigns 1 if the word appears in both offers of the corresponding pair and 0 otherwise, and (ii) data type specific similarity metrics, as introduced in Section 2.3.3, which are automatically generated using the Magellan framework [Konda et al., 2016]. For the record pair classification of symbolic methods, we train the following machine learning models: logistic regression, linear SVC, random forests, and XGBoost. For the subsymbolic methods, we use pre-trained and self-trained *fastText* embeddings in combination with the network types implemented by the DeepMatcher framework, i.e. RNN and hybrid.

The results of the experiments in terms of F1 score are summarized in Table 5.10. We use different font colors to denote the best performing model per size for each family of methods. Given the small training sets, the random forest and

logistic regression classifiers perform the best for all product categories in combination with both the co-occurrence and similarity-based features. When using the medium, large and xlarge training sets, there is no clear winner in performance for a specific classification model with regard to symbolic entity resolution methods. Concerning the results of the subsymbolic methods, we observe that RNN delivers the best results with the exception of Watches small/medium, Shoes small, and Cameras large, for which the hybrid model delivers the best results.

The subsymbolic entity resolution baseline methods are 7-23% better in F1 compared to the symbolic entity resolution methods. This confirms the findings of Chapter 3 that subsymbolic methods excel on tasks with high textuality and a large number of corner cases. Indeed, as reported in Chapter 3, the WDC product-related entity resolution tasks have a high textuality level with up to 109 words per offer on average, as well as a large number of corner cases estimated to be up to 98% for the xlarge tasks. Finally, we observe that although the F1 scores improve with the increase of the training set size for the subsymbolic methods, with the only exception a small drop of 1.52 percentage points from watches large to xlarge, this is rarely the case for the F1 scores retrieved with the symbolic methods. For these families of methods, we notice large drops in F1 score with the increase of the training set size which are up to 32 percentage points, e.g. in the case of large to xlarge training sets of the watches category when LinearSVC model with similarity-based features is applied. We assume that this effect is due to the increasing dimensionality of the training data, which cannot always be captured by traditional machine learning models relying on symbolic features leading to a decrease in performance [Caruana et al., 2008].

5.4 Related Work

In this section, we first compare the training sets that can be derived from the WDC Product Corpus to existing entity resolution benchmark tasks, and then we refer to related works that have used the WDC Product Corpus as part of their methodological or experimental setup.

Comparison to Benchmark Entity Resolution Tasks. In Chapter 3, we introduced various benchmark tasks of different topical domains and sizes. In this section, we will focus and compare in detail the WDC Product Corpus to (i) benchmark product matching tasks and (ii) the largest publicly available matching tasks.

Many of the existing benchmark tasks contain records describing products, similarly to the WDC Product Corpus. *Abt-Buy* and *Amazon-Google* are two widely-used product tasks [Köpcke et al., 2010]. Additionally, Gokhale et al. [2014] provide publicly the *Walmart-Amazon* benchmark task. All three product-related tasks derive from two e-shops and contain up to 1,300 manually labeled record pairs. Petrovski and Bizer [2017] provide a gold standard for product data extraction and matching with product offers from 32 different e-shops.

Table 5.10: F1 scores of baseline experiments using the training sets extracted from the WDC English Product Corpus.

The different font colors denote the best-performing model per size for each family of methods.

Category	**Symbolic ER - Word co-occurrence features**															
	Logistic regression				Linear SVC				XGBoost				Random forest			
	small	medium	large	xlarge	small	medium	large	xlarge	small	medium	large	xlarge	small	medium	large	xlarge
computers	**61.20**	73.00	80.90	82.09	60.99	**74.55**	**81.51**	**83.39**	58.08	58.55	58.69	55.85	52.24	48.20	49.71	44.11
cameras	**61.75**	**70.26**	**82.08**	72.83	47.45	69.67	75.99	**73.03**	48.94	55.06	56.38	54.95	39.03	49.20	48.26	36.32
watches	**62.97**	**69.54**	**78.14**	**77.01**	62.88	68.96	77.86	76.30	53.31	55.65	56.06	53.28	42.04	45.49	32.09	16.67
shoes	**71.90**	**79.19**	71.32	**71.76**	70.69	78.80	**71.70**	70.73	63.75	65.01	66.99	67.69	58.29	60.96	56.00	52.55
	Symbolic ER - Similarity-based features															
	Logistic regression				Linear SVC				XGBoost				Random forest			
	small	medium	large	xlarge	small	medium	large	xlarge	small	medium	large	xlarge	small	medium	large	xlarge
computers	54.12	55.06	54.96	54.79	54.96	54.60	54.81	54.79	57.78	62.86	**69.22**	**69.23**	**59.23**	**64.13**	65.91	64.72
cameras	**55.63**	55.54	54.93	56.70	55.02	56.82	55.03	56.12	51.45	56.21	59.89	**64.21**	54.28	**59.44**	**60.76**	62.63
watches	55.63	57.11	41.44	30.58	55.41	57.14	57.04	25.06	58.33	**65.85**	66.67	**60.76**	**67.16**	64.36	**68.27**	58.65
shoes	51.42	51.21	50.20	49.59	50.51	51.15	49.74	49.37	57.39	60.82	54.97	47.00	**64.07**	**62.96**	**63.45**	**63.27**
	Subsymbolic ER - fastText embeddings															
	RNN + self-trained fastText				RNN + pre-trained fastText				Hybrid + self-trained fastText				Hybrid + pre-trained fastText			
	small	medium	large	xlarge	small	medium	large	xlarge	small	medium	large	xlarge	small	medium	large	xlarge
computers	68.18	76.46	89.56	92.87	**70.55**	**77.82**	**89.55**	**93.42**	65.13	76.17	86.31	88.31	66.88	77.48	87.98	92.24
cameras	67.38	75.09	86.73	88.19	**68.59**	**76.53**	86.57	**89.25**	64.12	75.69	81.53	85.44	63.11	73.69	**87.19**	85.67
watches	65.27	78.73	**92.78**	94.13	64.75	78.53	91.28	**94.40**	**66.70**	**83.56**	91.06	93.44	66.32	82.21	91.57	91.92
shoes	73.08	77.32	87.59	90.54	72.91	**79.48**	**90.39**	**92.70**	71.23	75.64	85.57	87.65	**73.86**	78.02	88.65	91.23

Mudgal et al. [2018] use several large product tasks for evaluating deep learning-based entity resolution methods. However, these are not publicly available. Crescenzi et al. [2021] curate ALASKA, a corpus consisting of offers describing 70 thousand distinct real-world products from 71 e-shops. ALASKA supports multiple tasks along the data integration pipeline, such as schema matching and entity resolution. The product offers are extracted using focused crawling and a product specification extraction tool, which inspects the HTML page and extracts key/value pairs from specification tables. The ALASKA corpus has been used for training, evaluating, and comparing entity resolution methods submitted to the 2020 SIGMOD Programming Contest[9] as well as to the DI2KG challenges in 2019 and 2020.[10] In contrast to the WDC Product Corpus, which allows for automatic generation of training sets using pairs of offers of the same or different clusters, the training sets derived from the ALASKA corpus are manually curated and labeled. For the SIGMOD and DI2KG entity resolution challenges different training sets were provided to the participants, with the largest one containing 44 thousand matching pairs.

The largest publicly available benchmark tasks for entity resolution in terms of the number of matching pairs include records from the citations and music domain. The Citeseer-DBLP contains more than 550 thousand matching pairs of citation records deriving from two data sources. The Falcon-Song corpus, originating from the Million Songs Dataset provided by McFee et al. [2012] contains more than 1.2 million matching pairs of song records. Both of the tasks are publicly available in the Magellan data repository.[11]

Table 5.11 gives a comparison overview of the discussed benchmark tasks and corpora for entity resolution along the following dimensions: topical domain, public availability, number of data sources from which the records originate, and the number of matching pairs that can be used for training. The last two lines in the overview table contain the information along the mentioned dimensions for the training sets that can be derived from the WDC Product Corpus and WDC English Product Corpus, respectively. The amount of matching pairs refers to the maximum amount of pairs of offers that can be retrieved by combining pairwise the offers of the same clusters in each corpus. Concerning the number of matching pairs of the WDC Product Corpus, we see that it is several orders of magnitude larger than the existing evaluation tasks in the area of product matching, including public as well as proprietary tasks. Compared to the Falcon-Songs task, the WDC English Product Corpus is 17 times larger. Concerning the number of sources, the WDC English Product Corpus covers 43,293 sources, while the existing tasks contain records from at most 71 sources. This indicates that even the WDC English Product Corpus alone can be used to derive the largest, in terms of the number of matching record pairs, and most heterogeneous, in terms of the number of data sources, training sets for entity resolution.

[9] http://www.inf.uniroma3.it/db/sigmod2020contest

[10] http://di2kg.inf.uniroma3.it

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

Table 5.11: Comparison overview of benchmark entity resolution tasks.

Dataset	Domain	Publicly available	# Data sources	# Matching pairs
Walmart-Amazon [Gokhale et al., 2014]	electronics	yes	2	1,154
Amazon-Google [Köpcke et al., 2010]	software	yes	2	1,300
Abt-Buy [Köpcke et al., 2010]	product	yes	2	1,097
DM-Clothing [Mudgal et al., 2018]	clothing	no	1	105,608
DM-Electronics [Mudgal et al., 2018]	electronics	no	1	98,401
DM-Home [Mudgal et al., 2018]	home	no	1	111,714
DM-Tools [Mudgal et al., 2018]	tools	no	1	96,836
DM-Company [Mudgal et al., 2018]	company	yes	not reported	28,200
Citeceer - DBLP (Magellan repository)	citations	yes	2	558,787
Falcon - Songs (Magellan repository)	music	yes	1	1,292,023
WDC - Electronic Products GS [Petrovski and Bizer, 2017]	electronics	yes	32	1,500
Alaska [Crescenzi et al., 2021]	product	yes	71	44,039
WDC Product Corpus	product	yes	79,126	40,582,671
WDC English Product Corpus	product	yes	43,293	20,773,304

Related Works using the WDC Product Corpus Since its publication in 2019, the WDC Product Corpus has been used by several related works on entity resolution but also for other tasks, such as product categorization. Below, we summarize these works, present their results and, if applicable, compare them to our baseline results in Table 5.12.

The WDC Product Corpus was provided to the participants of the MWPD2020 challenge on product entity resolution [Zhang et al., 2020b]. Three of the six participating systems used the corpus in addition to the provided training set to extract additional training data for fine-tuning transformer-based language models.

Li et al. [2020] use the category-specific training sets (Computers, Cameras, Watches, Shoes) from the WDC English Product Corpus for fine-tuning DITTO, a deep-learning entity resolution system based on pre-trained transformer-based language models. Additionally, the WDC Product Gold Standard is used for the evaluation of DITTO's performance and its comparison to DeepMatcher [Mudgal et al., 2018]. The results of Li et al. [2020] indicate that DITTO outperforms DeepMatcher by up to 4.0 F1 percentage points for all training sets of size large or xlarge, with the Shoes category being the only exception for which it underperforms by 2%. The improvement is more significant for the medium and small training sets, as in this case it outperforms by up to 18% in all categories.

Peeters et al. [2020a] use the computer-specific training sets derived from the WDC English Product Corpus for fine-tuning a BERT-based language model on the product entity resolution task. The fine-tuned results significantly outperform DeepMatcher (up to 20%) while achieving comparable performance to the results of DITTO, with the differences in F1 ranging between -0.98 to +1.59. Apart from the fine-tuning step, the authors derive 3.5 million additional training offer pairs from the WDC English Product Corpus, which they use for intermediate in-domain training. The intermediate training occurs before the fine-tuning step, considering the binary product matching objective as well as the masked language modeling objective. Compared to the fine-tuned model, the intermediate training step boosts

the overall performance by up to 11.84 F1 percentage points when using only the binary product matching objective and by up to 14.64 F1 percentage points when using the additional masked language modeling objective.

Peeters and Bizer [2021] use offer pairs as well as distinct offer records with their identifiers from the computer-specific training sets for fine-tuning a BERT model using a dual objective: the binary matching objective similarly to the previous work and a multi-class classification objective, i.e. the model is tasked to predict the identifier of each offer. The developed model is called JointBERT and outperforms DITTO for the computer-related tasks given enough training data, i.e. when the large and xlarge sets are used for training, by up to 5% in F1. In their later work, Peeters and Bizer [2022] apply supervised contrastive learning to pre-train a transformer encoder which is then fine-tuned on the entity resolution task at hand. They use the computer-specific training sets derived from the WDC English Product Corpus for evaluation and compare their results to DeepMatcher [Mudgal et al., 2018], RoBERTa [Liu et al., 2019], DITTO [Li et al., 2020] and their previous work JointBERT [Peeters and Bizer, 2021]. The experimental results show that their supervised contrastive learning model performs by 0.8 to 8.84 F1 percentage points better than the best baseline method. The performance improvement varies depending on the size of the WDC training sets, i.e. larger improvements are observed for small and medium training sizes.

Wilke and Rahm [2021] use the Shoes training set to train a multi-modal entity resolution method that builds on DeepMatcher and uses both textual and visual features. The authors add a corresponding image crawled from the Web to each offer in the Shoes training set. The results show that combining images with the textual attributes of the offers can improve the model performance in comparison to only exploiting the textual attributes by up to 2%.

Tu et al. [2022] use the WDC Product Gold Standard datasets of the Shoes, Watches, Computers, and Cameras categories to evaluate the effect of domain adaptation. Domain adaptation refers to a set of techniques for effectively reusing labeled data from a source task to a target task for which little or no training data are available. The WDC tasks of the different categories are combined pairwise, e.g. Shoes as a source task and Watches as a target task. The results show that domain adaptation can improve over naive transfer learning by up to 8.3 F1 percentage points, while for some combinations of tasks, the difference is either minimal (< 0.7) or negative (-1.5 for the computers-watches combination). This finding is attributed to the "proximity" of the source and target tasks.

Finally, Brinkmann and Bizer [2021] use offers from the WDC Product Corpus for pre-training a ROBERTa model for the task of hierarchical product categorization. The authors extract a small and a large set of offers from the WDC Product Corpus, with 75 thousand and 1.1 million offers, respectively. The attributes title, description, and category are considered for the pre-training phase. The results indicate that the pre-training step using the offers from the WDC Product Corpus can boost the performance of the ROBERTa model by up to 1.22% in terms of weighted F1 score.

Table 5.12: Comparison overview of F1 scores of baseline experiments and related work using the training sets extracted from the WDC English Product Corpus.

		Baselines presented in Section 5.3.2			Results from related work				
Category	Size	Symbolic Word co-occurrence	Symbolic Similarity-based	Subsymbolic fastText	DITTO [Li et al., 2020]	Intermediate train. [Peeters et al., 2020a]	JointBERT [Peeters and Bizer, 2021]	R-SupCon [Peeters and Bizer, 2022]	Multi-modal DL [Wilke and Rahm, 2021]
computers	small	61.20	59.23	63.27	80.76	96.53	77.55	**95.21**	
cameras		61.75	55.63	68.59	**80.89**				
watches		62.97	67.16	66.70	**85.12**				
shoes		71.90	64.07	73.86	**75.89**				
computers	medium	74.55	64.13	77.82	88.62	96.58	88.82	**98.50**	
cameras		70.26	59.44	76.53	**88.09**				
watches		69.54	65.85	83.56	**91.12**				
shoes		79.19	62.96	79.48	**82.66**				
computers	large	81.51	69.22	89.55	91.70	95.82	96.90	**98.50**	
cameras		82.08	60.76	87.19	**91.23**				
watches		78.14	68.27	92.78	**95.69**				
shoes		71.70	63.45	**90.39**	88.07				
computers	xlarge	83.39	69.23	93.42	95.45	97.37	97.49	**98.33**	
cameras		73.03	64.21	89.25	**93.78**				
watches		77.01	60.76	94.40	**96.53**				
shoes		71.76	63.27	**92.70**	90.11				83.60

5.5 Discussion and Conclusion

In this chapter, we presented the WDC Product Corpus for entity resolution, a corpus of 26 million product offers, grouped into 16 million clusters representing different real-world products and originating from 79 thousand websites with schema.org/Product annotations. For the curation of the corpus, we exploited semantically annotated product identifiers, investigated different errors related to schema.org/Product annotations, and developed a cleansing pipeline to reduce the impact of those errors.

We profiled the WDC Product Corpus along different dimensions, including the distribution of cluster sizes, the density of product-relevant schema.org properties, and the product categories to which the offers of the corpus belong. The profiling analysis showed that by combining pairwise all intra-cluster and inter-cluster offers, we could generate up to 40 million and more than 5 trillion matching and non-matching pairs of offers describing products of different categories. Only considering clusters of the English subset of the WDC Product Corpus with a size between 5 and 80, 20.7 million matching offer pairs and 2.6 trillion non-matching offer pairs can be derived. The derived matching and non-matching pairs can be used for training entity resolution models targeting the product entity resolution task. The training sets generated from the WDC Product Corpus are the largest and most heterogeneous, in terms of the number of data sources from which the records originate, among the training sets of all publicly available entity resolution benchmark tasks. Without the website owners putting semantic annotations into their HTML pages, it would have been much harder, if not impossible, to extract product offers from thousands of e-shops referring to the same products.

Additionally, we evaluated the cleanliness and training quality of the WDC Product Corpus. Based on a manually verified sample of pairs of offers, we evaluated the level of noise, i.e. intra-cluster offer pairs that refer to different real-world products, to be 6%. Using product category-specific matching and non-matching pairs derived from the WDC Product Corpus for training baseline entity resolution models, we showed that F1 scores up to 94.4% can be achieved when applying the deep learning-based model DeepMatcher [Mudgal et al., 2018]. The evaluation was based on a manually verified test set of product offers having no product identifier information deriving from the same corpus. Finally, we presented multiple subsymbolic entity resolution models from the related work, such as DITTO [Li et al., 2020] and JointBERT [Peeters and Bizer, 2021] using training subsets derived from the WDC Product Corpus for model training and fine-tuning. Those models are able to achieve significant improvements in terms of F1 score in comparison to the baseline DeepMatcher results. The high matching performance that can be achieved by training entity resolution models with offer pairs from the WDC Product Corpus clearly proves the utility of the Semantic Web as a rich source of distant supervision for product-related entity resolution tasks.

Part III

Active Learning for Entity Resolution

Chapter 6

Foundations of Active Learning

Active learning is a supervised learning paradigm. It is based on the fundamental idea that a machine learning model can achieve better predictive performance if trained on a small but carefully chosen set of labeled instances [Settles, 2012]. "Traditional" supervised learning methods are executed sequentially with respect to the labeling process, i.e. a set of instances is selected upfront, labeled, and used for training a machine learning model. We will refer to these methods as *passive* in order to distinguish them from the active learning ones [Settles, 2012]. On the contrary, active learning methods are iterative methods. In each iteration, a query selection component guides the labeling process to select only a subset of all available unlabeled instances for labeling [Cohn et al., 1994]. The selected instances are typically labeled by a human annotator [Settles, 2012].

Active learning is a good fit for supervised learning tasks for which abundant unlabeled instances are available but obtaining their labels for training is difficult, costly, or time-consuming. As such, it has been applied for different applications, like image classification [Beluch et al., 2018], named entity recognition [Shen et al., 2004], information extraction [Zhuang et al., 2020], as well as entity resolution [Chen et al., 2019; Isele and Bizer, 2013; Sarawagi and Bhamidipaty, 2002]. An overview of different active learning-based methods for different types of applications is provided by Settles [2012]. A more focused survey on active learning methods for natural language processing tasks is provided by Olsson [2009]. Ren et al. [2021] provide an overview of active learning methods that use deep-learning-based models. Meduri et al. [2020] give an overview of active learning methods for entity resolution.

In this part of the thesis, we turn our focus on active learning for entity resolution as a means of reducing the labeling effort while achieving comparable performance to passive entity resolution methods. In this chapter, we discuss the main concepts around active learning, concentrating on the most prominent techniques employed for entity resolution.

This chapter is structured into four sections. Section 6.1 discusses the main scenarios in which the query selection component can ask queries to guide the

labeling process. Section 6.2 focuses on one of the main scenarios, i.e. the pool-based active learning, describes its workflow for the task of entity resolution, and discusses relevant workflow desiderata. In Section 6.3, we discuss different families of query selection strategies. Finally, in Section 6.4, we present commonly used metrics for evaluating the performance of active learning methods for entity resolution.

6.1 Query Selection Scenarios

There exist several ways in which the query selection component can ask queries to guide the labeling process during active learning, among which the following three have been distinguished by Settles [2012]: (i) membership query-synthesis, (ii) stream-based selective sampling, and (iii) pool-based sampling. In the following, we describe the three query selection scenarios.

Membership Query-Synthesis In the membership query-synthesis scenario, the query selection component synthesizes new queries instead of selecting existing unlabeled instances [Angluin, 1988]. The new queries are interpolated, i.e. they are described with the same features and lie within the feature value ranges of the input space defined by the existing unlabeled data [Baum and Lang, 1992]. A bottleneck that can occur in this query selection scenario is that often the generated queries are hard to be interpreted by the human annotator [Baum and Lang, 1992; Settles, 2012]. This bottleneck is tackled by the following scenarios.

Stream-based Selective Sampling In the stream-based selective sampling scenario, the query selection component receives a stream of unlabeled instances and decides for each one of them if they should be labeled or not [Cohn et al., 1994]. This query selection scenario has been employed in different applications, mostly focused on natural language processing tasks, such as word sense disambiguation [Fujii et al., 1998] and part-of-speech tagging [Dagan and Engelson, 1995].

Pool-based Sampling A pool-based scenario assumes that all unlabeled instances are provided at once [Settles, 2012]. In this scenario, the query selection component has always access to a pool of all unlabeled instances, in contrast to stream-based selective sampling, where the unlabeled instances are provided in small batches as a stream. In the context of this thesis, we consider entity resolution as a static task, i.e. the aim is to identify all matching record pairs among two or more static data sources. In such a setting, all unlabeled record pairs can be generated upfront, and therefore the pool-based scenario is the most fitting one. Pool-based active learning for entity resolution has been widely applied in related work, e.g. in [Isele and Bizer, 2013; Ngomo et al., 2011; Ngomo and Lyko, 2012; Sarawagi and Bhamidipaty, 2002]. An overview of pool-based active learning methods for entity resolution is given in [Meduri et al., 2020; Papadakis et al., 2021]. In the sections and chapters to follow, we will solely focus on pool-based active learning and not discuss the other two query selection scenarios any further.

6.2 Pool-based Active Learning

As already discussed in the previous section, the pool-based query selection scenario has been widely employed by active learning methods for entity resolution. In this section, we discuss the main components and general workflow of pool-based active learning methods for entity resolution.

6.2.1 Components

There exist four main components within a pool-based active learning workflow: the unlabeled pool, the labeled set, the query selection, and the learner [Settles, 2012].

Unlabeled Pool The unlabeled pool contains all instances for which the label is yet unknown. In the context of entity resolution, the unlabeled pool contains pairs of records without any matching or non-matching label. Typically, the record pairs in the unlabeled pool are the output of a blocking step, as discussed in Section 2.3.2.

Labeled Set The labeled set contains the set of instances that have been labeled by the human annotator. In the context of entity resolution, the labeled set contains record pairs with matching or non-matching labels. Initially, the labeled set is either empty, e.g. in the work of Isele and Bizer [2013], or is initialized with a small set of seeding record pairs, e.g. in the work of Qian et al. [2017]. The initialization of the labeled set is a non-trivial problem, which will be further discussed in Chapter 7.

Query Selection The query selection component is responsible for selecting instances from the unlabeled pool to be labeled by the human annotator. For an entity resolution task, the query selection component employs a certain query selection strategy, which assesses all record pairs of the unlabeled pool according to a utility measure [Settles, 2012], often referred to as *informativeness* [Meduri et al., 2020; Papadakis et al., 2021]. The record pairs that are assessed as most informative are selected for labeling. Commonly used query selection strategies will be covered in Section 6.3.

Learner The learner component is the machine learning model representing the solution of the task. For an entity resolution task, as defined in the context of this thesis, the learner is a binary classifier relying on any type of machine learning algorithms, such as random forests, SVM, or neural networks. An overview of different types of learners employed by active learning methods for entity resolution is provided in [Meduri et al., 2020].

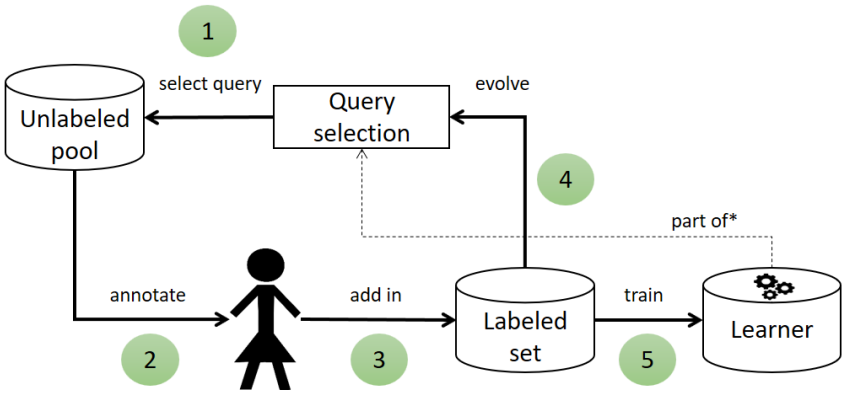


Figure 6.1: Pool-based active learning workflow.

6.2.2 Workflow

After having introduced the four main components of pool-based active learning, we now describe the complete workflow in the context of entity resolution. The main pre-requisite for starting the active learning workflow is the initialization of the labeled set with seeding matching or non-matching pairs. As already introduced, this is a non-trivial task, which will be further analyzed in Chapter 7. Alternatively, a few methods do not use seeding record pairs but randomly initialize the query selection component in order to be able to select the first instance for labeling [Isele and Bizer, 2013; Ngomo and Lyko, 2012]. Figure 6.1 presents the general workflow of pool-based active learning after initializing the labeled set with at least one matching and one non-matching record pair. It should be noted that for some specific query selection strategies (learner-aware query strategies), which will be covered in Section 6.3, the learner component is part of the query selection component. We denote this with the dotted line connecting the two components. In the following, we describe the general workflow.

In each active learning iteration, the query selection component assesses the informativeness of all record pairs of the unlabeled pool. The record pair(s) that are assessed as the most informative are selected and removed from the unlabeled pool (Step 1). The selected record pair(s) are forwarded to a human annotator, who assigns a label *matching* or *non-matching* (Step 2). The labeled record pair(s) are added to the labeled set (Step 3). The labeled set is used to evolve the query selection component according to the employed query strategy (Step 4) and to train the learner (Step 5). The quality of the learner is evaluated on an external test set. In order to assess the quality of the learner in each iteration, it is common to train the learner and evaluate its predictions on the test set at the end of each active learning iteration [Meduri et al., 2020].

6.2.3 Workflow Desiderata

The general workflow described in the previous section is followed by most pool-based active learning methods for entity resolution [Kasai et al., 2019; Meduri et al., 2020; Qian et al., 2017], as well as for other applications, such as text and image classification [Hoi et al., 2006; Zhang and Chen, 2002]. However, there exist certain workflow desiderata which can vary among the methods. In the following, we list different workflow desiderata and discuss how they are realized in related works as well as in the context of this work. As the related works that will be discussed tackle different tasks and do not solely focus on entity resolution, we will use the more general term *instance* instead of *record pair*.

Amount of Annotators A pool-based active learning workflow can include one or more human annotators in the labeling process. In the case that more annotators are involved, widely known as *crowdsourcing*, each instance selected for labeling by the query selection component receives more than one label. Techniques combining active learning with crowdsourcing have been widely explored [Agarwal et al., 2013; Ipeirotis et al., 2014]. Crowdsourcing active learning techniques depend their query selection strategy on the redundancy of labels [Bouguelia et al., 2018; Sherif et al., 2020]. In the context of this thesis, we consider a single human annotator while each instance is assigned a single label, similarly to many related works, e.g. [Isele and Bizer, 2013; Kasai et al., 2019].

Correctness of Labels The assigned labels by the human annotator can be either assumed as always correct or error-prone. The existence of noisy labels is a general problem in machine learning which affects the prediction quality of the classification model and has been studied in both passive [Natarajan et al., 2013; Song et al., 2022] and active learning settings [Bouguelia et al., 2018]. In the latter, the quality of the classification model, i.e. the learner, can be further deteriorated due to the query selection component that can be misled by the presence of noise in the labeled set [Bouguelia et al., 2018].

Commonly, the case of a flawless human annotator is assumed in related active learning works [Chen et al., 2019; Sarawagi and Bhamidipaty, 2002]. However, there exists a line of work on active learning which considers that labeling errors can occur [Bouguelia et al., 2018; Sherif et al., 2020; Zhang et al., 2015]. To circumvent the problem of error-prone labels, some active learning methods use crowdsourcing techniques and exploit the disagreement of the labels provided for the same instance [Agarwal et al., 2013; Ipeirotis et al., 2014; Sherif et al., 2020], or rely on re-labeling mechanisms [Bouguelia et al., 2015, 2018; Zhang et al., 2015]. In the latter case, the aim is to spot and correct the errors using the learner's confidence [Bouguelia et al., 2015], estimating the influence of erroneously labeled instances [Zhang et al., 2015], or by comparing the learner's predictions with and without the labels which are suspected to be wrong [Bouguelia et al., 2018]. In our work, we assume a flawless human annotator who always provides correct labels.

Amount of Queries per Iteration In each active learning iteration, one, i.e. single queries, or more, i.e. batch queries, instances can be selected for labeling. The amount of queries per iteration varies significantly in related work. For example, single queries are considered in [Bouguelia et al., 2018; Isele and Bizer, 2013]. Batch queries of size ten are applied in [Meduri et al., 2020; Ngomo and Lyko, 2012]. Small batch sizes are inefficient for deep learning-based active learning methods due to the long training times [Ren et al., 2021]. Therefore, for such methods, larger batch sizes are preferred, which typically are in the range of 20 [Kasai et al., 2019] to 100 [Nafa et al., 2020]. In the context of this thesis, we consider single queries in each active learning iteration.

Stopping Criterion The active learning workflow iterates until a stopping criterion is met [Settles, 2012]. Typically, the stopping criterion is a certain labeling budget, e.g. 100 labels. In our work, we use a budget-based stopping criterion similar to many pool-based active learning methods [Chen et al., 2019; Isele and Bizer, 2013; Meduri et al., 2020]. For example, Isele and Bizer [2013] consider a budget of 50 labels and Bouguelia et al. [2018] consider a budget of 250 labels. Related works that consider a varying batch query size set their budget limitation with respect to the number of active learning iterations. For example, Chen et al. [2019] consider a varying batch query size from 10 to 30 in their experimental setup and set 199 active learning iterations as their stopping criterion. Alternative stopping criteria for active learning have been proposed in [Olsson and Tomanek, 2009; Vlachos, 2008]. The common ground of these methods is that they aim to quantify the confidence of the learner regarding the most informative unlabeled instances and stop iterating once the confidence has converged to a stable level over multiple iterations.

Table 6.1 (left) gives a comparative overview of different active learning methods for entity resolution with respect to the realization of the discussed pool-based active learning workflow desiderata. The stopping criterion dimension is not reported on the table, as all active learning methods that focus on entity resolution apply the budget-based stopping criterion.

6.3 Query Selection Strategies

As already discussed, the active learning query selection component is responsible for assessing the informativeness of the record pairs in the unlabeled pool. To do so, it employs a query selection strategy, also referred to more compactly as *query strategy* [Settles, 2012]. An overview of different query strategies is provided in [Settles, 2012]. A distinction between different families of query strategies applied on entity resolution tasks is presented in [Meduri et al., 2020; Papadakis et al., 2021].

Table 6.1: Comparison of active learning methods for entity resolution with respect to the pool-based active learning workflow desiderata and the query strategy categorization.

Method	Workflow desiderata			Query strategy categorization				
	# Annotators	Correctness of labels	# Queries per iteration	Based on classification	Based on heuristics	Based on monotonicity	Learner-aware	Learner-agnostic
[Tejada et al., 2001]	1	correct	single	✓ (committee-based)				✓
[Sarawagi and Bhamidipaty, 2002]	1	correct	single	✓ (committee-based)				✓
[de Freitas et al., 2010]	1	correct	single & batch	✓ (committee-based)			✓	
[Arasu et al., 2010]	1	correct	single			✓	✓	
[Ngomo and Lyko, 2012]	1	correct	batch (10)	✓ (committee-based)			✓	
[Isele and Bizer, 2013]	1	correct	single	✓ (committee-based)			✓	
[Ngomo et al., 2013]	1	correct	batch (10)	✓ (committee-based)	✓		✓	
[Christen et al., 2015]	1	error-prone	batch (calculated)		✓			✓
[Qian et al., 2017]	1	correct	batch (12)		✓		✓	
[Kasai et al., 2019]	1	correct	batch (20)	✓ (margin-based)			✓	
[Chen et al., 2019]	1	correct	batch (10-30)	✓ (committee-based)			✓	
[Sherif et al., 2020]	2-16	error-prone	batch (10)	✓ (committee-based)			✓	
[Nafa et al., 2020]	1	correct	batch (up to 100)		✓		✓	

Meduri et al. [2020] distinguish two categories of query strategies, *learner-aware* and *learner-agnostic*, given the access of the query selection component to the learner. Learner-aware query strategies use the learner to assess the informativeness of the instances of the unlabeled pool. In contrast, learner-agnostic query strategies do not consider the learner for assessing the informativeness of the instances. A different categorization is provided by Papadakis et al. [2021] concerning the different types of assumptions taken for assessing how informative a record pair is. In the following, we present the three categories of query strategies as distinguished by Papadakis et al. [2021] and discuss related works belonging to these categories. In addition, we distinguish the related works into the categorization scheme of Meduri et al. [2020], i.e. learner-aware and learner-agnostic. Table 6.1 (right) gives an overall comparison overview of different active learning methods for entity resolution with respect to the categorization of their query strategy.

6.3.1 Heuristic-based

Heuristic-based query strategies rely on the feature vectors of the record pairs in the unlabeled pool for selecting ambiguous to the learner instances [Papadakis et al., 2021]. Ngomo et al. [2013] extend a classification-based query strategy by adding a heuristic for filtering the unlabeled record pairs, which are likely falsely predicted by the learner. Christen et al. [2015] propose AdInTDS, a learner-aware query strategy that relies on the clustering of the unlabeled record pairs and estimates the purity of the clusters. The representative record pairs of the estimated unpure clusters are considered most informative and therefore selected for labeling. Qian et al. [2017] develop a learner-aware active learning method for entity resolution which applies rule learning. The labeled set is used for training the rule learning model, which is then applied to the record pairs of the unlabeled pool. The query strategy uses a set of heuristics and selects record pairs that are likely falsely predicted by the learner, given their feature vector. Similarly, Nafa et al. [2020] aim to select likely falsely predicted record pairs by exploiting a risk model, which assesses the mislabeling risk of the unlabeled record pairs.

6.3.2 Classification-based

Classification-based query strategies exploit the confidence of one or more machine learning models, trained on the current labeled set for assessing the informativeness of the unlabeled record pairs [Papadakis et al., 2021]. In the case that these models include the learner, the query strategies are considered learner-aware, otherwise learner-agnostic [Meduri et al., 2020]. The classification-based query strategies can be further distinguished into *committee-based* and *margin-based*. Committee-based query strategies use multiple models, i.e. the committee. The learner can be part of the committee. The models in the committee are trained on the current labeled set and vote their predictions on the unlabeled record pairs. The informativeness of each record pair is estimated as the disagreement of the votes

of the committee. Margin-based query strategies assess as most informative the record pairs which are close to the decision boundary determined by one or more classifiers.

Classification-based query strategies have been widely adopted by active learning methods for entity resolution. Early approaches focus on committee-based query strategies that comprise either models trained on different subsets of the labeled set [Tejada et al., 2001] or are different parametrizations of the same learner [Sarawagi and Bhamidipaty, 2002]. Multiple works rely on committees of linkage rules which evolve using genetic programming [de Freitas et al., 2010; Isele and Bizer, 2013; Ngomo and Lyko, 2012; Ngomo et al., 2013]. Linkage rules typically comprise a set of property, transformation, comparison, and aggregation operators [Isele and Bizer, 2013]. Chen et al. [2019] propose HeALER, a committee-based query strategy whose committee members are different types of classifiers. Finally, Mozafari et al. [2014] as well as Kasai et al. [2019] use margin-based query strategies, which estimate the informativeness of each unlabeled record pair based on the classification probability of the learner.

6.3.3 Monotonicity-based

Monotonicity-based active learning methods rely on the monotonicity assumption of precision, i.e. given a matching record pair with an overall similarity s, any record pair with a similarity score $> s$ can be safely considered as matching [Tao, 2018]. An example of a monotonicity-based query strategy is applied in the work of Arasu et al. [2010]. The authors rely on the monotonicity assumption of precision and select batches of unlabeled record pairs for annotation that are estimated to increase the precision of the learner.

6.4 Evaluation Metrics

In their survey paper, Meduri et al. [2020] propose four dimensions for evaluating the overall quality of active learning methods for entity resolution: learner quality, labeling effort, latency, and interpretability.

Learner Quality To measure the learner quality, the common evaluation metrics for entity resolution, as presented in Section 2.4, are used: precision, recall, and F1 score. These metrics are calculated with respect to the predictions of the learner of the last active learning iteration on an external test set. This allows reporting the quality of the learner given a pre-defined labeling budget. In order to measure the quality of the learner during active learning, in both earlier and later iterations, the F1 score is typically measured and reported after each active learning iteration.

To compare different active learning methods across all iterations, an aggregated F1 score is needed. Bouguelia et al. [2015] report the average score of all iterations. Alternatively, the area under the F1 score curve is commonly used [Moza-

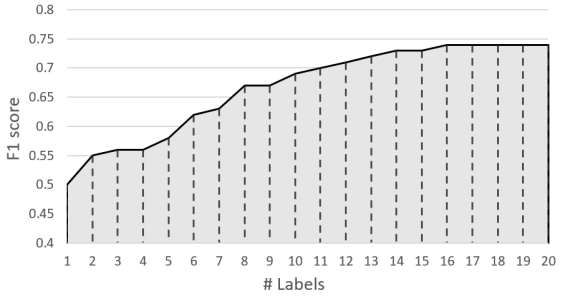


Figure 6.2: Example of approximating the F1-AUC using the trapezoidal rule.

fari et al., 2014; Sherif et al., 2020]. The area under the F1 scores curve, abbreviated with F1-AUC, is calculated as the definite integral between two points. One technique of approximating the definite integral is the composite trapezoidal rule. The trapezoidal rule partitions the curve into equal subintervals and approximates the area under the intervals as a trapezoid. In the case of active learning, each subinterval can be considered as the area of the F1 curve between two points with labeled data $[i, i+1]$, as shown in Figure 6.2 and indicated with the dotted vertical lines. More formally, the trapezoidal rule for approximating the definite integral between two points a and b in the F1 curve considering n partitions is given by Equation 6.1.

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}\left[f(x_0) + 2f(x_1) + ... + 2f(x_{n-1}) + f(x_n)\right] \tag{6.1}$$

where:

$\Delta x = \frac{b-a}{n}$, subinterval, i.e. one labeled example for the case of active learning
$x_i \ = \alpha + i\Delta x$

Labeling Effort In the context of our work, we assume that each record pair is equally difficult to be assessed as matching or non-matching by the human annotator. Therefore, the labeling effort is quantified only by the number of record pairs that are manually labeled during active learning and is independent of other dimensions, such as the size or the ambiguity of the descriptions of the record pairs provided to the human annotator for labeling.

Latency The metric of latency refers to the overall time required for query selection. Considering that during active learning, a human annotator needs to be available throughout the complete labeling process, latency is an additional important evaluation metric. In our work, the three metrics mentioned above are used to measure the effectiveness of active learning methods.

Interpretability The metric of interpretability evaluates how readable and interpretable the output model is to the user [Meduri et al., 2020]. However, the dimension of interpretability solely relies on the classification model used as the learner, independently of whether the model appears in a passive or an active learning setting. For example, rule-based models are, in general, more interpretable than neural network-based models. Therefore, in our work, we do not consider the metric of interpretability for evaluating the results of active learning methods for entity resolution.

Chapter 7

Unsupervised Bootstrapping of Active Learning for Entity Resolution

After having discussed the foundations of active learning, starting from this chapter, we turn our focus on pool-based active learning methods as a means of reducing the labeling effort for entity resolution. As already introduced, pool-based active learning methods for entity resolution use a query strategy for assessing how informative the record pairs of the unlabeled pool are. The record pairs assessed as most informative are selected for labeling and added to the labeled set, which is further used for training the learner. Additionally, we saw in Section 6.1, that the majority of active learning methods either use classification-based query strategies or/and are learner-aware. For such methods, the pre-requisite for the pool-based active learning workflow to start is the initialization of the classification model(s) used as part of the query strategy or/and the learner.

A problem that frequently arises is the lack of labeled record pairs before active learning starts, which further hinders the initialization of the classification model(s), known as the *cold start* problem [Konyushkova et al., 2017]. The cold start problem is a non-trivial bottleneck for active learning methods and different techniques have been applied to circumvent it. Such techniques include random initialization of the classification model(s) which are part of the query strategy and learning [Isele and Bizer, 2013; Ngomo and Lyko, 2012], transfer learning [Kasai et al., 2019] or labeling of a seeding subset of record pairs [Chen et al., 2019; Konyushkova et al., 2017; Qian et al., 2017; Sarawagi and Bhamidipaty, 2002]. All of these approaches rely on certain assumptions which can vary among different entity resolution tasks, such as the distribution of pre-calculated similarity scores [Chen et al., 2019], or increase the human labeling effort. The related work will be discussed in further detail in Section 7.1.

In this chapter, we present an unsupervised method for assisting both the initialization as well as the complete workflow of active learning, to which we refer

as *unsupervised bootstrapping*, thus covering the contribution [C4] of the thesis. In contrast to existing active learning initialization approaches, our method comes at no additional labeling cost and is independent of the entity resolution task at hand. Our unsupervised bootstrapping method relies on the unsupervised aggregation of the similarity-based scores of the pool record pairs into one overall score and a thresholding heuristic. The overall scores are compared against the heuristically defined threshold and the record pairs of the unlabeled pool receive matching and non-matching labels which are subject to some degree of noise. The unsupervised labeled record pairs are further exploited for initializing active learning. Additionally, they are also part of the complete active learning workflow, i.e. query selection and training of the learner. The query selection relies on the committee-based query strategy of HeALER [Chen et al., 2019], while we use a random forest classifier as a learner. Finally, we perform a three-fold evaluation on six entity resolution tasks across different profiling groups, as presented in Chapter 3. First, we compare the results of our proposed thresholding heuristic to baseline threshold-based unsupervised entity resolution methods. Second, we compare our unsupervised bootstrapped active learning method to symbolic baselines using the HeALER query strategy [Chen et al., 2019] and random sampling or transfer learning for initialization. Third, we compare our active learning method to subsymbolic baselines, inspired by the work of Kasai et al. [2019]. The subsymbolic baselines use either our unsupervised method or transfer learning for initialization and rely on the DeepMatcher framework [Mudgal et al., 2018].

The contributions of this chapter are summarized as follows:

- We propose a thresholding heuristic for unsupervised entity resolution that uses a domain-independent scoring function.
- We propose a method for bootstrapping active learning that comes at no additional labeling cost and guarantees high anytime performance within a limited annotation budget.
- We perform an extensive evaluation on six entity resolution tasks with different profiling characteristics.
- We compare our proposed method to symbolic and subsymbolic active learning methods.

This chapter is structured into four sections. In Section 7.1, we discuss the related work with respect to the initialization techniques applied by active learning methods for entity resolution and distinguish the latter into symbolic and subsymbolic. Section 7.2 presents our methodology on bootstrapping active learning in an unsupervised fashion. In Section 7.3, we present the experimental results first of our proposed thresholding heuristic and second of the complete active learning workflow and compare them against baseline symbolic and subsymbolic methods. Finally, in Section 7.4, we summarize the main findings of this chapter.

The methodology as well as the evaluation of the results presented in this chapter have been partially published in the Proceedings of the 17th Extended Semantic Web Conference [Primpeli et al., 2020]. The evaluation of the subsymbolic baseline methods is a joint work with Stephan Waitz and the results have been partially published in his Master thesis [Waitz, 2021]. The code developed by Stephan Waitz can be used for reproducing the subsymbolic active learning baseline experiments and is publicly available.[1] Additionally, we provide the code and datasets used for the rest of the experimental evaluation of this chapter.[2]

7.1 Related Work

Pool-based active learning has been widely used for entity resolution in related work. As already identified in Section 6.1 of the previous chapter, the majority of the pool-based active learning methods for entity resolution apply classification-based and/or learner-aware query strategies. Therefore, they require a mechanism for initializing the active learning workflow. In this related work section, we first discuss how these methods initialize the active learning workflow. Next, we distinguish the methods into symbolic and subsymbolic, report their current findings, and discuss some practical considerations which are important to the comparison of active learning methods on clear grounds. A comparison overview of pool-based active learning methods with respect to the criteria discussed in this section is presented in Table 7.1.

Initializing Active Learning A common approach for initializing active learning is manually labeling a randomly selected subset of record pairs from the unlabeled pool. We report the amount of sampled and manually labeled record pairs required for initialization in different active learning works, if explicitly documented in the respective publications, in Table 7.1 and column *Initialization method (seed size)*. Meduri et al. [2020] select 30 record pairs as seeds and add them to the labeled set. Qian et al. [2017] initialize the labeled set with 5 matching and 5 non-matching pairs. Similarly, Brunner and Stockinger [2019] and Sherif et al. [2020] use 10 randomly sampled and manually labeled record pairs as seeds. Wang et al. [2021b] and Nafa et al. [2020] use larger randomly sampled seeding sets of 30 to 565 record pairs which account for 20% of all matching and non-matching pairs per task after blocking. According to Wang et al. [2021b], a larger seeding set is necessary as it guarantees an initial learner of high quality which further positively impacts the complete active learning process. Similarly, Nafa et al. [2020] use larger seeding sets for training the deep learning-based learner of the first iteration. The problem with randomly sampling seeding record pairs for initializing active learning is that, considering the high imbalance of matching and non-matching pairs in an entity resolution task, it might be hard to find even one matching pair

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

[2] https://github.com/wbsg-uni-mannheim/UnsupervisedBootAL

within a small amount of randomly sampled record pairs. This increases the overall human labeling effort, as in order to find e.g. 5 matching record pairs [Qian et al., 2017] a larger amount of record pairs needs to be inspected.

Other related works follow a more principled random sampling approach and randomly select seeding record pairs from different areas of the similarity score distribution [Chen et al., 2019; Sarawagi and Bhamidipaty, 2002; Tejada et al., 2001]. However, these methods need to pre-define a number of similarity groups as well as the number of record pairs from each group that needs to be labeled. These two parameters can vary among different tasks and therefore require manual inspection [Chen et al., 2019]. Furthermore, all initialization methods that rely on a manually labeled seeding set of record pairs increase the human labeling effort in contrast to our approach.

Existing works that do not increase the labeling effort for initializing the active learning workflow can be distinguished into three categories. First, works that randomly initialize the models of the query selection and learner components [Isele and Bizer, 2013; Ngomo and Lyko, 2012; Ngomo et al., 2013]. The common ground of these works is that they rely on populations of linkage rules, evolving with the use of genetic programming. In the very first iteration, a random population of linkage rules is generated. Second, works that rely on transfer learning for initialization [Kasai et al., 2019]. However, this assumes that there exist abundant labeled record pairs of a similar topical domain of high relatedness, i.e. a model trained on the labeled record pairs can be transferred to the entity resolution task at hand [Thirumuruganathan et al., 2018]. Although domain adaptation can be applied for aligning the distributions of the labeled and unlabeled record pairs, its effect in terms of performance has been shown to vary significantly given the *closeness* of the source labeled task to the target unlabeled task [Tu et al., 2022].

Third and similar to our work, Bogatu et al. [2021] rely on an unsupervised approach for extracting a small seeding set of potentially matching and non-matching record pairs. More concretely, the authors perform unsupervised representation learning and derive numeric vectorized representations for all records. Next, they apply nearest–neighbor search and compute the distances between the closest and the furthest record representations. The record pairs having the minimum and maximum distances are assigned non-matching and matching labels respectively and are used as seeding pairs. It is worth noting that the work of Bogatu et al. [2021] was published one year after the publication of our proposed unsupervised bootstrapping method [Primpeli et al., 2020].

Symbolic Active Learning Methods Similar to symbolic entity resolution methods, symbolic active learning methods for entity resolution perform record pair comparison using data type-specific similarity metrics, as discussed in Section 2.3.3. The works that rely on linkage rules use a combination of transformation and similarity operators to compare the records [Isele and Bizer, 2013; Ngomo and Lyko, 2012; Ngomo et al., 2013; Qian et al., 2017].

Table 7.1: Comparison of active learning methods with respect to the initialization approach, entity resolution steps, and usage of training data enlargement techniques and a validation set.

Method	Initialization method (seed size)	Symbolic	Subsymbolic	Record pair comparison	Record pair classification	Training data enlargement	Use of external validation set (size)
[Tejada et al., 2001]	random sampling from the sim. score distribution	✓		similarity-based	decision tree		
[Sarawagi and Bhamidipaty, 2002]	random sampling from the sim. score distribution (2)	✓		similarity-based	decision tree		
[de Freitas et al., 2010]	random sample (2)	✓		linkage rule operators	genetic programming		
[Ngomo and Lyko, 2012]	random model initialization	✓		linkage rule operators	genetic programming		
[Isele and Bizer, 2013]	random model initialization	✓		linkage rule operators	genetic programming		
[Ngomo et al., 2013]	random model initialization	✓		linkage rule operators	genetic programming		
[Qian et al., 2017]	random sample (10)	✓		linkage rule operators	rule learning		
[Kasai et al., 2019]	transfer learning		✓	fastText	DeepMatcher RNN	✓	✓(20%)
[Chen et al., 2019]	random sampling from the sim. score distribution (4-10)	✓		similarity-based	decision tree		
[Brunner and Stockinger, 2019]	random sample (10)	✓		similarity-based	SVM, random forests, neural network		
[Sherif et al., 2020]	random sample (10)	✓		similarity-based	WOMBAT		
[Nafa et al., 2020]	random sample (50-575)		✓	fastText	DeepMatcher (hybrid)		✓(5-25%*)
[Bogatu et al., 2021]	unsupervised based on sim. scores		✓	variant auto-encoder representations	siamese neural network		✓
[Wang et al., 2021b]	random sample (30-500)	✓		similarity-based	random forest	✓	✓(20%)

Different classification models have been used as learners in symbolic active learning methods for entity resolution. The early approaches of Tejada et al. [2001] and Sarawagi and Bhamidipaty [2002] use decision tree models. Later, multiple approaches emerged, using the genetic programming algorithm for learning linkage rules [Isele and Bizer, 2013; Ngomo and Lyko, 2012; Ngomo et al., 2013]. Other works relying on linkage rules developed their own rule learning algorithms, e.g. Qian et al. [2017] with LEARNRULE and Sherif et al. [2020] with WOMBAT. Brunner and Stockinger [2020] experiment with different types of learners, including random forests, SVM, and logistic regression, while Wang et al. [2021b] apply random forests.

In their benchmark study, Meduri et al. [2020] evaluate different types of classification models as learners, including SVM, random forests, and rule learning. Additionally, they evaluate three query strategies: committee-based, margin-based, and heuristic-based. The findings of the benchmark study show that random forests with learner-aware committee-based strategies outperform the rest of the tested models and query strategy combinations on nine different entity resolution tasks.

Concerning the composition of the committee members in committee-based strategies, Chen et al. [2019] have shown that a heterogeneous committee, i.e. the models are different types of classifiers, delivers better results in comparison to different parametrizations of the same classifier. Given the findings of Meduri et al. [2020] as well as of Chen et al. [2019], we use a heterogeneous committee-based query strategy and a random forest as learner for building our symbolic active learning pipeline for entity resolution.

Subsymbolic Active Learning Methods Subsymbolic active learning methods for entity resolution rely on distributed representations for the record comparison step and use deep learning-based architectures for the query selection and learner components. In the work of Kasai et al. [2019] and Nafa et al. [2020], the DeepMatcher framework [Mudgal et al., 2018] is used for record pair comparison and classification. Both methods rely on pre-trained fastText embeddings while the RNN and hybrid architectures are used, respectively. Kasai et al. [2019] deliver promising results in terms of F1 score, showing that their active deep learning-based method achieves close to passive learning results produced by symbolic entity resolution methods. In this chapter, we develop and consider for experimental comparison a subsymbolic approach, mainly inspired by Kasai et al. [2019].

Bogatu et al. [2021] develop *VAER*, a Variational Active Entity Resolution method which aims to reduce the overall labeling effort while leveraging the advantages of deep learning by decoupling the record pair comparison and the record pair classification steps. The authors perform unsupervised representation learning on record level by exploiting variational auto-encoders, which are deep generative models typically used for dimensionality reduction [Kingma and Welling, 2014]. Variational auto-encoders consist of an encoder which aims to map the input, i.e. attribute values of a record, to a lower dimension probability distribution, and a de-

coder which ensures that the transformed input can be reconstructed to produce the original one. Once the records have been mapped to their learned numerical representations, VAER applies active learning for selecting informative record pairs for labeling, which are further used to train a siamese neural network.

Finally, Jain et al. [2021] develop DIAL, a deep indexed active learning approach for entity resolution. A significant methodological difference to the existing symbolic and subsymbolic approaches is that DIAL integrates the blocking and the matching steps in the active learning loop. Both, the matcher and the blocker rely on transformer-based pre-trained language models. In contrast to the typical pool-based scenario, which is studied in this part of the thesis, the unlabeled pool in DIAL is filled with different record pairs after every iteration, given the blocking results. Additionally, the query selection component considers the blocking results of each iteration as part of the query strategy.

Training Data Enlargement In order to circumvent the absence of large amounts of training data during active learning, which can lead to the overfitting of the learner [Anthony and Biggs, 1997], training data enlargement techniques have been applied in related work [Kasai et al., 2019; Wang et al., 2021b]. The common ground of the works using training data enlargement techniques is that they do not increase the labeling effort as they solely rely on confidently predicted unlabeled record pairs by the query selection component. The confident unlabeled pairs receive their predicted label and are added to the labeled set. Wang et al. [2021b] apply a committee-based query strategy and calculate the confidence of the predictions of the unlabeled record pairs as the agreement among the votes of the committee members. Kasai et al. [2019] apply a learner-aware margin-based query strategy and calculate the confidence of the predictions of the unlabeled record pairs as the prediction probability of the learner.

Use of External Validation Set A critical observation that can be made from some of the related works applying active learning for entity resolution, is that they use an external validation set, e.g. [Kasai et al., 2019; Nafa et al., 2020; Wang et al., 2011]. We report which works use external validation sets as well as their relative sizes in Table 7.1 and column *Use of external validation set (size)*. The relative size denotes the ratio of all record pairs after blocking for each task used in the experiments of the respective works. The validation set is exploited for tuning the hyperparameters of the models, which are part of the active learning pipeline, or for model selection. Although benchmark entity resolution tasks should be accompanied by validation sets to allow a fair comparison of passive entity resolution methods, as discussed in Chapter 3, exploiting them in an active learning setting perplexes the interpretation of the active learning results. One cannot differentiate if the achieved performance is due to the labeled set, gathered during active learning, or the external validation set. Kasai et al. [2019] use a validation set of 20% of the record pairs after blocking for model selection. Nafa et al. [2020] assume the existence of a validation set for fine-tuning the model of their query selection component. For the evaluation of their method, Nafa et al. [2020] use the DeepMatcher

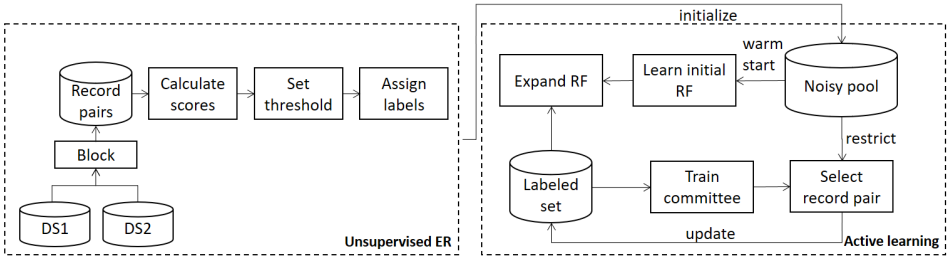


Figure 7.1: Workflow of unsupervised bootstrapping of active learning for entity resolution.

entity resolution tasks [Mudgal et al., 2018] along with the provided splits for training (60%), validation (20%) and testing (20%). In order to test the robustness of their method, the authors vary the size of the validation set from 25% to 100% of the original provided validation set. This results in multiple validation sets per task containing a minimum of 5% and a maximum of 20% of the record pairs after blocking. Bogatu et al. [2021] exploit the validation sets that come together with the benchmark entity resolution tasks used for experimentation to fine-tune the hyper-parameters of both components of their variational active resolution model, i.e. the representation learning step as well as the active learning step. However, the size of the validation sets is not explicitly reported. Similarly, Wang et al. [2021b] use a validation set comprising 20% of the record pairs after blocking for fine-tuning the parameters of the learner. In our experiments, we account for this observation and do not use an external validation set.

7.2 Methodology

We tackle the cold start problem of active learning by applying an unsupervised entity resolution method for bootstrapping, i.e. initializing and assisting, the active learning workflow. Our method comes at no additional labeling cost. In this section, we describe the details of our proposed method for bootstrapping active learning for entity resolution. Our method comprises two main methodological steps: unsupervised entity resolution and active learning. The unsupervised entity resolution step relies on the aggregation of the similarity-based features of all record pairs in the unlabeled pool and a thresholding heuristic. The output of this step is a pool of unsupervised labeled and weighted record pairs. The unsupervised entity resolution step will be described in Section 7.2.1. In the active learning step, the unsupervised labeled record pairs are used for initializing the active learning workflow, but also as part of the query selection and learner components. The active learning step will be described in Section 7.2.2. Figure 7.1 gives an overview of the complete workflow of our method.

7.2.1 Unsupervised Entity Resolution

In order to bootstrap the active learning workflow, we apply an unsupervised entity resolution method and assign weighted matching and non-matching labels to the record pairs of the unlabeled pool. We remind the reader that the unlabeled pool construction is typically preceded by a blocking step.

Unsupervised entity resolution methods can be distinguished into threshold-based, rule-based, and clustering-based [Christen, 2012; Papadakis et al., 2021]. Threshold-based methods use a threshold value, which can be either pre-defined or calculated, for splitting the record pairs into matching and non-matching, given their aggregated attribute similarities [Kejriwal and Miranker, 2015; Oulabi and Bizer, 2019]. Rule-based methods employ matching rules for classifying the record pairs as matching or non-matching. A matching rule is comprised of similarity scores combined with conjunctions, disjunctions, and negations [Christen, 2012]. Clustering-based methods apply different clustering algorithms, such as k-means [Elfeky et al., 2002], and create clusters of potentially matching and non-matching record pairs. Considering that typically rule-based and clustering-based techniques require manually designing matching rules or setting a set of clustering parameters, we employ a threshold-based method. Below we describe the details of the unsupervised entity resolution step.

Similarity Score Aggregation

The unsupervised matching step starts by calculating similarity-based features for each record pair after blocking. We use the data type-specific similarity metrics on attribute level, as described in Section 2.3.3, and generate a numerical feature vector for each record pair. Additionally, we calculate the cosine score with tf-idf weighting over the concatenated values of all attributes. The overall cosine similarity score is added as an additional feature to the feature vector. In the case that the similarity score cannot be computed for an attribute combination, because either one or both values are missing, we assign the out-of-range score -1. This allows any classifier to consider the relevant record pairs without dropping or replacing the missing values.

We summarize the feature vector values into one value per record pair and assign this score as its aggregated similarity score. A similarity score close to 1 gives a strong signal that the record pair matches, whereas a similarity score close to 0 indicates that it does not match. We calculate the aggregated similarity score per record pair as a weighted linear combination of all its non-missing feature values. The overall cosine similarity receives a weight of 0.5, while all other features share equally a weight of 0.5. We additionally weigh every feature value with the overall density of the corresponding feature. The rationale behind this weighting scheme is that we assume features that derive from attributes with few missing values to be more important for solving the matching task in comparison to features deriving from non-dense attributes. The aggregated similarity score s_p of a record pair p is calculated using Equation 7.1.

$$s_p = 0.5 \times \text{cosine\_tfidf} + 0.5 \times \frac{\sum_{i=1,\mathrm{f}_{ip}\neq -1}^{n} \mathrm{f}_{ip} \times d_i}{|\mathrm{f}_{ip} \neq -1|} \tag{7.1}$$

where:

n = the number of the features of record pair p
f_{ip} = the value of feature f_i of record pair p
d_i = the density of feature f_i

Thresholding

After the feature values have been aggregated to one similarity score per record pair, a threshold value needs to be defined for assigning matching labels to the record pairs in an unsupervised fashion. The record pairs with an aggregated similarity score above the threshold value are assigned the label *matching*. Otherwise, they are assigned the label *non-matching*. Typically, thresholding methods are static with the threshold being 0.5, i.e. the middle value of the similarity score range [Kejriwal and Miranker, 2015; Oulabi and Bizer, 2019].

Another thresholding technique that has been explored for the task of image segmentation [Sezgin and Sankur, 2004], is Otsu's thresholding method [Otsu, 1979]. Otsu's method selects as a threshold the value that maximizes the variance between the two classes and therefore expects that the distribution of values is bimodal, i.e. two clear peaks appear in the histograms of similarity scores without any long-tail values.

To bypass the bimodality assumption of Otsu's method, Ng [2006] developed a variation of Otsu's method, known as the *valley-emphasis* threshold. The valley-emphasis threshold, which has also been used in the area of image segmentation, is calculated using Equation 7.2, where p_t is the relative frequency of grayscale t, ω is the probability of each class and μ is the mean gray-level value of each class.

$$t\_valley = ArgMax\left\{(1-p_t)\left(\omega_1(t)\mu_1^2(t) + \omega_2(t)\mu_2^2(t)\right)\right\} \tag{7.2}$$

For the task of image segmentation, the relative frequency is calculated as $p_t = n_i/n$, where n_i is the number of occurrences of gray level i and n is the total number of pixels. We adjust the valley-emphasis method to fit the entity resolution task by performing the following two adaptations: (i) We round the similarity scores to the second decimal and calculate n_i as the frequency of the rounded similarity score. In this way, we aggregate the occurrences of infrequent values, which allows for reasonable n_i, as the similarity scores can have an arbitrary number of decimal digits. (ii) We set n as the number of occurrences of the most frequent similarity score and not to the total number of record pairs, which would be the direct equivalent to the number of pixels. The reason for this adaptation is to allow

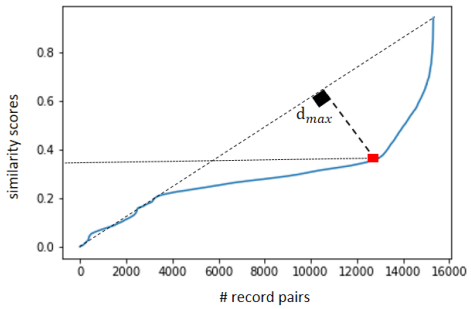


Figure 7.2: Elbow point at 0.368 of the cumulative histogram of similarity scores of record pairs from an example ER task.

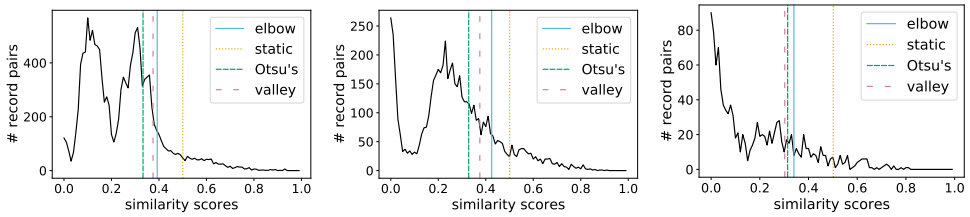
the valley-emphasis method to have an effect over Otsu's method, as otherwise the weighting factor $(1 - p_t)$ will always be very close to 1.

Additionally to the existing thresholding methods, we propose a novel thresholding method that determines the threshold value as the *elbow point* (also denoted knee or point of maximum curvature [Satopaa et al., 2011]) of the cumulative histogram of the similarity scores of all record pairs after blocking. The elbow value can be approximated as the point with the maximum perpendicular distance to the vector between the first and the last point of the cumulative histogram. Figure 7.2 shows the elbow point of the cumulative histogram of similarity scores for an example entity resolution task. From the histogram, we can see that 12.8K pairs in this example entity resolution task have a similarity score below the elbow point.

The elbow thresholding method adjusts to the similarity score distributions of different entity resolution tasks in contrast to static thresholding, while it does not assume bimodality like Otsu's thresholding. To further illustrate this, we present the similarity scores for three entity resolution tasks in Figure 7.3. The three tasks belong to different profiling groups, defined in Chapter 3: the task of Figure 7.3a belongs to *Group 1: Dense Data, Simple Schema*, the task of Figure 7.3b belongs to *Group 3: Sparse Data, Complex Schema*, and the task of Figure 7.3c belongs to *Group 5: Textual Data, Many Corner Cases*. It becomes obvious that the similarity score distributions among different entity resolution tasks can vary significantly, e.g. the task of Figure 7.3a appears to have a clear bimodality, while the similarity scores of the record pairs of the task of Figure 7.3c resemble a power-law distribution. Therefore, it becomes clear that a static threshold value is unsuitable for all tasks. Additionally, Otsu's threshold, even when bimodality appears, is moved towards the long tail of the distribution (Figure 7.3a). Finally, the adjusted valley and the proposed elbow method produce similar threshold values.

Confidence Weights

Apart from their unsupervised matching labels, the record pairs are assigned weights, which indicate how confident our unsupervised method is for the predicted label.



(a) ER task of Group 1 **(b)** ER task of Group 3 **(c)** ER task of Group 5

Figure 7.3: Histograms of similarity scores of ER tasks of different profiling groups and threshold boundaries per method.

This is necessary as the record pairs that are expected to be noisier should affect less the active learning step in comparison to more confident pairs. The confidence weight of a record pair is calculated as the normalized distance of its aggregated similarity score s_p to the threshold value t.

Therefore, record pairs close to the decision boundary t will receive a confidence weight close to 0, while record pairs whose similarity scores are the highest or the lowest in the similarity score distribution will receive a confidence weight close to 1. We use the Equation 7.3 for calculating the confidence weight c_p of a record pair p, given a threshold value t and a similarity score distribution S.

$$c_p = \begin{cases} \frac{|s_p - t|}{t - \min(S)} & , \text{if } s_p < t \\ \frac{|s_p - t|}{\max(S) - t} & , \text{if } s_p > t \\ 0 & , \text{if } s_p = t \end{cases} \tag{7.3}$$

7.2.2 Active Learning

Considering the findings of Meduri et al. [2020] and Chen et al. [2019], as discussed in the related work Section 7.1, we use a heterogeneous learner-aware committee-based query strategy and a random forest classifier as the learner for the active learning step. Apart from its good performance in active learning settings [Meduri et al., 2020], the random forest model allows for incremental training by gradually adding new trees to the forest, a characteristic which is used in our methodology and will be explained later in this section in more detail.

In a typical pool-based active learning setting, the pool contains unlabeled record pairs. In our proposed method, the pool contains unsupervised labeled record pairs, subject to some degree of noise. Thereafter, we will refer to the pool of the active learning step of our methodology as *noisy pool*, to differentiate it from the typical active learning setting. The existence of labels in the noisy pool allows us to initialize the models of the query selection and learner components. Our initialization method comes at no additional labeling cost, in contrast to the

majority of active learning methods for entity resolution that rely on a manually labeled seeding set of record pairs, as described in Section 7.1. In the following, we describe how the noisy pool is exploited for initializing the query selection and learner components as well as for assisting the complete active learning workflow.

Initializing the Learner

Before starting the active learning workflow, we use the record pairs of the noisy pool and train a random forest classifier using the default parameters of the model, as set from the scikit-learn library version 0.19.2[3]: 10 estimators, i.e. trees, a minimum split size of 2, while allowing sample replacement and maximum depth. The confidence weights of the record pairs are considered as training weights upon learning. This allows near to zero weighted leaf nodes of the individual trees of the random forest classifier to be ignored. In this way, we avoid the over-fitting of the initial random forest classifier to the most unconfident record pairs of the noisy pool.

Initializing the Committee

We bootstrap the models of the committee for our query strategy by adding one most confident positive and one most confident negative pair of the noisy pool to the labeled set. The initialized labeled set is used for training the committee models of the first active learning iteration.

Refining the Committee

We use a heterogeneous committee to select the most informative record pair for labeling. This has been shown to perform better than committees of the same classification model with different model parameterizations [Chen et al., 2019]. Our committee comprises five linear and non-linear classification models with default parameters[4]: logistic regression, linear SVM, decision tree, XGBoost, and random forest. The first four classifiers have been shown to achieve good accuracy with little training data [Chen et al., 2019]. As we use a random forest classifier for the learner component of the active learning workflow, we add this classifier to the committee. In every active learning iteration, each model in the committee is trained on the current labeled set. Next, it votes its predictions on all record pairs of the noisy pool, i.e. every record pair receives five votes. The record pairs with the maximum disagreement are considered to be the most informative. We measure disagreement using vote entropy, calculated with Equation 7.4. A further possibility would be to add weights to the votes of the committee and calculate a weighted vote entropy considering the prediction probabilities of the committee models. However, as the committee comprises different types of classification

[3] https://scikit-learn.org/0.19/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[4] As defined by the scikit-learn library version 0.19.2

models, we expect their probability distributions to be distorted and hard to compare [Niculescu-Mizil and Caruana, 2005]. Although calibration methods exist for adjusting the predictions of different classification models to the actual posterior probabilities, such as Platt Calibration [Platt et al., 1999] and Isotonic Regression [Zadrozny and Elkan, 2001], these have not been studied in our work.

We restrict the number of most informative pairs to those whose majority vote disagrees with the unsupervised label of the record pair. From this restricted set, one pair is randomly selected for labeling. In this way, we aim to select pairs to query whose unsupervised label might be wrong and can therefore lead to the addition of new information to the random forest model learned during initialization.

$$vote\_entropy_p = -\sum_{i=1}^{|Y|} \frac{V(y_i)}{C} \log \frac{V(y_i)}{C} \tag{7.4}$$

where:

Y = set of labels
$V(y_i)$ = number of votes for label y_i
C = number of members in the committee

Refining the Learner

We propose the incremental training of the learner in each active learning iteration to allow for a gradual *fading away* effect of the model learned in the initialization phase. To do so, we use a random forest classifier as the learner to which we gradually add more estimators, i.e. trees. Therefore, the model learned in the previous query iterations is not overwritten, a common practice in active learning settings [Chen et al., 2019; Meduri et al., 2020; Sarawagi and Bhamidipaty, 2002], but expanded. We start our training (active learning iteration 0) with the initialization of the learner, as explained previously, by fitting an initial number of trees on the noisy pool of record pairs. Each active learning iteration adds a small number of new trees to the model of the previous iteration. The added trees are trained on the current labeled set. In the early training iterations, we expect the added estimators to be of low quality and high disagreement on their predictions, as they are trained on small amounts of clean data. Therefore, in the early iterations, the initial learner, trained in the initialization phase, dominates its predictions over the ones of the added estimators. Once the added estimators become of better quality, given the expansion of the labeled set, their prediction agreements will increase, dominate the ones of the initial model and lead to model correction. We set the number of trees learned in the initialization phase to 10 and the incremental size of estimators per iteration to 2. In comparison to a regular random forest model, which is not expanded but retrained, our learner retains a "fade-away" effect of the noisy pool of record pairs. Finally, the incremental training of the learner resembles the boosting rounds of a gradient-boosted tree model, such as the XGBoost

Table 7.2: Profiling information of the ER tasks of the experimental setup.

Profiling group	Task	Attributes	Pool pairs		Test	
			# match.	# non-match.	# match.	# non-match.
Group 1: Dense data, simple schema	dbpedia_dnb	4	2,310	11,554	577	2,888
	dbpedia_viaf	5	2,552	12,764	801	4,006
Group 3: Sparse data, complex schema	wdc_phones	18	206	1,556	51	389
	wdc_headphones	14	180	983	45	245
Group 5: Textual data, many corner cases	abt_buy	3	878	4,854	219	1,213
	amazon_google	4	1,041	5,714	259	1,428

model [Chen and Guestrin, 2016]. The difference is that in each iteration, which would correspond to a boosting round in the case of XGBoost, the trained model does not focus on the errors of the model of the previous iteration but on the clean labeled record pairs.

7.3 Experimental Evaluation

In this section, we present the experimental results of our proposed method. First, a detailed description of the experimental setup is given. Next, we split and discuss the experimental results into three parts. In the first part, we present the results of existing thresholding methods and compare them to the results of the elbow thresholding method, described in Section 7.2.1. In the second part, we compare the results of our active learning method to symbolic active learning baselines. Finally, in the third part of our evaluation, we compare the results of our active learning method to subsymbolic active learning baselines, inspired by the work of Kasai et al. [2019].

7.3.1 Experimental Setup

We use six entity resolution tasks for our experimental evaluation from the author (two tasks) and the product (four tasks) domains. The two author entity resolution tasks belong to the profiling *Group 1: Dense Data, Simple Schema*, as defined in Chapter 3. The two tasks include records describing authors retrieved from DBpedia and linked to the DNB[5] and VIAF[6] data sources using *owl: sameas* links. In total, there exist 2,887 matching record pairs between DBpedia and DNB and 3,353 matching record pairs between DBpedia and VIAF. The DBpedia and DNB data sources have the following attributes in common: *author_name*, *birthdate*, *deathdate*, and *gender*. Between DBpedia and VIAF, the attributes *author_name*, *birthdate*, *deathdate*, *gender*, and a list of *works* are provided.

[5]https://www.dnb.de/wir

[6]http://viaf.org/

We use two e-commerce tasks with records describing *phones* and *headphones* from the Web Data Commons Project[7] with multiple non-dense attributes. Therefore these two tasks belong to *Group 3: Sparse Data, Complex Schema*, as defined in Chapter 3. For our experiments, we disregard all attributes that have a density lower than 0.10. After this filtering, the records of the *wdc_phones* task have attributes with densities ranging between 11% and 93%. The densities of the attributes for the *wdc_headphones* range between 10% and 91%.

Additionally, we consider the *abt_buy* and *amazon_google* e-commerce tasks [Köpcke and Rahm, 2008] which belong to *Group 5: Textual Data, Many Corner Cases*, as defined in Chapter 3. The records of the abt_buy task contain the attributes *product name*, *product description*, and *product price*. The records of the amazon_google task contain the attributes *product name*, *product description*, *manufacturer*, and *price*.

Finally, we remove the labels of the record pairs of the train split of each task. In a typical pool-based active learning setting, these are the pairs to be added to the unlabeled pool. In our setting, these pairs receive labels, using the unsupervised matching technique described in Section 7.2.1, and are added to the noisy pool before active learning starts. Table 7.2 summarizes the profiling information of the tasks used for experimentation in terms of record attributes, as well as the number of matching and non-matching training and test record pairs.

7.3.2 Comparison to Thresholding Baselines

We evaluate the elbow point thresholding method and compare it to static thresholding, for which the threshold is set to 0.4 and 0.5. Additionally, we perform a comparison to two thresholding methods for binary problems from the field of image segmentation, Otsu's method [Otsu, 1979] and the valley-emphasis method, [Ng, 2006] after the adjustments explained in Section 7.2.1.

Table 7.3 presents the results of the five compared thresholding methods in terms of sample correctness (accuracy) and F1 score. To put the unsupervised results into context, we present the difference Δ to the F1 score, achieved in a passive supervised learning scenario in which all training record pairs are manually labeled and used to train a random forest classifier.

Comparing the thresholding methods' results, we observe that our proposed elbow point method achieves better results in terms of F1 score for five of the six tasks in comparison to static thresholding when the threshold value is set to 0.5, with the exception of wdc_phones where it underperforms by 2%. For the rest of the tasks, the elbow method significantly dominates static@0.5 thresholding by an absolute F1 score difference varying from 1% to 20%. In comparison to the static thresholding when the threshold value is set to 0.4, the elbow method achieves better results in four of the six tasks. However, the absolute F1 score difference is smaller in comparison to the difference to the static@0.5 threshold and up to 8.4%.

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

Table 7.3: Comparison of thresholding methods and difference to passive learning.

Task	Thresholding method	Unsupervised Accuracy	Unsupervised F1	Passive F1	Δ to passive F1
dbpedia_dnb	elbow	0.918	0.722	0.976	**-0.254**
	static@0.4	0.920	0.721		-0.255
	static@0.5	0.894	0.538		-0.438
	Otsu's	0.833	0.602		-0.374
	valley	0.906	0.707		-0.269
dbpedia_viaf	elbow	0.956	0.862	0.983	**-0.121**
	static@0.4	0.957	0.857		-0.126
	static@0.5	0.915	0.663		-0.320
	Otsu's	0.743	0.542		-0.441
	valley	0.958	0.861		-0.122
amazon_google	elbow	0.892	0.588	0.729	-0.141
	static@0.4	0.899	0.624		**-0.105**
	static@0.5	0.882	0.441		-0.288
	Otsu's	0.825	0.600		-0.129
	valley	0.827	0.602		-0.127
abt_buy	elbow	0.896	0.674	0.818	**-0.144**
	static@0.4	0.878	0.657		-0.161
	static@0.5	0.912	0.660		-0.158
	Otsu's	0.794	0.562		-0.256
	valley	0.857	0.630		-0.188
wdc_phones	elbow	0.881	0.523	0.851	-0.328
	static@0.4	0.762	0.439		-0.412
	static@0.5	0.881	0.544		**-0.307**
	Otsu's	0.759	0.438		-0.413
	valley	0.757	0.438		-0.413
wdc_headphones	elbow	0.907	0.734	0.966	-0.232
	static@0.4	0.925	0.741		**-0.225**
	static@0.5	0.898	0.539		-0.427
	Otsu's	0.877	0.682		-0.284
	valley	0.910	0.738		-0.228

Otsu's thresholding method underperforms the adjusted valley method by a maximum absolute margin of 32%. It is interesting to observe that the valley method achieves very similar results to our proposed elbow method. However, the elbow method significantly outperforms the valley method for the wdc_phones tasks by 8%. Although the elbow heuristic does not always outperform all other thresholding methods, we see that, in contrast to the other methods, it does not suffer from large differences in F1 score across the tasks. This indicates that the elbow thresholding heuristic can generalize well independently from the underlying similarity score distribution, which can greatly vary among the different tasks. Finally, the elbow thresholding method achieves 11% - 32% lower results in terms of F1 score in comparison to the results achieved with full supervision and has an accuracy of 88% or higher. For the rest of the experimental evaluation, we consider the elbow thresholding method.

7.3.3 Comparison to Symbolic Active Learning Baselines

We compare our proposed method, which we abbreviate with *boot* in the rest of the experimental section, to three symbolic active learning baseline methods. The first baseline method uses a random forest classifier as learner, the committee-based query strategy of HeALER [Chen et al., 2019] and random initialization. The second baseline method is similar to the first, while it additionally uses an incrementally learned random forest classifier, similarly to *boot*. The third baseline method, inspired by the initialization method used in the work of Kasai et al. [2019], replaces the unsupervised initialization of *boot* with transfer learning. In the following, we present the details of the three symbolic baseline methods.

B1: Random Forest - Random Initialization

As the first baseline, abbreviated with *no_boot*, we consider an active learning setting with a pool containing all record pairs without labels or weights and an initially empty labeled set. As a query strategy, we apply first random sampling until at least one positive and one negative pair is included in the labeled set. After that, we apply the HeALER query-by-committee strategy, as described in Section 7.2.2. A random forest classifier is trained in every iteration with the pairs of the labeled set, using 10 estimators. With this baseline method, we aim to compare the two initialization techniques for active learning, i.e. random sampling and unsupervised entity resolution with the elbow thresholding heuristic.

B2: Random Forest with Warm Start - Random Initialization

The second baseline, abbreviated with *no_boot_warm*, is designed in the same way as the first one apart from the model training step. In this case, we use a warm start setting like in our proposed method, with a random forest classifier being incrementally expanded. Once the labeled set includes at least one positive and one negative pair, an initial random forest is trained using 10 estimators. Similar to our *boot* approach, in every iteration, two new estimators, i.e. trees, are trained on the labeled set and are added to the initial random forest classifier. This baseline guarantees that in every iteration the same number of trees as in the *boot* setting is retrained. In addition, it ensures that the total number of estimators of the random forest classifier in every iteration is the same as the one used for our method.

B3: Random Forest with Warm Start - Transfer Learning Initialization

The third baseline, abbreviated with *boot_TL*, uses transfer learning instead of unsupervised matching for initializing active learning. In this setting, we consider the existence of two entity resolution tasks for each experiment: a *source* task, for which all record pairs after blocking have been manually labeled, and a *target* task, for which no labeled record pairs are available and is the task we aim to solve. The source and target tasks need to be from the same topical domain and have the same

Table 7.4: Task relatedness scores per pair of tasks of the same profiling group.

Pair of ER tasks		Task relatedness
dbpedia_dnb	dbpedia_viaf	0.442
abt_buy	amazon_google	0.134
wdc_phones	wdc_headphones	0.212

schema. In our experimental setting, finding a source task is feasible, as there exist three pairs of related entity resolution tasks. For the cases that the schemata of the source and target tasks do not completely match, we reduce the feature vector to the features that can be derived from the intersection of the source and target schemata.

The only methodological difference between the *boot_TL* baseline method and our method *boot* is that the initialization step relies on transfer learning and not on unsupervised matching. More concretely, we train a random forest classifier on all labeled record pairs of the source task. We apply the trained model to the record pairs of the target task and derive their predicted labels *match* or *non-match*. Similar to the confidence-based weighting scheme of the *boot* method, we use instead of the normalized distance to a threshold value, the prediction probability of the random forest classifier trained on the source task and assign weights to all record pairs of the target task. We use the labeled and weighted record pairs of the target task to initialize the noisy pool. Afterward, we apply the active learning steps described in Section 7.2.2.

We expect some of the tasks of our experimental setting to be more closely related than others which can, in turn, affect the initialization step and influence positively the performance of the learner during active learning. For example, the tasks dbpedia_dnb and dbpedia_viaf are expected to be more related than the tasks amazon_google and abt_buy, as the first share one data source while the latter do not. One way of measuring how related two entity resolution tasks are, is to compute the task relatedness (TR), a metric introduced by Thirumuruganathan et al. [2018]. TR calculates how similar two tasks are by training a logistic regression classifier to predict the task from which each record pair originates. A high prediction quality signifies that the two tasks are dissimilar, while a low prediction quality signifies that the tasks are similar and are expected to have the same underlying matching patterns. We measure the prediction quality of the classifier using the Matthews correlation coefficient (MCC) and calculate the TR score as $1 - MCC$, similar to Thirumuruganathan et al. [2018]. Table 7.4 presents the task relatedness scores for the three pairs of tasks in our experimental setting. As expected given the shared data source, the tasks dbpedia_viaf and dbpedia_dnb have the highest relatedness score among the other task combinations.

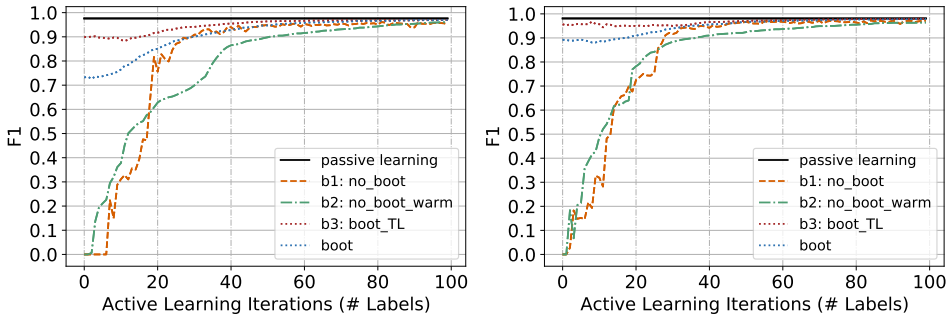
Results

We run each active learning experiment three times and allow an annotation budget of 100 labeled record pairs. We use single queries, i.e. in each active learning iteration, one record pair is selected, labeled, and added to the labeled set. We report the average F1 scores per iteration and the standard deviation (σ) to account for model stability on a separate test set. All experiments are run on a Linux server with Intel Xeon 2.4 GHz processors.

Figures 7.4, 7.5, and 7.6 show the F1 scores per iteration for the *boot* method in comparison to the three symbolic baseline methods. As we use single queries, each iteration equals to one annotation, e.g. on iteration 20, we have labeled 20 record pairs. Additionally, we indicate the upper learning bound of passive learning in which all available training data are used.

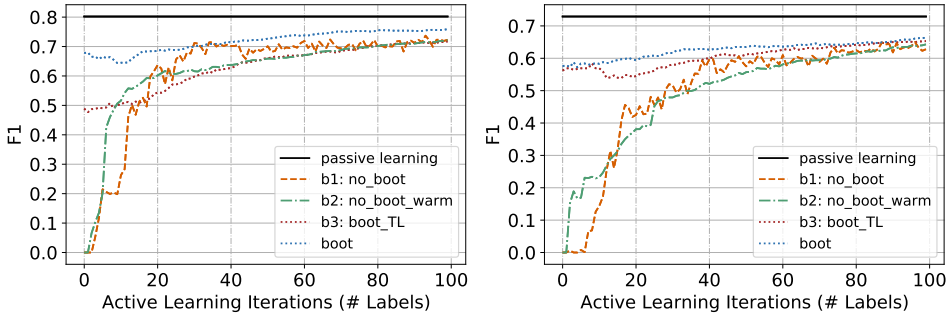
We observe that for all tasks, our method manages to solve the cold start problem. In the first active learning iterations (1-40 depending on the tasks), *boot* produces better F1 scores in comparison to the two baselines that use random initialization for all tasks. In comparison to the *boot_TL* baseline method, the results significantly vary. For the dbpedia_dnb and dbpedia_viaf tasks, initializing active learning with transfer learning produces better results during the first iterations in comparison to the unsupervised initialization. Already in the very first active learning iteration, the learner of the *boot_TL* baseline achieves F1 scores close to the passive learning results for these two tasks, i.e. 0.899 for dbpedia_dnb which is 7.7 percentage points lower than the passive F1 score and 0.957 for dbpedia_viaf which is 2.6 percentage points lower than the passive F1 score. However, this is not the case for the rest of the tasks in our experimental setting for which the *boot* method outperforms the *boot_TL* baseline in the first active learning iterations. The high transferability between the dbpedia_dnb and dbpedia_viaf tasks is attributed to their high task relatedness, which is the highest among all task combinations, as shown in Table 7.4. Considering an active learning setting with a limited budget in terms of manual annotations and no labeled data from another highly related task, our method is preferable as stopping at any iteration achieves acceptable results.

Once the baseline methods go through the cold start phase, their F1 curves approach the one of the *boot* method. Comparing our method *boot* to the baselines *no_boot* and *no_boot_warm*, which use random initialization, we observe that the curves overlap after approximately 30 iterations for the tasks of *Group 1: Dense Data, Simple Schema*, signifying that the unsupervised bootstrapping does not contribute to learning a better model in terms of quality. However, this is not the case for the tasks of the other two groups. For these tasks, the *boot* F1 curve dominates the *boot* and *no_boot_warm* baseline F1 curves until the final iteration or until the model converges to the upper learning bound of passive learning - a situation that happens for the wdc_headphones task. Therefore, the unsupervised bootstrapping continues to help to learn models of better quality in terms of F1 score, even after the cold start phase has passed for the tasks of *Group 3: Sparse Data, Complex Schema*, and *Group 5: Textual Data, Many Corner Cases*. A final observation that



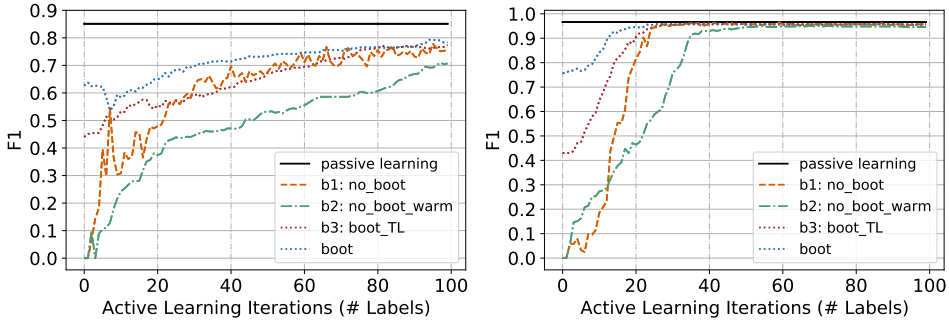
(a) dbpedia_dnb **(b)** dbpedia_viaf

Figure 7.4: Symbolic baselines - F1 per active learning iteration - Group 1 tasks.



(a) abt_buy **(b)** amazon_google

Figure 7.5: Symbolic baselines - F1 per active learning iteration - Group 5 tasks.



(a) wdc_phones **(b)** wdc_headphones

Figure 7.6: Symbolic baselines - F1 per active learning iteration - Group 3 tasks.

Table 7.5: Comparison to symbolic AL baselines.

Task	AL method	Latency	F1-AUC	F1(σ) 20th iter.	F1(σ) 60th iter.	F1(σ) 100th iter.
dbpedia_dnb	no_boot	0.91 sec.	79.29	0.756(0.197)	0.953(0.007)	0.952(0.009)
	no_boot_warm	0.96 sec.	76.40	0.628(0.317)	0.916(0.022)	0.961(0.009)
	boot_TL	1.18 sec.	**93.77**	0.916(0.020)	0.966(0.002)	**0.969(0.002)**
	boot	1.21 sec.	90.06	0.850(0.022)	0.958(0.008)	**0.969(0.001)**
dbpedia_viaf	no_boot	1.11 sec.	81.70	0.725(0.363)	0.967(0.005)	0.972(0.005)
	no_boot_warm	0.87 sec.	81.76	0.782(0.108)	0.937(0.043)	0.964(0.014)
	boot_TL	0.90 sec.	**95.70**	0.950(0.017)	0.973(0.004)	**0.980(0.002)**
	boot	1.39 sec.	93.91	0.909(0.014)	0.970(0.006)	0.979(0.002)
abt_buy	no_boot	0.63 sec.	61.16	0.637(0.080)	0.719(0.034)	0.723(0.031)
	no_boot_warm	0.66 sec.	61.15	0.602(0.086)	0.671(0.046)	0.722(0.039)
	boot_TL	0.75 sec.	62.31	0.538(0.076)	0.670(0.048)	0.717(0.029)
	boot	0.78 sec.	**71.15**	0.685(0.048)	0.738(0.033)	**0.759(0.029)**
amazon_google	no_boot	0.60 sec.	49.53	0.425(0.214)	0.610(0.063)	0.628(0.046)
	no_boot_warm	0.73 sec.	49.19	0.381(0.231)	0.581(0.063)	0.643(0.049)
	boot_TL	0.77 sec.	59.91	0.542(0.094)	0.624(0.061)	0.653(0.053)
	boot	0.85 sec.	**62.00**	0.594(0.055)	0.636(0.041)	**0.663(0.034)**
wdc_phones	no_boot	0.57 sec.	61.13	0.480(0.137)	0.712(0.063)	0.755(0.058)
	no_boot_warm	0.56 sec.	47.72	0.374(0.330)	0.555(0.206)	0.707(0.077)
	boot_TL	0.43 sec.	64.51	0.542(0.150)	0.692(0.064)	0.774(0.031)
	boot	0.58 sec.	**70.61**	0.649(0.050)	0.747(0.053)	**0.783(0.027)**
wdc_headphones	no_boot	0.53 sec.	80.98	0.816(0.242)	0.957(0.008)	0.957(0.008)
	no_boot_warm	0.59 sec.	75.71	0.464(0.386)	0.948(0.006)	0.946(0.004)
	boot_TL	0.47 sec.	88.45	0.878(0.104)	0.957(0.005)	0.957(0.005)
	boot	0.68 sec.	**92.54**	0.945(0.033)	0.955(0.007)	**0.957(0.005)**

can be drawn from the three figures, is that the *no_boot_warm* baseline underperforms the *no_boot* baseline in every active learning iteration. This shows that the warm start setting can perform well only when the initially learned model is of acceptable quality. The latter is guaranteed when active learning is initialized with unsupervised labeled record pairs but not otherwise. Comparing the F1 curves after the cold start phase of our method *boot* to the one of the *boot_TL* baseline, we see that they converge to the same results after a maximum of 80 iterations, even for tasks with low relatedness with abt_buy being the only exception. For the abt_buy task, *boot* continues to outperform *boot_TL* even after 100 iterations.

Table 7.5 presents the comparison results of our method abbreviated with *boot* and the symbolic active learning baselines in three active learning snapshots: 20th, 60th, and final 100th iteration. For each snapshot, we report the F1 score and standard deviation σ of the F1 scores among the five experimental runs. Additionally, we report the average waiting time per iteration in seconds, i.e. the time between two queries, which we denote as *latency*, as well as the area under the F1 curve for all iterations, which we denote as *F1-AUC*.

Concerning latency, we observe that all symbolic active learning methods have very small differences in waiting times per iteration. The waiting time increases slightly with the size of the task at hand, e.g. dbpedia_dnb needs approximately 1 sec. per iteration, while the time reduces in half for wdc_phones.

With regards to the performance of the symbolic active learning methods over all 100 iterations, which we measure with the area under the F1 curve (F1-AUC), we notice that the two baseline methods using random initialization achieve similar results in three out of the six matching tasks. The *boot* method is the clear winner among these two baselines achieving 9.48-12.47 points in F1-AUC more among all tasks. This indicates that the *boot* method performs best over all iterations in comparison to the *no_boot* and *no_boot_warm* methods, which was already clear from the learning curves presented in Figures 7.4-7.6. Comparing the F1-AUC scores of the *boot* and *boot_TL* methods, we see that the first underperforms the second by 3.71 and 1.79 points for the dbpedia_dnb and dbpedia_viaf tasks, respectively. This indicates that when labeled data from a highly related task are available, initializing active learning using transfer learning performs overall better than using unsupervised initialization. As already discussed, this is not the case for the rest of the tasks for which *boot* outperforms *boot_TL* by 2.02 to 8.84 F1-AUC points. With regards to the single snapshots, we observe that the improvement of our method over the baselines that use random initialization is more dominant during the cold start phase and up to 48% after 20 active learning iterations, i.e. the difference in F1 score between the *boot* and *no_boot_warm* results for the task wdc_headphones.

Concerning stability, we see that already in the 20th iteration, the *boot* method gives stable results with the standard deviation ranging from 0.01 to 0.05. At the same iteration point, the *no_boot* and *no_boot_warm* baseline methods are significantly more unstable, independently from the task, with the standard deviation ranging from 0.08 to 0.38. This shows that in the cold start phase, our proposed *boot* method does not only perform better in terms of F1 score in comparison to the baseline methods using random initialization but also produces more stable models. The stability of the *boot_TL* baseline method is dependent on the relatedness of the source and target tasks, similar to the overall performance scores. More concretely, the *boot_TL* produces stable results in the 20th iteration for the dbpedia_dnb and dbpedia_viaf tasks, while for the rest of the tasks, which are not highly related, the standard deviation ranges from 0.07 to 0.15.

On the 60th iteration, all models have recovered from the cold start phase and are therefore more stable for all methods with the exception of the *no_boot_warm* baseline, which remains highly unstable for the wdc_phones task ($\sigma = 0.206$). This extends our previous observation concerning the warm start setting, as it becomes obvious that without a good initial model, the warm start setting performs weakly in terms of both quality and stability. Finally, on the last iteration, the *boot* method produces models of at least the same stability in comparison to all symbolic baselines.

7.3.4 Comparison to Subsymbolic Active Learning Baselines

In this section of our experimental analysis, we compare our method *boot* to subsymbolic active learning methods for entity resolution from the related work. The subsymbolic baselines are adaptations of the method proposed by Kasai et al.

[2019]. As discussed in the related work Section 7.1, the deep learning-based active learning method of Kasai et al. [2019] exploits a validation set for model selection. The validation set comprises 20% of the record pairs after blocking. The use of a validation set increases the overall labeling effort, while it does not permit a clear comparison to our method, which only relies on the labeled set acquired during active learning for training the learner.

In order to be able to compare our method to the deep active learning method of Kasai et al. [2019] on common grounds, we adjust the latter so that the validation set is sampled from the labeled set acquired during active learning. In the following, we present the technical details of the subsymbolic baseline methods.

B4: DeepMatcher - Transfer Learning Initialization

General Overview The first subsymbolic baseline applies transfer learning for initializing the active learning workflow. During active learning, a learner-aware uncertainty-based strategy assesses the informativeness of the record pairs of the pool. Additionally, confident record pairs are used to enlarge the training set and a deep learning model based on the DeepMatcher architecture [Mudgal et al., 2018] is used as the learner. For tuning the parameters of the model, a validation set containing 25% of the record pairs of the enlarged labeled set is used. The latter is the only methodological difference to the method of Kasai et al. [2019] in which the validation set is external and acquired before active learning starts.

Initialization For initializing active learning, transfer learning is applied. To do so, a DeepMatcher model is trained on a similar entity resolution task, called *source task*, for which abundant labeled training record pairs are assumed to be available. The source labeled record pairs are added to the labeled set. As already discussed, in our experimental setting, finding a source task is feasible, as there exist three pairs of related entity resolution tasks.

Query Selection and Training Set Enlargement After the transfer learning step, the active learning workflow starts. In each iteration, the trained DeepMatcher model, i.e. the learner, is applied to all record pairs of the unlabeled pool and the entropy of the conditional probability distribution is computed. The entropy H for a record pair x_i, given the prediction probability $p(x_i)$ of x_i to be a match, is computed by Equation 7.5:

$$H(x_i) = -p(x_i)logp(x_i) - (1 - p(x_i))log(1 - p(x_i)) \tag{7.5}$$

A batch of 10 record pairs with the highest entropy is selected for labeling and added to the labeled set together with the corresponding labels assigned by the annotator. Additionally, a batch of 10 record pairs with the lowest entropy is considered as most confident and is added to the labeled set together with the corresponding labels predicted by the learner. Figure 7.7 depicts the entropy level

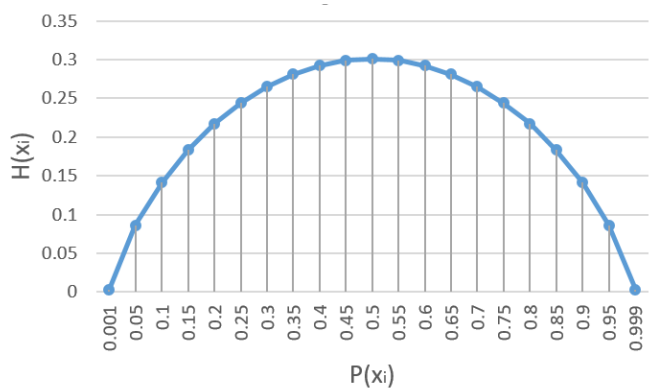


Figure 7.7: Entropy for different prediction probability scores.

for different probability scores. We observe that the highest entropy occurs when the probability score is 0.5, i.e. the classifier is unsure of its prediction. In contrast, the lowest entropy occurs when the prediction probability is close to 0 and close to 1, i.e. the classifier is confident about its negative or positive prediction.

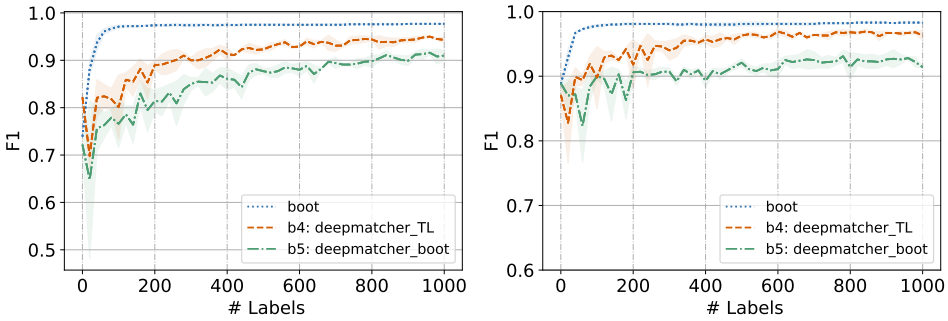
Weighting of Labeled Set The record pairs of the labeled set are weighted so that the influence of the pairs of the source task does not over dominate the record pairs of the target task. The source weights are computed by Equation 7.6:

$$source_{weight} = \frac{\text{\# labeled target pairs}}{\text{\# labeled target pairs} + \text{\# labeled source pairs}} \tag{7.6}$$

Learner In each iteration, the learner is re-trained with 75% of the record pairs in the labeled set and performs parameter optimization with the rest 25%. The learner is built using the DeepMatcher architecture [Mudgal et al., 2018] and applies pre-trained character-based fastText embeddings. In each active learning iteration, we train the learner for 20 epochs and a batch size of 20. We use the Hybrid attribute summarizer for the amazon_google and abt_buy tasks, which is a combination of RNN and Attention, since it has been shown to yield better results in comparison to other summarization techniques [Mudgal et al., 2018]. For all other tasks, we use the RNN solution as the attribute summarizer.

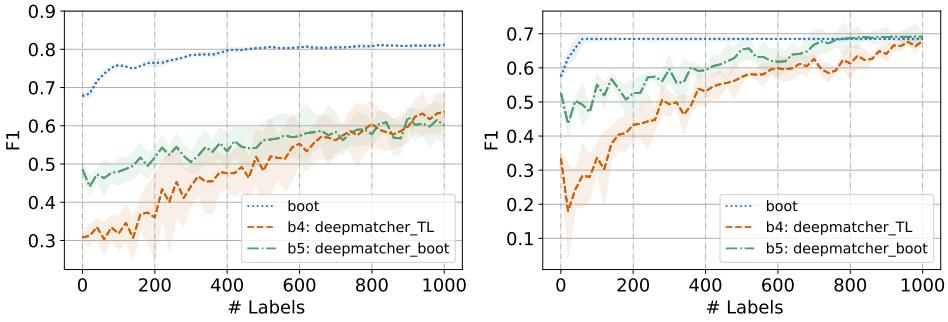
B5: DeepMatcher - Unsupervised Initialization

For the second subsymbolic baseline, we replace the transfer learning-based initialization with our unsupervised initialization method. Therefore, the labeled set is initialized with the unsupervised labeled record pairs. The re-weighting step is executed similarly to the one of the B4 baseline so that the influence of the unsupervised labels does not over dominate the manually labeled record pairs. All other steps, including the query selection strategy, the training set enlargement, and the training of the learner remain the same as the subsymbolic baseline method B4.



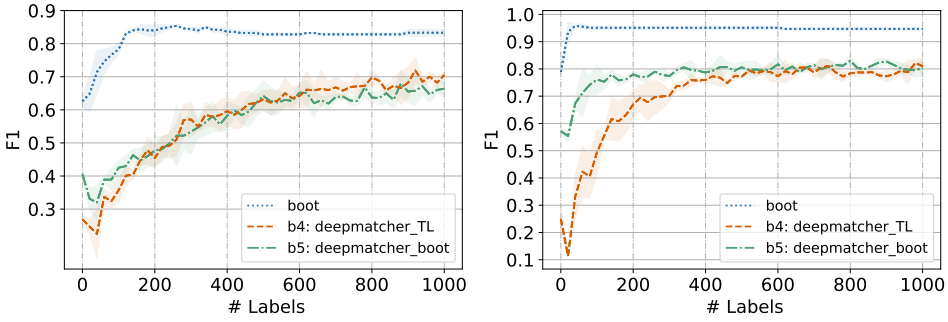
(a) dbpedia_dnb **(b)** dbpedia_viaf

Figure 7.8: Subsymbolic baselines - F1 and σ per # labels - Group 1 tasks.



(a) abt_buy **(b)** amazon_google

Figure 7.9: Subsymbolic - F1 and σ per # labels - Group 5 tasks.



(a) wdc_phones **(b)** wdc_headphones

Figure 7.10: Subsymbolic baselines - F1 and σ per # labels - Group 3 tasks.

Table 7.6: Comparison to subsymbolic AL baselines.

Task	AL method	Latency	F1-AUC	F1(σ) 2nd iter. (20 labels)	6th iter. (60 labels)	10th iter. (100 labels)	20th iter. (200 labels)	50th iter. (500 labels)	100th iter. (1000 labels)
dbpedia_dnb	**boot**	12.1 sec.	968.47	0.850(0.022)	0.958(0.008)	0.969(0.001)	0.974(0.002)	0.975(0.002)	0.977(0.001)
	deepmatcher_TL	679 sec.	906.26	0.698(0.026)	0.824(0.025)	0.801(0.068)	0.890(0.009)	0.923(0.004)	0.943(0.006)
	deepmatcher_boot	804 sec.	853.33	0.648(0.166)	0.763(0.027)	0.766(0.050)	0.814(0.023)	0.877(0.006)	0.910(0.010)
dbpedia_viaf	**boot**	13.9 sec.	977.71	0.909(0.014)	0.970(0.006)	0.979(0.002)	0.981(0.002)	0.981(0.003)	0.983(0.001)
	deepmatcher_TL	627 sec.	947.64	0.828(0.062)	0.894(0.008)	0.898(0.048)	0.918(0.039)	0.963(0.003)	0.963(0.007)
	deepmatcher_boot	930 sec.	907.51	0.870(0.026)	0.823(0.056)	0.900(0.006)	0.906(0.009)	0.921(0.006)	0.914(0.001)
abt_buy	**boot**	7.8 sec.	781.43	0.685(0.048)	0.738(0.033)	0.759(0.029)	0.765(0.012)	0.804(0.004)	0.812(0.002)
	deepmatcher_TL	1001 sec.	487.74	0.312(0.028)	0.303(0.043)	0.318(0.046)	0.360(0.108)	0.482(0.100)	0.637(0.045)
	deepmatcher_boot	645 sec.	549.23	0.441(0.025)	0.463(0.047)	0.480(0.035)	0.517(0.021)	0.562(0.049)	0.601(0.044)
amazon_google	boot	8.5 sec.	743.66	0.594(0.055)	0.636(0.041)	0.663(0.034)	0.681(0.002)	0.685(0.003)	0.685(0.005)
	deepmatcher_TL	442 sec.	521.11	0.180(0.137)	0.284(0.094)	0.336(0.042)	0.433(0.041)	0.573(0.017)	0.678(0.006)
	deepmatcher_boot	553 sec.	684.53	0.436(0.035)	0.492(0.063)	0.551(0.030)	0.526(0.043)	0.654(0.034)	**0.692(0.012)**
wdc_phones	**boot**	5.8 sec.	802.36	0.649(0.050)	0.747(0.053)	0.783(0.027)	0.840(0.017)	0.828(0.004)	0.833(0.010)
	deepmatcher_TL	54 sec.	570.83	0.247(0.021)	0.337(0.023)	0.357(0.037)	0.454(0.032)	0.632(0.027)	0.705(0.009)
	deepmatcher_boot	40 sec.	567.36	0.331(0.033)	0.389(0.008)	0.425(0.022)	0.477(0.044)	0.642(0.045)	0.664(0.026)
wdc_headphones	**boot**	6.8 sec.	946.94	0.945(0.033)	0.955(0.007)	0.957(0.005)	0.957(0.003)	0.958(0.003)	0.958(0.003)
	deepmatcher_TL	113 sec.	702.52	0.114(0.005)	0.423(0.095)	0.495(0.081)	0.670(0.047)	0.772(0.004)	0.810(0.031)
	deepmatcher_boot	83 sec.	781.47	0.554(0.018)	0.712(0.086)	0.761(0.053)	0.780(0.009)	0.806(0.017)	0.804(0.028)

Results

We run each active learning experiment three times and allow an annotation budget of 1000 labeled record pairs, similar to the experimental setup of Kasai et al. [2019]. The rationale behind allowing a much larger annotation budget in comparison to the symbolic active learning experiments is that deep learning methods need significantly larger amounts of training data in order to converge. Considering the long training times, we use batch queries of size 10 for the subsymbolic active learning baselines, i.e. 100 active learning iterations per experimental run. We report the average F1 scores per iteration and the standard deviation (σ) to account for model stability using a separate test set. Additionally, we report the average waiting time per iteration in seconds, i.e. the time between two batch queries for the subsymbolic baselines and the average time for ten single queries for our method *boot*, as well as the overall performance of each method considering the F1-AUC scores. All experiments are run on a Linux server with Intel Xeon 2.2 GHz processors and NVIDIA version 470.74 GPU.

We present the results of our method, abbreviated with *boot*, and the two subsymbolic active learning baselines B4 *deepmatcher_TL* and B5 *deepmatcher_boot*, as learning curves in Figures 7.8, 7.9 and 7.10. As we use batch queries, we plot the F1 score for different amounts of labeled pairs rather than number of iterations. Additionally, we denote the standard deviation σ using the light-colored area around the plotted F1 curves. Finally, we report the results for six different active learning snapshots in Table 7.6. For readability reasons, we highlight the method that outperforms in all iterations or the outperforming result in the final iteration.

Overall we observe that the subsymbolic active learning baselines have significantly larger latencies, due to the long training times required by deep learning-based models. The average waiting time per iteration ranges from 40 seconds for the wdc_phones task to 16 minutes for the abt_buy task. The increased latency can be prohibitive considering that for a complete active learning cycle of 100 iterations, a human annotator would need to be active for more than 26 hours.

Considering the F1-AUC scores, it becomes clear that the *boot* method has an overall better performance in comparison to the subsymbolic baseline methods for all tasks. The deepmatcher_TL and deepmatcher_boot methods underperform our symbolic active learning method, given the limited annotation budget of 1000 record pairs as it achieves the highest F1-AUC scores for all tasks. With regards to the F1 score results of specific active learning iterations, we see that our method outperforms the subsymbolic baselines for all tasks and a labeling budget of 500 record pairs by up to 32%, i.e. the difference in F1 score at the 50th iteration between *boot* and *deepmatcher_TL* for the abt_buy task. Our method still outperforms the subsymbolic baselines with a labeling budget of 1000 record pairs for all tasks apart from the amazon_google task, for which the deepmatcher_boot method outperforms. Among all evaluation tasks, the amazon_google is the task with the highest textuality. Additionally, we see that our symbolic active learning

method converges after a small number of record pairs has been labeled in comparison to the subsymbolic baselines, which still improve after 500 labels. More concretely, for the tasks wdc_headphones, dbpedia_dnb, and dbpedia_viaf, our method reaches F1 scores which are maximum 2% lower than the ones reached in the final iteration, already after 60 record pairs have been labeled. For the rest three of the tasks, our method improves only marginally (< 1%) between the labeling budgets of 500 and 1000 labels. In contrast, the difference in the F1 scores achieved by the subsymbolic methods between the labeling budgets of 500 and 1000 labels is larger and up to 15.5%.

Concerning the two initialization methods, i.e. transfer learning and unsupervised, we observe that none consistently outperforms the other. For example, for the dbpedia_dnb and dbpedia_viaf tasks, initializing the active learning DeepMatcher model with transfer learning performs better overall than our unsupervised initialization method. However, this is not the case for the other tasks, such as abt_buy, for which the unsupervised initialization outperforms the transfer learning initialization. This is consistent and verifies the findings of our analysis of the symbolic baseline results presented in Section 7.3.3, showing that initializing active learning with transfer learning produces better results in comparison to unsupervised matching only if the tasks are highly related but not otherwise. Finally, a clear comparison of the results for certain tasks and iteration combinations is rather limited given the F1 score differences in combination with the results' deviations. For example, deepmatcher_boot seems to outperform deepmatcher_TL in the 50th iteration for the wdc_phones task. However, the former has a standard deviation of 0.045, which signifies that for some of the runs deepmatcher_boot may underperform deepmatcher_TL.

7.4 Discussion and Conclusion

In this chapter, we presented an unsupervised method for bootstrapping active learning for entity resolution. Our method relies on a thresholding heuristic that considers pre-calculated similarity scores and assigns labels with some degree of noise to the record pairs of the unlabeled pool. The noisy labels are used for initializing the active learning process and throughout the whole active learning cycle for the training of the learner and query selection.

The proposed method comes at no additional labeling cost in contrast to the common practice of using a manually labeled seeding set of matching and non-matching record pairs applied in many related works, discussed in Section 7.1. In comparison to initialization approaches that do not increase the labeling effort by randomly initializing the learner, which is typically a set of linkage rules evolving using genetic programming [Isele and Bizer, 2013; Ngomo and Lyko, 2012; Ngomo et al., 2013], our method uses a random forest classifier as a learner. Tree-based models, such as random forests, have been shown to perform better than rule-based models in active learning settings for entity resolution [Meduri et al.,

2020]. Finally, our method does not assume abundant labeled data from a similar entity resolution task, which is required for approaches applying transfer learning as a means of initializing active learning [Kasai et al., 2019].

We used six entity resolution benchmark tasks with different profiling characteristics to evaluate our proposed unsupervised thresholding heuristic, as well as the overall active learning method. With regard to the unsupervised thresholding heuristic, we showed that it achieves overall better results independently from the underlying similarity score distribution in comparison to static, Otsu's, and valley-emphasis thresholding. With regard to the overall active learning method, we showed that it outperforms baseline active learning methods, which use the HeALER query strategy [Chen et al., 2019] and random initialization. The improvement in the F1 score is more dominant in the cold start phase (up to 48% after 20 iterations with single queries), while within a labeling budget of 100 labeled records pairs our method outperforms by up to 3% depending on the task. Additionally, we compared our method to an active learning symbolic baseline that uses transfer learning for initialization. We showed that the latter can only achieve good results and outperform our method if the source and target tasks are highly related. For pairs of tasks that are not highly related, our method delivers better results in terms of F1 score by up to 4.2% after 100 active learning iterations with single queries.

Finally, we compared our method to the subsymbolic active learning method for entity resolution of Kasai et al. [2019], which applies transfer learning for initialization and uses the DeepMatcher model as learner [Mudgal et al., 2018]. Considering that the method of Kasai et al. [2019] relies on an external validation set and the implementation is not publicly available, we re-implemented and adapted the method so that the validation set is part of the labeled set and the augmented training data. Furthermore, we combined our unsupervised initialization method with the active learning method of Kasai et al. [2019] into an additional subsymbolic baseline. Our comparison results showed that the subsymbolic methods entail significantly larger waiting times among the active learning iterations, i.e. up to 16 minutes for a batch query of 10 record pairs. This can be prohibitive in an active learning setting, given that a human annotator is steadily involved in the learning loop. With respect to the learner's performance in terms of F1 score, we observed that the unsupervised bootstrapped subsymbolic baseline outperforms our method given a labeling budget of 1000 record pairs for one of the six evaluation tasks, entailing a large number of corner cases and the highest textuality among the others. In contrast, for all other tasks, our symbolic active learning method consistently outperforms the subsymbolic baselines for a labeling budget up to 1000 record pairs. Given a labeling budget of 500 record pairs our method outperforms both subsymbolic baselines for all tasks by up to 32% in F1 score. This confirms the finding of Meduri et al. [2020], indicating that neural network-based active learning methods underperform traditional classifiers, such as random forest and SVM models, in active learning settings with a limited labeling budget for entity resolution.

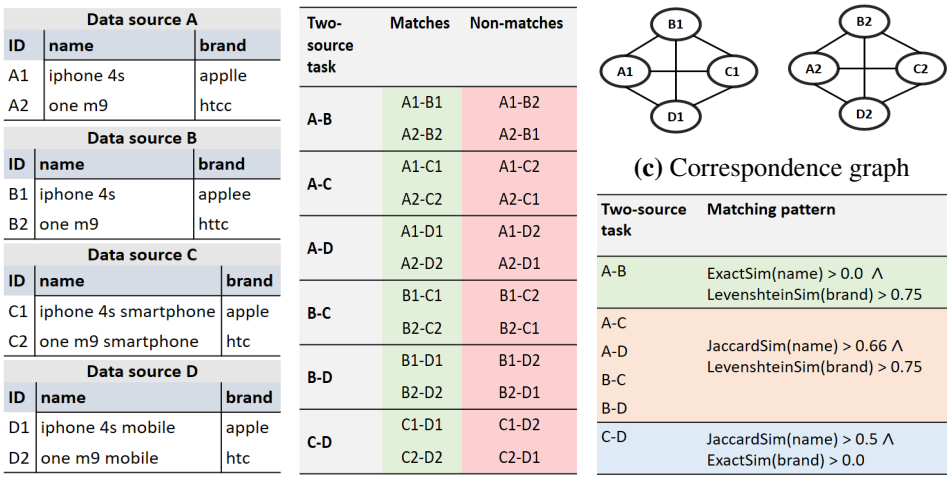
Chapter 8

Active Learning for Multi-Source Entity Resolution

Multi-source entity resolution is the task of identifying records that derive from more than two data sources and refer to the same real-world object [Christophides et al., 2020; Papadakis et al., 2021]. A multi-source entity resolution task can be viewed as a combination of multiple two-source tasks between pairs of data sources with potentially different underlying matching patterns. In the context of our work, we focus on multi-source tasks with records having the same schema while we set no assumption on the existence of duplicates in the same data sources. Within this context, the formal definition of the general entity resolution task, as stated in Section 2.2, is also applied for the multi-source entity resolution task.

Multi-source entity resolution tasks have inherent characteristics which are complementary to the ones of two-source tasks, described in Chapter 3. We hypothesize that these characteristics can be exploited to reduce the labeling effort in a pool-based active learning setting. Active learning has been barely applied for the task of matching records between multiple sources [Huang et al., 2018], while no previous work has explored the hypothesis that the performance of an active learning method can be improved if multi-source-related characteristics are taken into consideration.

We illustrate our hypothesis using the motivating example of Figure 8.1. The example multi-source task comprises four data sources with records describing mobile phones (Figure 8.1a). Combining pairwise the four data sources results in six two-source entity resolution tasks (Figure 8.1b). Given the overlap of records describing the same real-world object among the data sources, the multi-source task can be viewed as a correspondence graph with the edges denoting matching relations (Figure 8.1c). Exploiting graph signals, such as graph transitivity, can contribute to the discovery of potentially false learner predictions. For example, given the correspondence graph of Figure 8.1c and the learner predictions (A1-B1): match, (B1-C1): match, and (A1-C1): non-match, selecting the record pair (A1-C1) for labeling can lead to the discovery of a false negative prediction. Addi-



Data source A

ID	name	brand
A1	iphone 4s	applle
A2	one m9	htcc

Data source B

ID	name	brand
B1	iphone 4s	applee
B2	one m9	httc

Data source C

ID	name	brand
C1	iphone 4s smartphone	apple
C2	one m9 smartphone	htc

Data source D

ID	name	brand
D1	iphone 4s mobile	apple
D2	one m9 mobile	htc

(a) Data sources

Two-source task	Matches	Non-matches
A-B	A1-B1	A1-B2
	A2-B2	A2-B1
A-C	A1-C1	A1-C2
	A2-C2	A2-C1
A-D	A1-D1	A1-D2
	A2-D2	A2-D1
B-C	B1-C1	B1-C2
	B2-C2	B2-C1
B-D	B1-D1	B1-D2
	B2-D2	B2-D1
C-D	C1-D1	C1-D2
	C2-D2	C2-D1

(b) Two-source tasks of the multi-source task

(c) Correspondence graph

Two-source task	Matching pattern
A-B	ExactSim(name) > 0.0 ∧ LevenshteinSim(brand) > 0.75
A-C A-D B-C B-D	JaccardSim(name) > 0.66 ∧ LevenshteinSim(brand) > 0.75
C-D	JaccardSim(name) > 0.5 ∧ ExactSim(brand) > 0.0

(d) Groups of tasks

Figure 8.1: Example of a multi-source entity resolution task.

tionally, given the different value representations of the attributes used to describe the phone records, different groups of two-source tasks with similar matching patterns arise, as shown in Figure 8.1d and denoted with different colors. We consider a matching pattern as a disjunction of conjunctions of similarity-based features and threshold values. Exploiting the grouping signals during active learning can lead to the selection of more informative record pairs for labeling, by, for example, annotating only representative pairs from each group.

In this chapter, we turn our focus to active learning methods for reducing the labeling effort for multi-source entity resolution. We investigate how the graph and grouping signals that exist in multi-source entity resolution tasks, can be integrated within the active learning workflow for boosting the query selection and learner components, thus covering the contribution [C5] of the thesis. We develop *ALMSER*, an **A**ctive **L**earning algorithm for **M**ulti-**S**ource **E**ntity **R**esolution, which comes with two query strategies exploiting graph signals (ALMSERgraph) and grouping signals (ALMSERgroup). Additionally, ALMSER exploits graph signals to boost the learner's training with additional training data. We evaluate ALMSER on six multi-source entity resolution tasks and compare its results to baseline active learning methods using a common margin-based query strategy [Meduri et al., 2020; Mozafari et al., 2014] and the committee-based query strategy HeALER [Chen et al., 2019]. Our evaluation shows that graph and grouping signals lead to overall improved performance over the baseline methods when used for both query selection and the training of the learner.

The contributions of this chapter are summarized as follows:

- We are the first to tackle the problem of multi-source entity resolution with active learning by exploiting the inherent characteristics of the multi-source entity resolution tasks.
- We propose ALMSER, an active learning algorithm for multi-source entity resolution, which uses graph and grouping signals for query selection and training data enlargement.
- We evaluate ALMSER on five multi-source entity resolution tasks and show that it consistently outperforms baseline methods that do not use graph or grouping signals in terms of F1 score.

This chapter is structured into four sections. In Section 8.1, we present the methodological details of ALMSER. Section 8.2 presents the experimental setup and discusses the experimental results. Section 8.3 discusses related work on multi-source entity resolution. Finally, in Section 8.4, we discuss and summarize the main findings of this chapter.

The methodology as well as the evaluation of ALMSER presented in this chapter have been published in the Proceedings of the 20th International Semantic Web Conference [Primpeli and Bizer, 2021] and in the Proceedings of the 19th Extended Semantic Web Conference [Primpeli and Bizer, 2022]. The code and datasets used for the experimental evaluation of this chapter are publicly available.[1]

8.1 Methodology

In this section, we present our proposed active learning algorithm for multi-source entity resolution, which we abbreviate with ALMSER. We first summarize the overall process that is executed by ALMSER. Figure 8.2 gives an overview of the ALMSER workflow. The following subsections detail each step in the process.

We consider a pool-based active learning setting, as described in Section 6.2. We initialize ALMSER by bootstrapping the labeled set of record pairs in an unsupervised fashion, as described in the previous Chapter 7. After initialization, the following steps are executed: First, we train a random forest base learner using the current labeled set (1) and get base predictions for all unlabeled record pairs of the pool (2), which together with the labeled set are used to construct a correspondence graph (3). Next, we derive the clean components of the graph (4) and assign graph-inferred labels to the record pairs of the pool, which are part of the clean components (5). In the following step, the query strategy picks the most informative record pair for labeling (6). ALMSER comes in two variations with respect to the query strategy for assessing the informativeness of the pool record pairs: ALMSERgraph and ALMSERgroup. The ALMSERgraph query strategy considers the disagreement between the predicted labels of the base learner and

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

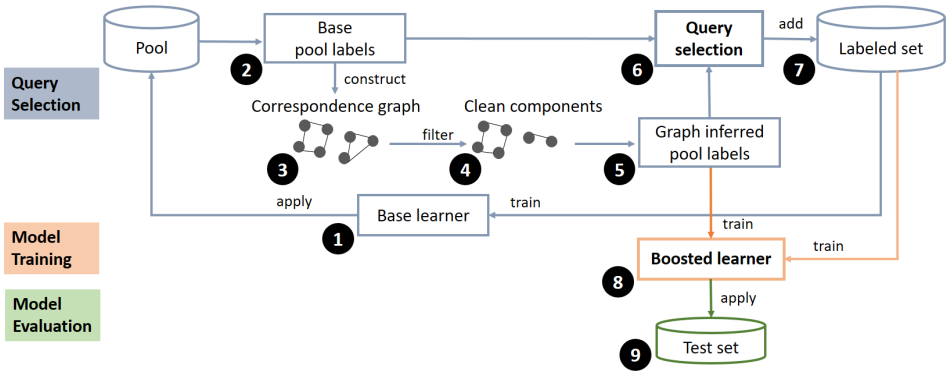


Figure 8.2: Overview of the ALMSER algorithm.

the graph-inferred labels. Additionally, the ALMSERgroup query strategy exploits grouping signals. The selected record pair is annotated as *match* or *non-match* and is added to the labeled set (7). We use the graph-inferred labels to derive additional training data, which together with the labeled set are used for training the random forest boosted learner (8). In order to evaluate how the performance of the boosted learner develops during the active learning process, we apply the boosted learner to an unseen test set after each iteration (9).

8.1.1 Initialization

The initialization of active learning is a non-trivial step, which has been shown to suffer from the cold start problem [Konyushkova et al., 2017] and was thoroughly discussed in Chapter 7. To circumvent the cold start problem, we use the unsupervised bootstrapping method proposed in Chapter 7 and initialize the labeled set. We apply the approach for each two-source task of the multi-source setting and select two record pairs per task: one with the highest and one with the lowest aggregated score. Considering that in the very early active learning iterations, the base model, which we use to construct the correspondence graph, is highly unstable, we perform the first 20 iterations using the committee-based query strategy HeALER [Chen et al., 2019]. Afterward, we switch to ALMSER.

8.1.2 Correspondence Graph Construction

After initializing the labeled set, the graph-boosted active learning with ALMSER starts. In each active learning iteration, we construct from scratch the correspondence graph of the multi-source task (steps 1-3 in Figure 8.2) with the aim of obtaining graph signals. The graph signals are used in later steps of our methodology for query selection and training of the learner. The correspondence graph contains

all distinct records of the record pairs in the pool and the labeled set as nodes. We add an edge to the graph for every confirmed matching record pair in the labeled set, while we add no edge for every labeled non-matching pair.

Additionally, we use the pool predictions of a random forest classifier, which we refer to as base learner, for inferring potential matching pairs. More concretely, in each iteration, the base learner is trained on all record pairs of the labeled set and applied to the record pairs of the pool. Each pool pair is assigned the predicted base label, *match* or *non-match* together with a confidence score, which is the predicted class probability of the base learner. We add an edge to the correspondence graph between the nodes of every pool record pair with a matching base label, while we add no edge if the base label is non-match.

Finally, we assign weights to the edges of the correspondence graph. Every edge that derives from the labeled set and is therefore confirmed to be true receives the weight of 100. The edges deriving from the base learner matching predictions are weighted according to their confidence score, which is in the range of 0.5 to 1.0. The significantly larger weight of 100 assigned to edges representing confirmed matching relations in comparison to the weight of edges representing predicted matching relations ensures that the labeled record pairs contribute more to the cleansing steps which follow the graph construction step. The value of 100 is empirically estimated for multi-source tasks comprising less than 10 sources that may contain duplicates. More formally:

Definition 8.1.1 (Correspondence Graph) *A correspondence graph is an undirected weighted graph $G = (R, E, w)$, with $E \subseteq \{\{r_i, r_j\} : r_i, r_j \in R\}$ being the set of edges and R being all the records appearing in all data sources S_D of the matching task. An edge $e\,\{r_i, r_j\} \in E$ exists $\iff$ $\{r_i, r_j\} \in \{L_{match} \cup P_{match}\}$, with L_{match} being the set of record pairs labeled as matching and P_{match} being the set of record pairs predicted as matching by the base learner. Every edge e is attributed a weight w as follows: $w = 100 \iff \{r_i, r_j\} \in L_{match}$ and $w = P(\{r_i, r_j\}\,|C = match)$, i.e. the prediction probability that the record pair is matching.*

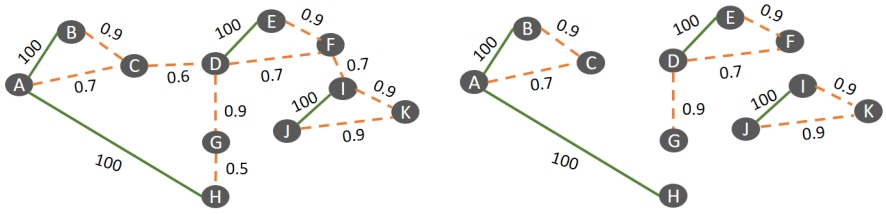
8.1.3 Correspondence Graph Cleansing

Exploiting the transitivity of the correspondence graph can lead to the discovery of false negative base learner predictions. For example, given three record pairs (A-B), (B-C), and (A-C) which have been predicted by the base learner as *match*, *match* and *non-match* respectively, we can infer using graph transitivity that (A-C) is also a matching pair and that it is likely a false negative prediction of the base learner.

However, given that the edges of the correspondence graph, deriving from the matching base learner predictions, are subject to some noise, a wrongly assigned edge can lead to a series of false positive record pairs. Therefore, we need to discover likely wrong edges and remove them from the correspondence graph. The

Labeled set	
A-B	match
D-E	match
D-H	non-match
I-J	match
A-H	match

(a) Example labeled set of current iteration



(b) Correspondence graph given labeled set and base-model predictions

(c) Correspondence graph after minimum cuts (D-C & G-H) and bridges (F-I) removal

Figure 8.3: Exploiting the graph to detect false positives - an example.

example in Figure 8.3 demonstrates this problem. Figure 8.3b shows an example graph with 11 nodes and weighted edges. The solid edges connect nodes of matching record pairs found in the labeled set of Figure 8.3a and are therefore assigned a weight of 100. The dotted edges represent the base labels and are assigned their corresponding confidence weights. The resulting graph is connected and forms one connected component, i.e. there is a path from any node to any other node in the graph, indicating that all nodes refer to the same real-world object. However, this cannot be the case as there is a confirmed non-matching pair (D-H) in the labeled set of Figure 8.3a. Therefore, the path between nodes D and H needs to be cut. Given the edge weights, we calculate the minimum cut of the graph. The edges which should be removed in order to cut the path between D and H are the following: (D-C) and (G-H), as their total edge weight is less than any other cut alternative. We can additionally observe that the edge (F-I) forms a bridge between the two components (E,D,F,G) and (I,J,K) and is a possible false positive. We consider an edge to be a bridge edge if it connects two nodes that have each a degree larger than two, i.e. they have three or more direct neighbors, and a clustering coefficient larger than 0, i.e. there exist at least one edge between the neighboring nodes. Figure 8.3c shows the graph after minimum cuts and bridges removal, which reveals three connected components.

We rely on these observations and remove edges between nodes of potentially false positive record pairs using a two-step procedure. First, we iterate over all non-matching pairs in the labeled set and for each pair, we check if there is a path between the two nodes-records in the graph. In case we find a path, we calculate the minimum cut of the graph considering the weights of the edges. For the calculation

of the minimum cuts, we use the *networkx* implementation[2], which is based on the max-flow min-cut theorem [Dantzig and Fulkerson, 1956]. As a second step, we identify bridge edges from the graph, as explained above, and remove them.

8.1.4 Clean Components Filtering

After cleansing the correspondence graph, we filter its clean components and assign graph-inferred labels to a subset of the pool record pairs (steps 4-5 in Figure 8.2) with the aim to get more accurate graph signals that can both identify wrong base learner predictions and lead to clean enlarged training data.

In order to derive the clean components of the correspondence graph, we first compute all connected components. Considering that smaller components are cleaner than larger ones, we assume a component to be clean if its size is equal to or smaller than the number of data sources to be matched, independently if all the nodes of the component derive from partially the same or different data sources. Although this heuristic comes naturally for deduplicated sources, we show during evaluation that it is also a good approximation for discovering the clean components of the graph in multi-source entity resolution tasks with non-deduplicated data sources.

We use the correspondence graph and the clean connected components to assign graph-inferred labels to a subset of the pool record pairs. To do so, we use the following simple heuristic: If the pair belongs to the same clean component, we assign a matching graph-inferred label. If there is no path in the correspondence graph between the two records of the pair, then we assign a non-match graph-inferred label. Finally, for record pairs belonging to non-clean components, no graph-inferred label is assigned. Although this heuristic results in the extraction of cleaner enlarged training data in comparison to the complete correspondence graph, which will be shown in our evaluation in Section 8.2.5, possible alternative directions for assigning graph-inferred labels could be considered, such as the overall weight of the edges of a connected component.

8.1.5 Query Selection

After filtering the clean components of the correspondence graph, the informativeness of the pool record pairs is assessed. To do so, two query strategies are implemented: ALMSERgraph and ALMSERgroup. ALMSERgraph exploits graph signals for selecting likely false positive and negative predicted pairs by the learner. ALMSERgroup uses, in addition to the graph signals, grouping signals that can be relevant to a multi-source entity resolution task considering the overlap of the matching patterns among the different two-source tasks. In the following, we describe the ALMSERgraph and ALMSERgroup query strategies.

[2]https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.\flow.minimum_cut.html

ALMSERgraph

The ALMSERgraph query selection strategy assigns binary informativeness scores to the record pairs of the clean components of the correspondence graph (Step 6 in Figure 8.2). Pairs without conflicts between the graph-inferred label and the base label are assigned a score of 0. Pairs whose graph-inferred label is different from the base label are assigned a score of 1 and are considered as most informative.

Margin-based and committee-based query strategies aim to select instances for which the learner or a committee of models produces non-confident predictions. On the contrary, the ALMSERgraph query strategy uses the clean components of the correspondence graph to pick instances that are most likely predicted wrong by the base learner. In that regard, ALMSERgraph can be categorized as a heuristic-based query strategy, as discussed in Section 6.3.1. These disagreements between the graph and the base learner hint toward matching patterns that are not covered yet by the base learner and can occur under two conditions: First, if the base learner has predicted the record pair to be a non-match and due to graph transitivity the graph-inferred label is match. Second, if the record pair has been predicted as match by the base learner, but the corresponding edge was found to be a bridge edge or was part of a minimum cut between confirmed non-matching pairs and therefore was removed during the cleansing step, as described in Section 8.1.3.

We illustrate the discovery of new matching patterns by graph transitivity with the example of Figure 8.4. The example presents three records from different data sources describing the same author (Figure 8.4a) and a subset of labeled pairs and base learner predictions (Figure 8.4b) which are used to construct the correspondence graph (Figure 8.4c). Given the matching pair (1a-2a) of the labeled set, the base learner might be trained to capture matching patterns based on the similarity of the *Lastname* and the *Works* attributes. However, it might wrongly predict the pair (1a-3a) as non-matching, as it has not yet learned the pattern that high similarity of *Birthdate* and *Firstname* also indicate a match. Based on graph transitivity, the pair (1a-3a) is assigned a matching graph-inferred label and receives an informativeness score of 1. Selecting this pair as a query candidate supports the model in learning the relevance of the *Birthdate* and *Firstname* attribute combination for matching.

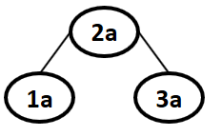
In order to ensure that the ALMSERgraph query strategy selects equally likely false positives, i.e. pairs with a non-match graph inferred label, and likely false negative pairs, i.e. pairs with a match graph-inferred label, we assign selection probability weights to all record pairs with an informativeness score of 1. For example, given 10 likely false negatives and 1 likely false positive, we assign the selection probability weights 0.1 for each false negative and 1.0 for the false positive pair. Finally, considering the selection probability scores, we perform weighted random selection over the candidate record pairs with an informativeness score of 1 and select one pair which is annotated and added to the labeled set (Step 7 in Figure 8.2). Alternative schemes for weighting the likely false predicted record pairs that could be considered but are not evaluated in our work could be based on

Source - ID	Firstname	Lastname	Birthdate	Works
1-a	Kiki	Dimoula	06.06.1931	In absentia
2-a		Dimoula		In absentia
3-a	Kiki		06.06.1931	In absentia

(a)

Labeled set (subset)	
(1a-2a)	match
Base learner predictions (subset)	
(1a-3a)	non-match
(2a-3a)	match

(b)



(c)

Figure 8.4: Discovery of matching patterns with ALMSERgraph - an example.

the prediction probability assigned by the base learner. In that case, the assigned weights would influence the query strategy to select either confident (prediction probability close to 1.0) or unconfident (prediction probability close to 0.5) pool record pairs. However, the confidence scores of a random forest classifier can be misleading mostly in the early active learning iterations due to overfitting, thus leading to query selection bias and low learner performance. Additionally, weighting schemes based on the overall edge weight or the vertex connectivity of the connected components from which the likely false predicted pool record pairs derive, could be further researched.

ALMSERgroup

A multi-source matching task can contain groups of two-source matching tasks sharing the same underlying matching patterns, as already motivated in the introduction of this chapter. We hypothesize that exploiting such grouping information can direct the active learning strategy to select record pairs covering all underlying matching patterns of the complete multi-source task with a smaller amount of annotations. We illustrate our hypothesis with the example of Fig. 8.1. The pairwise combinations of the four data sources result in six matching tasks, which given the underlying matching patterns, can be grouped into three groups, as shown in Fig. 8.1d. In such a setting, the active learning query strategy should distribute the queries for labeling over the tasks A-B, C-D and any of the {A-C, A-D, B-C, B-D}, as the latter have all the same underlying matching pattern. However, none of the existing active learning query strategies for entity matching exploits such grouping information.

In order to investigate whether the labeling effort can be further reduced by

exploiting such grouping signals, we develop ALMSERgroup, a variation of the ALMSERgraph query strategy. ALMSERgroup filters the pool to only include record pairs belonging to matching tasks that are representative of a cluster of similar matching tasks. We explain below how representative tasks are selected. In this way, ALMSERgroup avoids picking record pairs for annotation from similar tasks. During active learning, the ALMSERgraph query strategy is applied using the reduced pool. In the case of no disagreements between the learner predictions and the graph-inferred labels among the record pairs of the reduced pool, HeALER is used as a fallback query strategy.

In order to identify two-source tasks with similar matching patterns in an unsupervised way, we first compute the task relatedness (TR) between all pairs of two-source tasks [Thirumuruganathan et al., 2018]. TR calculates how similar two tasks are by training a logistic regression classifier to predict the task from which each record pair originates. We measure the prediction quality of the classifier using the Matthews correlation coefficient (MCC) and calculate the TR score as $1 - MCC$ [Thirumuruganathan et al., 2018]. Algorithm 2 Part A shows the pseudocode for calculating the relatedness scores of all pairwise combinations of two-source tasks T of a multi-source entity resolution task.

Given the TR scores of each pairwise combination of two-source tasks, we cluster them such that the overall mean TR score of all clusters is maximized. We determine the optimal number of clusters by penalizing the overall mean TR score with a penalty factor α multiplied by the number of clusters. In this way, we prefer smaller amounts of clusters over larger ones which results in a smaller pool of representative record pairs for the query strategy to choose from. Finally, we identify the most representative two-source tasks of each cluster, considering their TR to all other tasks of the same cluster, and select only the record pairs of the representative tasks for initializing the unlabeled pool. Algorithm 2 Part B shows the pseudocode for finding the most representative tasks T_{submax}, which maximize the overall task relatedness.

8.1.6 Boosted Learner Training

Training on small amounts of data tends to cause the classification models to overfit the training data and not be able to generalize well on unseen data [Anthony and Biggs, 1997]. To circumvent the overfitting of the learner on the small labeled set acquired during active learning, we enlarge the set used for training the learner, which therefore we call *boosted learner*.

Similar to the base learner, we use a random forest classifier as the boosted learner, assuming that a random forest model with a large number of estimators can expand to fit the matching patterns of all tasks in a multi-source entity resolution setting. The enlarged training set contains, in addition to all pairs of the labeled set, the record pairs of the clean connected components of the current iteration whose labels are not manually verified but inferred from the correspondence graph.

In a real-world active learning setting, we would train the boosted learner at

Algorithm 2 ALMSERgroup: selecting a subset of two-source entity resolution tasks to query from.

Input: T feature vectors without labels of each two-source task,
α subset size penalty,
R = $[|T|]\,[|T|] \leftarrow 0$ init. relatedness matrix,
T_{sub} all subsequences of T
Output: subset of two-source tasks $T_{submax} \sqsubseteq T$ that maximizes the mean relatedness

```
// Part A: calculate task relatedness
```
for i in T **do**
  for j in T **do**
    if $T_i \neq T_j$ **then**
      $X \leftarrow concat(T_i, T_j)$
      $y \leftarrow ["T_i" \times |T_i|, "T_j" \times |T_j|]$
      $m \leftarrow LogReg()$
      $R_{ij}, R_{ji} \leftarrow MCC(m, X, y)$
    end if
  end for
end for
```
// Part B: find task subsequence with max. overall
relatedness
```
for t_{sub} in T_{sub} **do**
  $R_{sub} \leftarrow R_{i \in t_{sub}, j \in T}$
  $t_{subRLTD} \leftarrow mean(R_{sub}) - a \times |t_{sub}|$
end for
$T_{submax} \leftarrow t_{sub}$ of $\max(t_{subRLTD})$
return T_{submax}

the last active learning iteration, as it does not affect the query selection, i.e. the query strategies of ALMSER are agnostic towards the boosted learner. However, so that we can evaluate the boosted learner along each active learning iteration, we train and apply it to the test set as a final step of each iteration (Steps 8 and 9 in Figure 8.2).

8.2 Experimental Evaluation

We evaluate ALMSER with its two variant query strategies, ALMSERgraph and ALMSERgroup, using five multi-source entity resolution tasks having different profiling characteristics. In this section, we first present the evaluation tasks and the experimental setup. Afterward, we compare ALMSER to two baseline active

Table 8.1: Multi-source entity resolution tasks used for evaluating ALMSER.

Multi-source task	# Data sources	# Pairs (in K)		Range of		
		matches	non-matches	Schema complexity	Sparsity	Corner cases
MusicBrainz	5	16.1	369.7	[3-5]	[0.05-0.12]	[0.08-0.42]
MusicBrainz_mut	5	16.1	369.7	[3-6]	[0.05-0.23]	[0.06-0.62]
computers	4	4.8	69.6	[3-4]	[0-0.05]	[0.02-0.30]
computers_mut	4	4.8	69.6	[3-6]	[0-0.18]	[0.24-0.50]
restaurants	4	11.2	56.5	[4-7]	[0-0.08]	[0.05-0.19]

learning methods that do not use graph and grouping signals. Additionally, we perform an ablation study by evaluating the distinct components of ALMSER that exploit graph or grouping signals separately. Finally, we evaluate the cleanliness of the enlarged training data used for boosting the learner.

8.2.1 Multi-Source Entity Resolution Tasks

We use five multi-source entity resolution tasks for our experimental evaluation. The tasks cover the domains of music, products, and restaurants. Table 8.1 contains profiling information about the five tasks, including the number of sources to be matched, as well as the amount of matching and non-matching pairs per task. The last three columns in Table 8.1 show the value ranges of the profiling dimensions schema complexity, sparsity, and corner cases for the two-source entity resolution tasks that make up each multi-source task. Details on the profiling dimensions and their calculation were discussed in Chapter 3, Section 3.4.2.

MusicBrainz Task The MusicBrainz multi-source task has been published by the Database Group of the University of Leipzig.[3] The task contains song records from the MusicBrainz database.[4] Each data source is a modified version of the original data source and therefore the two-source tasks that make up the multi-source task have different underlying patterns. In five of the ten two-source tasks the attributes *album*, *length*, and *title* are most relevant for matching, while for the rest of the tasks different attributes reveal the underlying matching patterns such as *title* and *song number* or *title* and *artist*.

MusicBrainz_mut Task Additionally to the original MusicBrainz multi-source task, we curate a modified version of it, abbreviated with *MusicBrainz_mut*. For the curation of the modified version, we remove 30% of the values of the attribute *album* in two of the five data sources. Additionally, we remove 10% of the values of the attribute *title* in one of the five data sources. Finally, we add noise in the

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

[4]https://musicbrainz.org/doc/MusicBrainz_Database

artist values of 30% of the records in one of the five data sources. The noise is added in the form of pre-defined string values, e.g. "the band of" or "an album by", thus decreasing the token similarity of values referring to the same artist. The modifications lead to an increased schema complexity, sparsity, and amount of corner cases in comparison to the original MusicBrainz task, as shown in Table 8.1.

Computers Task We use the English version of the WDC Product Corpus for Entity Resolution, presented in Chapter 5, and derive a subset of computer product records published on four e-commerce websites. For generating the computers task, we first filter the offers of the English Corpus that contain specification tables. Next, we use the apriori algorithm [Agrawal and Srikant, 1994] for finding a subset of websites that frequently appear in the same clusters, i.e. they annotate offers describing the same real-world products. We identify the following four websites having the largest overlap of offers describing the same real-world product: *harddrivesdirect.com*, *sillworks.com*, *tweakers.net*, and *usapartsdirect.com*. We manually align the schema across the offers of the four websites and derive the following 11 attributes: *title, description, brand, part number, sub-category, category, generation, capacity, spindle speed, manufacturer*, and *price*. It is worth noting that the *part number* attribute is not enough for matching the records across all data sources, as its density is 90%. Thus, additional attributes are required for matching, such as title and description, which results in a schema complexity larger than one, as shown in Table 8.1.

Computers_mut Task Similarly to the MusicBrainz task, we curate a modified version of the computers task, which we abbreviate with *computers_mut*. We modify the computers task by removing 30% of the *part number* attribute values from one data source. This leads to an increased schema complexity, sparsity, and amount of corner cases in comparison to the computers task, as indicated in Table 8.1. While the underlying matching patterns of the original task focus mostly on the combination of the *title*, *part number*, and *description* attributes, the mutated task requires additional attributes, such as *category*, *capacity*, and *generation*.

Restaurants Task The restaurant-related multi-source task derives from the Magellan repository,[5] which provides a large number of two-source entity resolution tasks. We retrieve four of the restaurant data sources that have been crawled from large restaurant aggregators and use the phone number as weak supervision in order to establish the complete mappings between all data source pairs. While three of the six two-source entity resolution tasks have a low containment of corner cases (<10%) and can be solved only with address-related attributes, the rest of the two-source tasks require additional attributes such as *name*, *cuisine*, and *website*.

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

Record Pair Comparison We turn the records of all tasks into feature vectors by calculating data type-specific similarity scores, as described in Section 2.3.3. For string attributes, the following similarity scores are calculated: Levenshtein, Jaccard, extended Jaccard with inner Levenshtein, exact similarity, and Jaccard containment. For numeric attributes, the numeric similarity is calculated, as described in Section 2.3.3. In the case that a similarity score cannot be computed for an attribute combination because of missing values, we assign the out-of-range score -1. This allows any classifier to consider all record pairs without dropping or replacing the missing values.

Task Scenarios The selected multi-source tasks cover two distinct scenarios. The first scenario includes entity resolution tasks of duplicate-free data sources. Therefore their correspondence graph forms connected components of maximum size equal to the total amount of sources, which is the case for the MusicBrainz and MusicBrainz_mut tasks. The second scenario covers tasks of non-deduplicated data sources. This results in components that are larger than the total amount of sources, which happens for the computers, computers_mut, and restaurants tasks. We cover both scenarios in order to evaluate if ALMSER is robust towards tasks including non-deduplicated sources, considering that the training data enlargement step relies on the clean connected components of the correspondence graph. We remind the reader that we heuristically defined as clean the components of the graph whose size is equal to or smaller than the number of data sources to be matched, as described in Section 8.1.4.

8.2.2 Experimental Setup

We split the multi-source tasks into two subsets: one for initializing the pool that is available for query selection and one for testing. In order to ensure that there is no leakage by graph transitivity from the pool set to the test set, we split the record pairs to pool pairs and test pairs based on the connected components of the complete correspondence graph with a ratio 70%:30%.

We execute three runs for each active learning experiment and allow 200 iterations for each run. In each iteration, we allow single queries, i.e. 200 record pairs have been labeled in total by the end of each experimental run. All classification models used as part of ALMSER or the baselines are parametrized with the default parameters of the python scikit-learn library, version 0.24.2. Given that we are interested in solving the multi-source task globally, we report the F1 score achieved by the learner in each iteration over the complete test set and do not average the F1 scores achieved for each two-source task separately. Finally, we report the upper learning bound of passive learning for which all record pairs of the pool together with their respective labels are used for model learning. All experiments were run on a Linux server with Intel Xeon 2.4 GHz processors.

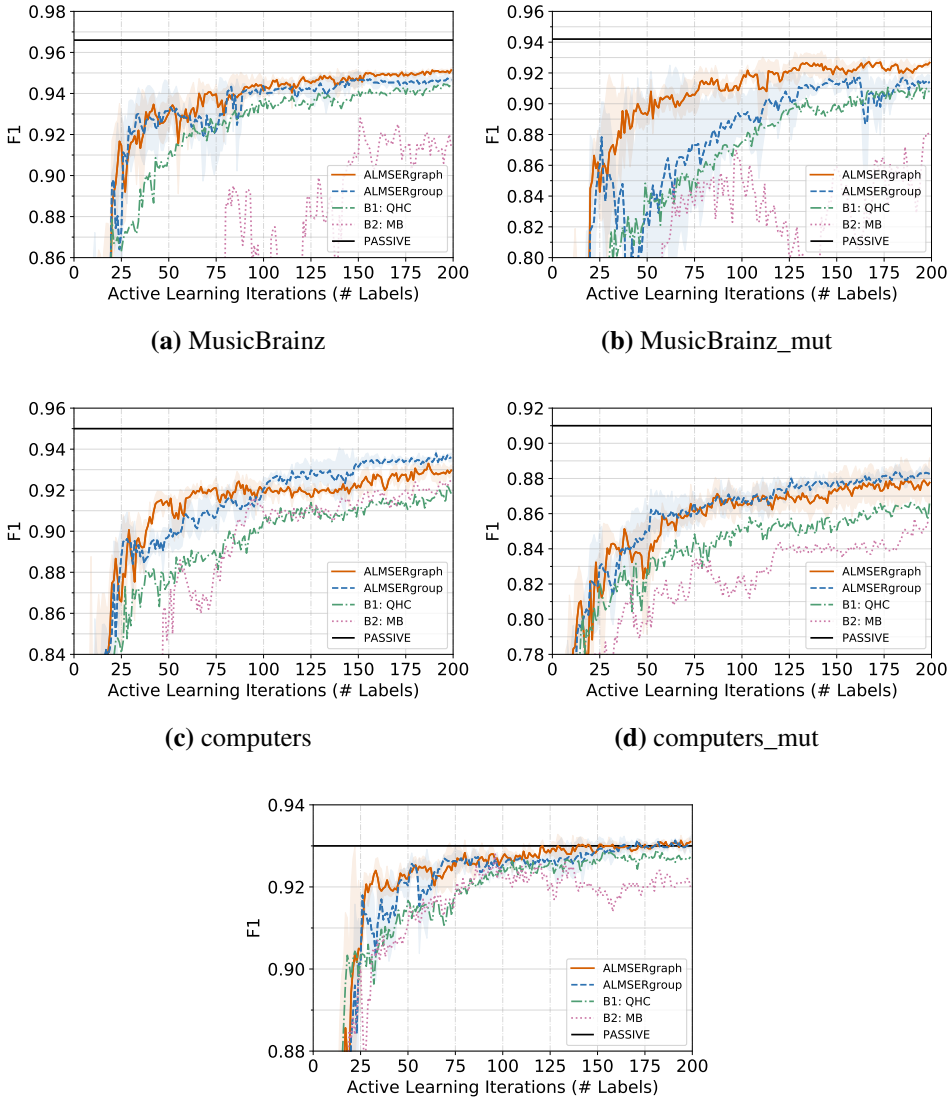
8.2.3 Comparison to Baselines

Baseline Methods We compare the two variants of ALMSER, i.e. ALMSERgraph and ALMSERgroup, to two baseline active learning methods with different classification-based query strategies and no graph signals. The first baseline method, abbreviated with QHC, uses the committee-based query strategy of the HeALER algorithm [Chen et al., 2019], which measures the informativeness of each candidate record pair as the disagreement of the predictions of a committee of heterogeneous classification models. The second baseline method, abbreviated with MB, uses a margin-based query strategy [Meduri et al., 2020; Mozafari et al., 2014]. The MB baseline is a learner-agnostic method, i.e. the classification model used as part of the query strategy is different from the learner. It selects the query candidates with a minimum distance to the decision hyperplane defined by an SVM classifier. Similar to ALMSERgraph and ALMSERgroup, a random forest classifier is used as a learner for the baseline methods. However, the learners of the baseline methods do not use graph signals and therefore are trained only on the labeled set. In order to ensure a comparable start of the learning process for all methods, we apply the same initialization step, described in Section 8.1.1. In the rest of this section, we first present the overall experimental results and compare the active learning methods across all iterations and passive learning. Afterward, we present detailed results of three active learning snapshots.

Comparison Overview of Active Learning Methods Figure 8.5 shows the average F1 score curves of ALMSERgraph, ALMSERgroup, and the two baseline methods for each multi-source entity resolution task per iteration. We can observe that as the active learning process unfolds, ALMSERgraph and ALMSERgroup outperform both baselines for all tasks. The MB baseline underperforms the QHC baseline for all tasks, while it fails to converge after 200 iterations for both the MusicBrainz and the MusicBrainz_mut tasks. The performance of ALMSERgraph and ALMSERgroup is comparable in three of the five tasks. ALMSERgraph significantly outperforms ALMSERgroup for the MusicBrainz_mut task. Additionally, ALMSERgroup is highly unstable considering the standard deviation of the F1 scores until the 100th iteration.[6] This indicates that exploiting grouping signals in addition to the graph signals is not helpful for the MusicBrainz_mut task.

Comparison to Passive Learning When 200 record pairs have been annotated, the F1 scores achieved by ALMSERgraph for all tasks are 0 to 3.2 percentage points lower than the passive learning results. The difference of the ALMSERgroup results to passive learning is 0.4 to 2.8 percentage points, while for the QHC baseline, it is 0.4 to 5.0 percentage points. The passive learning results are calculated by training a random forest classifier with all pairs from the pool as training

[6] We show the standard deviation of the F1 scores per iteration with the light-colored area around the plotted curves. To ease readability, we depict the standard deviation only for the ALMSERgraph and ALMSERgroup methods.



(a) MusicBrainz

(b) MusicBrainz_mut

(c) computers

(d) computers_mut

(e) restaurants

Figure 8.5: Comparison of ALMSERgraph and ALMSERgroup to active learning baselines and passive learning.

Table 8.2: Comparison of passive learning results with single and multiple (one per two-source task) random forest models.

Task	F1 - Single RF	F1 - Multiple RFs
MusicBrainz	0.966	0.960
MusicBrainz_mut	0.942	0.944
computers	0.950	0.944
computers_mut	0.910	0.903
restaurants	0.930	0.933

data and are indicated with the black horizontal lines in the plots of Figure 8.5. Finally, we want to verify that a single random forest model is enough to capture the matching patterns of all two-source tasks of the multi-source setting. To do so, we compare the passive learning F1 score, achieved by a single random forest trained on all available training data, to the F1 score achieved by task-specific models trained on the training data of each two-source task. Table 8.2 presents the comparison of the passive learning results. We observe that the F1 scores achieved with a single random forest model are similar to the ones achieved with multiple models. This confirms our hypothesis that a single random forest model can expand enough to fit all matching patterns of the multi-source setting.

Detailed Comparison of Active Learning Methods In the upper part of each iteration line in Table 8.3, we compare the F1 scores of the baseline methods MB and QHC to ALMSERgraph and ALMSERgroup at three snapshots of the active learning process. We observe that ALMSERgraph achieves a quicker gain in F1 score in the earlier iterations of the active learning process. ALMSERgraph outperforms the QHC and MB baselines by up to 5.5 and 13.4 percentage points, respectively, at the 75th iteration. At the same iteration, ALMSERgroup performs similarly to QHC for the MusicBrainz task, while for the rest of the tasks it outperforms QHC by up to 2.3 and MB by up to 11.6 percentage points.

Although ALMSERgraph and ALMSERgroup outperform the two baselines that do not use graph signals, even in the 200th active learning iteration, the gain in F1 is reduced. ALMSERgraph has improved performance by up to 1.9 percentage points in comparison to QHC and up to 4.8 percentage points in comparison to MB. For ALMSERgroup, the difference is smaller with up to 1.8 percentage points in comparison to QHC and up to 3.5 percentage points in comparison to MB.

With respect to the latency, ALMSERgraph and ALMSERgroup require more time for query selection in comparison to the QHC and MB baselines. For example, one baseline iteration for the computers task without graph signals takes 2.9-3.15 seconds, while one ALMSER iteration takes 14.58-15.10 seconds. This is due to the correspondence graph construction and cleansing steps, which are required for the ALMSERgraph and ALMSERgroup methods but not for the QHC and MB baselines that do not use graph signals.

Table 8.3: Comparison to baselines with no or partial graph signals.

Iteration	AL method	MusicBrainz	MusicBrainz_mut	computers	computers_mut	restaurants
F1 @ 75th	MB	0.805	0.836	0.881	0.833	0.915
	QHC	0.921	0.851	0.891	0.841	0.917
	ALMSERgraph	**0.939**	**0.906**	**0.921**	0.862	**0.926**
	ALMSERgroup	0.921	0.874	0.906	**0.864**	**0.926**
	MB_boost_learner	0.877	0.833	0.889	0.827	0.914
	QHC_boost_learner	0.891	0.891	0.894	0.843	0.924
	ALMSERgraph_qs	0.912	0.841	0.919	0.842	0.916
	ALMSERgroup_qs	0.887	0.822	0.898	0.859	0.917
F1 @ 125th	MB	0.890	0.817	0.912	0.840	0.919
	QHC	0.932	0.893	0.909	0.854	0.925
	ALMSERgraph	**0.946**	**0.920**	0.918	0.873	0.929
	ALMSERgroup	0.943	0.905	0.926	**0.876**	0.929
	MB_boost_learner	0.866	0.856	0.910	0.846	0.917
	QHC_boost_learner	0.914	0.914	0.901	0.865	**0.930**
	ALMSERgraph_qs	0.924	0.884	**0.927**	0.868	0.923
	ALMSERgroup_qs	0.913	0.865	0.924	0.872	0.919
F1 @ 200th	MB	0.914	0.879	0.925	0.854	0.920
	QHC	0.945	0.908	0.918	0.866	0.927
	ALMSERgraph	**0.951**	**0.927**	0.930	0.878	0.931
	ALMSERgroup	0.947	0.914	0.936	0.883	0.931
	MB_boost_learner	0.903	0.872	0.922	0.859	0.924
	QHC_boost_learner	0.926	0.926	0.916	0.871	**0.932**
	ALMSERgraph_qs	0.934	0.896	**0.938**	**0.884**	0.926
	ALMSERgroup_qs	0.918	0.888	0.932	0.883	0.921
F1-AUC 50th-200th	MB	128.81	123.84	135.68	124.82	138.02
	QHC	140.07	132.43	135.62	127.66	138.51
	ALMSERgraph	**141.57**	**137.51**	**139.19**	130.13	**139.18**
	ALMSERgroup	140.99	133.92	138.48	**130.78**	139.03
	MB_boost_learner	138.28	131.74	136.56	127.21	138.52
	QHC_boost_learner	136.39	136.39	134.93	128.82	138.35
	ALMSERgraph_qs	138.37	131.38	138.96	128.95	138.21
	ALMSERgroup_qs	137.02	127.19	137.64	130.19	137.48

8.2.4 Ablation Study

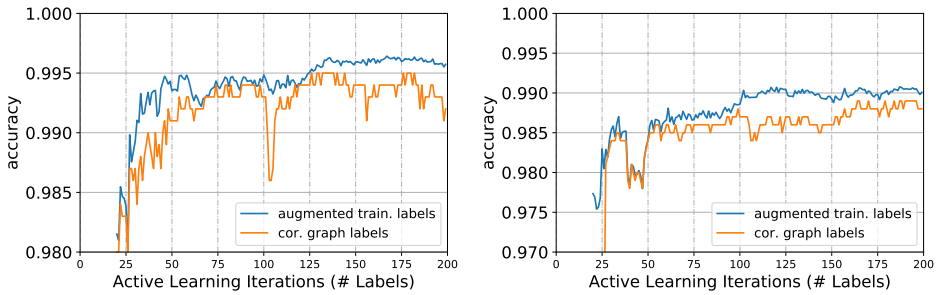
In this part of our experimental analysis, we evaluate the components of ALMSER, which are boosted using graph and grouping signals separately. To do so, we construct the following four setups that use partial graph and grouping signals and compare them to the methods employing full (ALMSERgraph, ALMSERgroup) or no (MB, QHC) graph and grouping signals:

1. ALMSERgraph_qs, a variation of ALMSERgraph which utilizes the graph signals only as part of the query strategy but not for boosting the learner with additional training data.
2. ALMSERgroup_qs, a variation of ALMSERgroup which uses graph and grouping signals only as part of the query strategy but not for boosting the learner with additional training data.
3. QHC_boost_learner, which applies the QHC query strategy for selecting candidates and uses graph signals for enlarging the training data and boosting the random forest learner.
4. MB_boost_learner, which applies the MB query strategy and uses graph signals for enlarging the training data and boosting the random forest learner.

We present the F1 scores in three snapshots of the active learning process for all methods that use partial graph and/or grouping signals in Table 8.3. Additionally, we report the area under the F1 curve (F1-AUC) from iteration 50 to iteration 200 for all methods.

Comparing the methods that do not use any training data enlargement technique, i.e. MB, QHC, ALMSERgraph_qs, and ALMSERgroup_qs, we can see there is no clear winner among the other query selection strategies. For example, QHC performs best for the MusicBrainz and MusicBrainz_mut tasks with respect to the F1-AUC scores, while ALMSERgraph_qs and ALMSERgroup_qs perform best for the computers and computers_mut tasks, respectively. Finally, for the restaurants tasks all of these four methods perform comparably as the differences of the F1-AUC scores are rather small. This indicates that the positive contribution of graph and grouping signals to the active learning query strategy is subject to the task at hand. We will investigate the impact of the profiling characteristics of the multi-source tasks on the performance of the different query selection strategies in Chapter 9.

We observe that the best performing methods for all tasks and snapshots use partial, e.g. ALMSERgraph_qs at iteration 125 for the computers, or full graph and grouping signals, e.g. ALMSERgroup at iteration 75 for the computers_mut task. However, in terms of F1-AUC, ALMSERgraph achieves overall better results between iterations 50 and 200 in all tasks except computers_mut. For the latter, ALMSERgroup outperforms in terms of F1-AUC.



(a) MusicBrainz_mut **(b)** restaurants

Figure 8.6: Comparison of the accuracy of the augmented training data and the labels derived from the complete correspondence graph.

8.2.5 Training Data Enlargement Evaluation

In the last part of our experimental evaluation, we report the size and correctness of the augmented training set, which results from the clean components of the correspondence graph, as explained in Section 8.1.4. Figure 8.6 presents the accuracy of the augmented training set in comparison to the accuracy of the complete correspondence graph for the MusicBrainz_mut and the restaurants tasks in each active learning iteration. We observe that our heuristic for filtering clean components extracts a cleaner part of the correspondence graph, as the accuracy of the augmented training set (blue line) exceeds the one of the complete graph (orange line) in each iteration. This occurs for both task scenarios, i.e. multi-source tasks with duplicate-free (MusicBrainz_mut) and non duplicate-free (restaurants) data sources.

Exploiting the record pairs from the clean components for training the boosted learner results in large amounts of additional training pairs. However, only the subset of record pairs in the augmented training set with a graph-inferred label different from the base label can give additional matching information to the boosted learner. Table 8.4 shows the size of the augmented training set, the number of record pairs in the training set with a disagreement between the graph-inferred and the base labels, as well as the ratio of correct graph-inferred labels to all record pairs with a disagreement in three active learning snapshots. Although the size of the augmented training set is much larger in comparison to the clean labeled data, there is only a relatively small amount of disagreements between the predictions of the base learner trained on the labeled set and the graph-inferred labels. Considering that for the majority of those disagreements, the graph-inferred label is correct (78.8-96.7%, as shown in Table 8.4), we conclude that the additional matching information that the boosted learner receives from the graph is subject only to a small amount of noise. Thus, it successfully supports the discovery of additional matching patterns which are not covered yet by the record pairs in the labeled set.

Table 8.4: Augmented training set in three AL snapshots.

Iteration	Musicbrainz_mut			restaurants		
	#Train pairs (K)	#Disagr.	% Correct graph	#Train pairs (K)	#Disagr.	% Correct graph
75th	256.5	1,476	0.966	42.8	73	0.821
125th	256.4	1,506	0.967	42.5	52	0.788
200th	254.7	1,009	0.874	42.6	27	0.814

8.3 Related Work

Multi-source entity resolution has been previously studied in related work. It is highly related to the task of knowledge base synthesis, which focuses on the "ingestion, disambiguation and enrichment of entities from a variety of sources" [Bellare et al., 2013]. There exist two main lines of research on multi-source entity resolution, which either focus on solving the scalability issues of integrating data from multiple sources [Hertling and Paulheim, 2021; Shen et al., 2007] or on developing supervised and unsupervised techniques for deriving models that can generalize well on all tasks of the multi-source setting [Jin et al., 2021; Saeedi et al., 2017, 2018].

In the work of Hertling and Paulheim [2021] on knowledge graph matching, different matching execution plans are evaluated in terms of accuracy and latency. Each execution plan follows a different matching order, defined by knowledge graph-related measures, such as the number of classes, instances, and statements. Similarly, the supervised SOCCER framework proposed by Shen et al. [2007] defines an efficient order of executing pairwise entity resolution tasks. The basic hypothesis of SOCCER is that different data sources come with different degrees of semantic ambiguity. This leads to domain-specific matchers not being able to fit all matching patterns among the different data source pairs. As matchers, the authors consider rules defined by an expert. These works confirm that defining a matching execution plan and employing one matching model for each pair of data sources is relevant for large-scale multi-source settings with more than 50 data sources. However, in Section 8.2.3, we showed that for multi-source entity resolution tasks with up to five data sources, training a single random forest model on all record pairs after blocking delivers similar results to multiple models trained on single two-source tasks.

Saeedi et al. [2017] compare different unsupervised clustering-based methods for multi-source entity resolution and propose CLIP [Saeedi et al., 2018]. Although CLIP is an unsupervised approach, it requires an expert to define domain-specific matching rules. The rules are then used to construct a correspondence graph. Additionally, the clustering approach of CLIP assumes duplicate-free sources, an assumption that is not necessary for our proposed method.

Jin et al. [2021] approach multi-source entity resolution with transfer learning. The authors consider a setting in which labeled and unlabeled data from new sources come incrementally and need to be integrated with record pairs that have already been matched. In this context, they study the following three challenges that can be related to the new unseen data: (i) missing attribute values, (ii) new attributes, and (iii) different value distributions. Their proposed system, AdaMEL addresses those challenges by leveraging existing labeled data as well as large amounts of unlabeled data in a transfer learning scenario with domain adaptation. In comparison to our active learning-based approach that relies on carefully selected record pairs (up to 200), AdaMEL assumes that the record pairs of half of the data sources in the multi-source setting are labeled. JointBERT [Peeters and Bizer, 2021] applies deep learning techniques for multi-source entity resolution and treats the matching problem in parallel as a binary and multi-class classification task. Similar to the work of Jin et al. [2021], JointBERT is applied only in a passive learning setting and therefore requires large amounts of training data (2K-70K record pairs).

Active learning has been barely applied for tackling the task of multi-source entity resolution. To the best of our knowledge, Huang et al. [2018] are the first to propose an active learning-based method for multi-source entity resolution. However, differently from our active learning setting, their work considers error-prone human annotators. The main focus is on improving the interaction with the human annotator by querying an optimal comparative table of records originating from multiple sources. The evaluation of the authors relies on user satisfaction studies and indicates that presenting the queries as comparative tables rather than single queries can increase the accuracy of the annotators' answers.

8.4 Discussion and Conclusion

In this chapter, we presented ALMSER, an active learning method for multi-source entity resolution. ALMSER uses graph and grouping signals which are inherent to multi-source entity resolution tasks for query selection and training of the learner. ALMSER comes with the query strategies ALMSERgraph and ALMSERgroup, which, to the best of our knowledge, are the first query strategies specially tailored to multi-source entity resolution tasks.

Compared to CLIP, an unsupervised clustering method for multi-source entity resolution within the FAMER framework [Saeedi et al., 2018, 2020], ALMSER does not assume deduplicated data sources. Additionally, CLIP relies on linkage rules, i.e. combinations of similarity functions and attributes, for calculating similarity links among the clustered records. Considering that the linkage rules need to be domain-specific, an expert is required to define them. On the other hand, ALMSER uses record pairs labeled by a human annotator during active learning for training a classification model.

We evaluated ALMSER with its two query strategies on five multi-source en-

tity resolution tasks containing up to five data sources. The relatively small amount of data sources contained in the tasks used for evaluation makes it possible to use the subset of the Cartesian product selected after blocking during active learning within an acceptable runtime (up to 15 seconds per iteration). In that regard, scalability issues that arise in larger multi-source entity resolution settings are not addressed in the context of our work. Such issues have been addressed in related work by defining optimal matching execution plans [Hertling and Paulheim, 2021; Shen et al., 2007] or with the parallel design and execution of the entity resolution method [Saeedi et al., 2017].

We identified that ALMSER reaches close to passive learning results achieved with a symbolic baseline method for all tasks after 200 iterations. More concretely, ALMSERgraph reaches 0 to 3.2 and ALMSERgroup reaches 0.4 to 2.8 percentage points in F1 score lower than passive learning after 200 iterations with single queries. This implies a significantly smaller labeling effort in comparison to passive learning, which requires 57K to 386K labeled record pairs considering the tasks used in our evaluation.

Additionally, we compared ALMSER to two baseline active learning methods using a committee-based and a margin-based query strategy and no graph or grouping signals. The comparison results indicated that ALMSER consistently outperforms the two baselines in all tasks. The F1 score difference is larger in the earlier iterations than the later ones, with ALMSERgraph outperforming the committee-based baseline by up to 5.5 percentage points in F1 score in the 75th iteration and up to 1.9 percentage points in F1 score in the 200th iteration.

Finally, we performed an ablation study to evaluate the distinct components of ALMSER that exploit graph and grouping signals. We identified that when the signals are used for both query selection and learner training, overall better performance is achieved in terms of F1-AUC. However, we saw that in some iterations, methods using partial graph or grouping signals outperform ALMSER. Interestingly, the ablation study results indicated that the positive contribution of graph and grouping signals to the active learning query strategy is subject to the task at hand. This finding will be further investigated in the following Chapter 9.

Chapter 9

Impact of the Profile of Multi-Source Entity Resolution Tasks on Active Learning

Multi-source entity resolution tasks inherently offer graph and grouping signals, which can be exploited during active learning to reduce the overall labeling effort, as demonstrated in Chapter 8. However, the degree of graph and grouping signals may vary across different multi-source tasks and is dependent on the profile of the data sources to be matched. This observation is motivated by the results of Chapter 8, indicating that the positive contribution of graph and grouping signals to the query selection active learning component is dependent on the task at hand. For example, if there is no significant overlap of records describing the same real-world object across the data sources of the multi-source task, graph signals are expected to be of low quality. Similarly, grouping signals are potentially not helpful for multi-source entity resolution tasks comprising two-source tasks with different underlying matching patterns.

In this chapter, we analyze the impact of the profiling characteristics of multi-source entity resolution tasks on the performance of active learning methods which exploit different signals as part of their query strategy, thus covering contributions [C6] and [C7] of the thesis. To do so, we propose three profiling dimensions for describing multi-source entity resolution tasks that can affect the utility of graph and grouping signals. The proposed profiling dimensions are complementary to the ones presented in Chapter 3, which solely focus on two-source entity resolution tasks. To enable our analysis, we develop ALMSERgen, a multi-source entity resolution task generator, and curate a continuum of 252 multi-source tasks along the suggested dimensions. In contrast to existing entity resolution task generators [Hildebrandt et al., 2020; Ioannou et al., 2013; Saveta et al., 2015], ALMSERgen covers desiderata relevant to the multi-source setting. We evaluate the following three active learning query strategies on the generated tasks as well as on five benchmark tasks: (i) HeALER, a committee-based query strategy proposed

by Chen et al. [2019], (ii) ALMSERgraph, a query strategy which exploits graph signals, introduced in Chapter 8, and (iii) ALMSERgroup, a query strategy which exploits graph and grouping signals, introduced in Chapter 8. By analyzing our evaluation results, we identify the best performing active learning query strategies for groups of multi-source entity resolution tasks sharing the same characteristics.

The contributions of this chapter are summarized as follows:

- We propose a set of dimensions for profiling multi-source entity resolution tasks which are relevant to the active learning query strategy.
- We develop ALMSERgen, a tool for generating multi-source entity resolution tasks along the suggested profiling dimensions.
- We analyze the experimental results of a continuum of 252 multi-source entity resolution tasks generated by ALMSERgen as well as five benchmark tasks and study the effect of their profiling characteristics on the performance of active learning methods using different query strategies.

This chapter is structured into five sections. In Section 9.1, we discuss the related work on task generators for entity resolution. Section 9.2 presents the dimensions for profiling multi-source entity resolution tasks. Section 9.3 presents the multi-source task generator ALMSERgen, which we use for generating a continuum of multi-source entity resolution tasks and enabling our analysis. In Section 9.4, we present the experimental setup and the results of our analysis. Finally, in Section 9.5, we summarize the main findings of this chapter.

The methodology as well as the evaluation of the results presented in this chapter have been published in the Proceedings of the 22nd Extended Semantic Web Conference [Primpeli and Bizer, 2022]. The code for ALMSERgen is publicly available[1] along with the 252 generated tasks used in our experimental analysis.[2]

9.1 Related Work

Numerous benchmark studies have compared the performance of different supervised entity resolution methods in passive learning [Achichi et al., 2017; Christophides et al., 2020; Konda et al., 2016; Köpcke et al., 2010] and active learning settings [Meduri et al., 2020]. However, there has been no benchmark study comparing active learning methods for multi-source entity resolution tasks to the best of our knowledge.

To allow a fair comparison of evaluation results, the existing benchmark studies typically use multiple entity resolution tasks. These tasks contain (i) records deriving from one or more data sources, such as proprietary and public databases [Saeedi

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

[2] http://data.dws.informatik.uni-mannheim.de/benchmarkmatchingtasks/almser_gen_data/

et al., 2017] and (ii) a manually labeled correspondence set of matching and non-matching record pairs. Alternatively, synthetic tasks curated with data generator tools can be used for evaluation [Ferrara et al., 2011]. Entity resolution task generators use a single de-duplicated data source as a starting point and generate mutated versions of it by applying different transformations [Ferrara et al., 2011; Ioannou et al., 2013]. The correspondence set can be thus acquired with no labeling effort by simply assigning a matching label between all mutated records originating from the same record in the initial data source. Any pair of records originating from different records in the initial data source is assigned a non-matching label.

There exist several data generators which focus on curating entity resolution tasks for Linked Data and have been used for evaluating link discovery frameworks [Achichi et al., 2017]. Such data generators produce entity resolution tasks with varying degrees of difficulty by applying structural, syntactic, and semantic transformations. The first approach towards synthetically generating entity resolution tasks for link discovery frameworks was developed by Ferrara et al. [2011] and called SWING (Semantic Web INstance Generation). SWING is based on Description Logics and therefore the structural and semantic transformation operations are bound to Description Logics, e.g. addition of disjointness restrictions. With respect to the syntactic transformations on value level, SWING offers the following three operations: (i) addition of random tokens, (ii) deletion of random tokens, and (iii) modification of random tokens. A more flexible data generator tool is EMBench, developed by Ioannou et al. [2013]. EMBench has more expressive power in comparison to SWING, as it supports aggregation and grouping operations. Additionally, EMBench extends the set of syntactic operations on value level of SWING by permutating words and replacing them with their acronyms, initials, or abbreviations. Saveta et al. [2015] develop SPIMBENCH, which relies on the value transformations of SWING but additionally sets a severity level to determine the degree of the applied modification. In comparison to SWING, EMBench, and SPIMBENCH, our data generation tool ALMSERgen applies a similar set of transformations on value level as well as user-defined severity levels. However, structural transformations are not supported since schema matching is not studied within the context of our work.

Besides data generation tools focused on evaluating link discovery frameworks, there exist several tools for generating entity resolution tasks without accounting for class and property hierarchies. An overview of such data generators is provided by Hildebrandt et al. [2020]. The DBGen [Hernández and Stolfo, 1998] and Febrl [Christen, 2009] data generators curate synthetic tasks given a pre-defined schema with person-related attributes, such as name and birthdate. The modifications applied to each attribute are hard-coded and include typographical errors, word permutations, and substitution of names with initials. The GeCo data generator [Tran et al., 2013] is a successor of the Febrl data generator and allows more flexibility regarding the schema of the generated task, as it enables the user to define their own schema. A drawback of GeCo is that it does not support parallel execution, thus leading to increased runtimes. A more systematic and scalable ap-

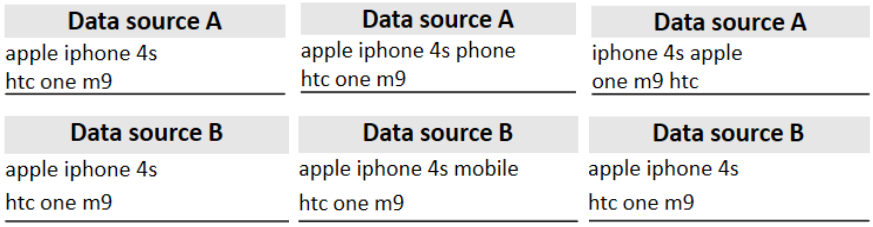
proach to generating synthetic entity resolution tasks is DaPo, developed by Hildebrandt et al. [2020]. DaPo defines an error model containing multiple error concepts such as typographical, OCR, phonetic, and formatting errors. The different concepts can be combined within a single transformation. Additionally, DaPo is built upon Apache Spark, allowing vertical, i.e. distribution of the workload to multiple cores of a single machine, and horizontal scalability, i.e. distribution of the workload to multiple machines. Similarly, ALMSERgen supports parallelism and allows for vertical scalability. In contrast to DaPo, no horizontal scalability is supported. Finally, the existing generators of entity resolution tasks can be used for curating multiple modified versions of a single data source and can thus construct a multi-source entity resolution task. However, none of them consider multi-source entity resolution task-related desiderata, which we cover in our work.

9.2 Profiling Dimensions for Multi-Source Tasks

Multi-source entity resolution tasks have certain inherent properties which are complementary to the profiling dimensions of two-source tasks, covered in Chapter 3, and can impact active learning. In this section, we define three dimensions for profiling multi-source entity resolution tasks: entity overlap, value heterogeneity, and value pattern overlap. In the following, we present how each dimension is calculated and discuss its relevance to the active learning query strategy.

Entity Overlap The dimension of entity overlap (EO) refers to the ratio of entities, i.e. distinct real-world objects as per Definition 2.2.2, that appear in more than two sources over the entities of the multi-source task that appear in two or more sources. Transforming the multi-source task into a correspondence graph, the dimension of entity overlap is calculated as $\frac{|CC_{size \geqslant 3}|}{|CC_{size \geqslant 2}|}$, where CC are the connected components of the correspondence graph. An entity overlap of 0 indicates that all entities are represented by records appearing in a maximum of two of the data sources. An entity overlap of 1 indicates that all entities are represented by records in at least three data sources. For the calculation of the entity overlap level, further fine-grained weighting factors could be considered, e.g. in how many more than three sources an entity appears. However, in the context of our work, we make a cut at three sources which is the minimum number of overlapping entities required for graph transitivity signals to fire.

We expect a multi-source task with no entity overlap to offer low-quality graph signals. Given that the maximum size of connected components is two nodes, i.e. records, no additional information can be extracted from the correspondence graph considering different graph signals, such as graph transitivity. In such multi-source entity resolution tasks, non-graph-based query strategies are expected to outperform graph-based ones.



(a) VH=0.0 **(b)** VH=0.5 **(c)** VH=1.0

Figure 9.1: Example multi-source tasks with different value heterogeneity levels.

Value Heterogeneity The dimension of value heterogeneity (VH) captures the heterogeneity of the relevant attribute values of the records that appear in different data sources and represent the same entity. We remind the reader that relevant attributes are those that contribute to the solution of an entity resolution task and are heuristically defined, as described in Section 3.4.1. The heterogeneity of values may derive from different surface forms of the same value, e.g. *iphone 4s* vs. *iphone 4s smartphone* vs. *4s iphone*, as well as spelling errors, e.g. *apple* vs. *applle*. We compute value heterogeneity as the ratio of entities in a multi-source entity resolution setting that are represented by records with dissimilar values in all their relevant attributes to all entities. The example of Figure 9.1 presents three tasks with different levels of value heterogeneity. The task of Figure 9.1a has VH=0, as both entities are represented by records with identical values in sources A and B. On the other hand, the tasks of Figures 9.1b and 9.1c have a value heterogeneity level of 0.5 and 1.0, respectively, as 50% (9.1b) and 100% (9.1c) of the entities are represented by records with different values.

We expect that multi-source entity resolution tasks with a low level of value heterogeneity are easy to solve. Considering that the matching and non-matching pairs are almost perfectly separable for such tasks, the learner can reach a high prediction accuracy even with a small number of labeled record pairs. On the other hand, given a task with high value heterogeneity, we expect that a small number of labeled record pairs can lead to the overfitting of the learner. In that case, exploiting the correspondence graph for directing the query strategy to pick record pairs that are likely erroneously predicted by the overfitted learner can be helpful.

Value Pattern Overlap The dimension of value pattern overlap (VPO) refers to the number of groups of data sources of the multi-source entity resolution setting adhering to the same attribute value patterns. The overlap of value patterns results from similar surface forms or types of errors within the record values of the data sources. For example, within the e-commerce phone product domain, different e-shops may share one of the following surface forms for representing the names of smartphones: [model] [model generation], *e.g. iphone 4s*, [model] [model generation] [product type], *e.g. iphone 4s smartphone* or [model generation] [model], *e.g. 4s iphone*.

Data source A	Data source B	Data source C
name	name	name
apple iphone 4s phone	apple iphone 4s mobile	apple iphone 4s product
htc one m9 phone	htc one m9 mobile	htc one m9 product

(a) Data sources with the same attribute value pattern

Two-source task	Matching pattern
A-B A-C B-C	JaccardSim(name) >= 0.75

(b) Matching pattern per two-source entity resolution task

Data source A	Data source B	Data source C
name	name	name
apple iphone 4s	apple iph0ne 4s	iphone 4s apple
htc one m9	htc 0ne m9	one m9 htc

(c) Data sources with different attribute value patterns

Two-source task	Matching pattern
A-B	LevenshteinSim(name) >= 0.9
A-C	JaccardSim(name) = 1.0
B-C	RelaxedJaccard(name) = 1.0

(d) Matching pattern per two-source entity resolution task

Figure 9.2: Example multi-source tasks with high (a-b) and low (c-d) value pattern overlap.

Data sources with overlapping value patterns can produce groups of entity resolution tasks with similar underlying matching patterns. We consider a matching pattern as a disjunction of conjunctions of similarity-based features and threshold values. We illustrate this observation with the example of Figure 9.2, which presents two multi-source entity resolution tasks. In the first task of Figure 9.2a, all data sources follow the same value pattern for the name attribute, which is a concatenation of multiple fine-grained attributes: [brand] [model] [model generation] [product type]. Although the [product type] is different for every data source, the value pattern overlaps in all three data sources. This further leads to a unified matching pattern across all two-source tasks, as shown in Figure 9.2b. On the other hand, the records of the data sources of the second multi-source task contain heterogeneous value patterns: [brand] [model] [model generation] for data source A, [brand] [misspelled model] [model generation] for data source B and [model] [model generation] [brand] for data source C. This results in distinct matching patterns for each two-source entity resolution task, as shown in Figure 9.2d.

We calculate the value pattern overlap as $\frac{1}{G_{VPO}}$, with G_{VPO} indicating the number of groups of data sources having the same value pattern. In order to align the range of the VPO level to the ranges of EO and VH, we rescale the VPO level values to the range [0,1]. A value pattern overlap of 1 indicates that all data sources contain records with the same value pattern and therefore construct pairwise tasks with the same underlying matching patterns. On the other hand, a value pattern overlap of 0 indicates that the records of each data source contain different value patterns and therefore the pairwise tasks contain distinct underlying matching patterns. We expect those query strategies that can identify groups of entity resolution tasks sharing similar matching patterns and distribute the queries across the groups can outperform query strategies that ignore the grouping information.

It should be noted that in the context of the exploratory analysis presented in this chapter, it is possible to exactly calculate the G_{VPO} score and thus the value

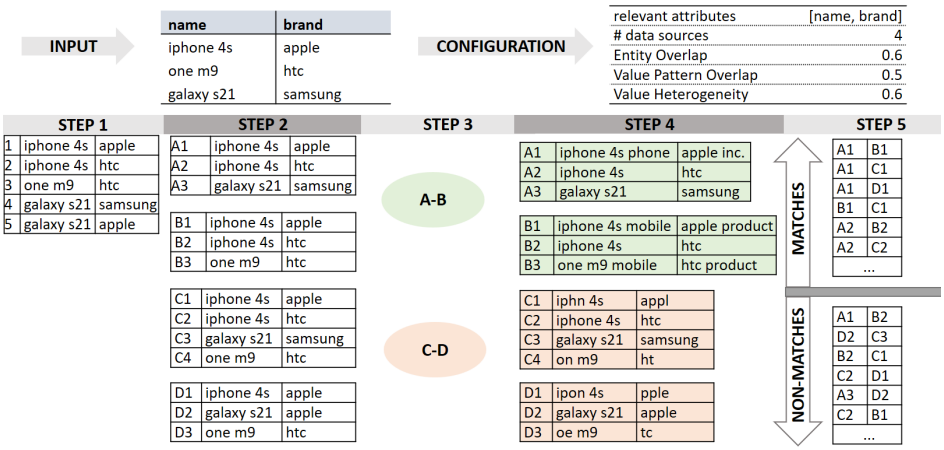


Figure 9.3: Multi-source entity resolution task curation with ALMSERgen.

pattern overlap level. This is due to the fact that our analysis is based on generated multi-source tasks for which we pre-define and inject a set of attribute-value patterns to generated data sources, as will be presented in Section 9.3. However, in a real-world setting, the attribute value patterns are usually not fixed and therefore it is hard to precisely calculate the G_{VPO} level. For these cases, the value pattern overlap can be estimated by the overlap of matching patterns among the two-source tasks of the multi-source setting. One way to approximate the latter is to calculate the naive transfer learning scores among all combinations of two-source tasks [Thirumuruganathan et al., 2018]. We will present in detail and use this approximation for estimating the VPO level of benchmark tasks in our experimental analysis presented in Section 9.4.4.

9.3 ALMSERgen: a Multi-Source Task Generator

In order to test the performance of different active learning query strategies over a large continuum of multi-source test cases, we develop ALMSERgen, a multi-source entity resolution task generator. ALMSERgen takes a set of records as input and generates a multi-source task of a pre-defined amount of data sources by replicating the input record set and injecting transformations along the three dimensions explained in Section 9.2. In the following, we explain each component of ALMSERgen along with the illustrated example of curating a multi-source entity resolution task given a pre-defined configuration of Figure 9.3.

Step 1: Complement Initial Set Depending on the domain and the integration task at hand, different attributes might be relevant for matching. For example, for the task of matching phone records, one might consider that the combination of phone name and phone brand identifies a distinct phone, while in more fine-

grained entity resolution tasks, the phone color might also be important. We call the attributes that are useful for distinguishing real-world objects of a specific domain and are relevant for entity resolution *relevant attributes*, and they are given as input to ALMSERgen. Considering that the input set of records may not contain enough examples for the relevant attributes to show, ALMSERgen artificially activates the relevant attributes by replicating 20% of the input records and replacing a subset of the relevant attribute values with random non-identical values of the same attribute. The non-relevant attribute values are simply copied from the original record to the replicated records. In the example of Figure 9.3, the input set contains three records. Given that the relevant attributes are configured to be *name* and *brand*, ALMSERgen generates the additional records *2.iphone 4s - htc* and *5.galaxy s21 - apple*, which represent phone entities different from the ones that the records 1 and 4 represent.

Step 2: Distribute Records over Data Sources Next, the entity overlap level (EO) of the multi-source task is fixed. Given a pre-defined EO level value in the range [0,1], we iterate over all initial entities (IE) produced in Step 1 and add a subset of them, the amount of which equals $EO \times |IE|$, to more than two data sources. For specifying in how many more than two sources the selected subset of entities should be added, we follow a power-law distribution, i.e. most entities are described by records in few data sources while a few entities are described by records in all data sources. Therefore, given that a record describing the same entity is selected to be added in more than two data sources, the probability that it is added in x data sources is $1/x$, with $x > 2$. In the illustrated example, the EO level is set to 0.6, i.e. 60% of the five entities produced in Step 1, are added to more than two sources: the entity with id 1, which is added in four sources, and the entities with ids 2 and 3, which are added in three sources.

Steps 3-4: Inject Groups of Patterns In the next step, the levels of value pattern overlap (VPO) and value heterogeneity (VH) are fixed. These two dimensions are interwoven, considering that VH defines how many records across all data sources contain heterogeneous representations for the same real-world object and VPO controls the similarity of the value patterns of the records across all data sources. Given the pre-defined VPO level, ALMSERgen creates groups of data sources to which the same value pattern is injected. The same value pattern is injected in the records of the groups representing a subset of entities, the amount of which is $VH \times |IE|$, with IE being the initial entities generated in Step 1 and VH being the value heterogeneity level in the range of [0,1].

A value pattern is comprised of distinct combinations of attributes and value transformations. ALMSERgen offers the following value transformations similarly to existing data generators for entity resolution [Ioannou et al., 2013; Saveta et al., 2015]:

1. Addition of random characters
2. Deletion of random characters
3. Modification of random characters
4. Shuffling and modification on word level
5. Shuffling of words
6. Addition of random words
7. Subtraction of (5/10/20)% of the value
8. Addition of (5/10/20)% of the value

Transformations 1-6 are performed on string attributes, while transformations 7-8 are applied only on numerical attributes. Finally, for the transformations 1-4, a level of severity in the range of [0.1, 0.5] is randomly picked for each value pattern, i.e. maximum of 50% of the characters can be modified or deleted to ensure that the identity of each entity is not completely altered and remains distinguishable. After this step, the curation of the data sources of the multi-source setting is completed.

In the example of Figure 9.3, the VPO level is set to 0.5 and the VH level is set to 0.6. This further implies that the data sources will be grouped into two groups of overlapping value patterns, G1: A-B and G2: C-D. For each group, one combination of attribute and value transformation is randomly chosen and injected into the records describing 60% of the five entities of Step 1, i.e. the entities with ids 1, 3, and 5. The value pattern for both attributes of the records in G1 is the *addition of random words*. The value pattern for G2 is the *deletion of random characters with a severity of 0.2*.

Step 5: Derive Matching and Non-Matching Pairs In the final step, ALMSERgen derives the complete set of matching pairs considering all pairwise combinations of replicated records referring to the same entity, e.g. A1-B1. For deriving hard non-matching pairs, we extract all combinations of records and their corresponding negative examples injected in Step 1, e.g. A1-C2. Additionally, we randomly pick easy non-matching record pairs, e.g. A1-C3, until the ratio of matching to non-matching pairs is 1/3.

9.4 Experimental Evaluation

In this section, we present the results of our experimental analysis. First, we provide the details of the ALMSERgen configuration and the active learning setup. Next, we present the experimental results on a continuum of 252 generated tasks and discuss our main findings on the performance of three active learning methods using different query strategies with respect to the profiling characteristics of the tasks. Finally, we verify our findings on the five multi-source entity resolution tasks used in the evaluation Section 8.2 of Chapter 8.

Table 9.1: Explanation of VPO levels in ALMSERgen.

VPO	Groups of data sources with the same attribute value pattern	# Groups of data sources	# Groups of two-source tasks
0.0	{1}{2}{3}{4}{5}{6}	6	15
0.2	{1,2}{3}{4}{5}{6}	5	11
0.4	{1,2}{3,4}{5}{6}	4	8
0.6	{1,2}{3,4}{5,6}	3	6
0.8	{1,2,3}{4,5,6}	2	3
1.0	{1,2,3,4,5,6}	1	1

9.4.1 ALMSERgen Configuration

We provide a set of 1000 records, each describing a song entity as input to ALMSERgen. The input dataset is a subset of the last.fm song dataset.[3] Each song record is described with the following four attributes: title, release, artist, and country. We configure all of the four attributes as relevant ones and set the number of curated data sources of each setting to 6. We iterate in steps of 0.2 in the range [0.0, 1.0] for the dimensions of entity overlap (EO) and value pattern overlap (VPO) and in steps of 0.1 in the range [0.2, 0.8] for the value heterogeneity (VH) dimension. The defined ranges and steps result in the curation of 252 multi-source entity resolution tasks. As the VPO levels are rescaled to match the ranges of the other two dimensions, we present the details of each VPO level in terms of groups of data sources, amount of resulting groups of data sources, and amount of resulting groups of two-source tasks in Table 9.1. For example, a VPO level of 0.8 indicates that two groups of three data sources each are formed. The data sources of the same group, e.g. 1, 2, and 3 as shown in Table 9.1, are injected the same attribute value pattern. This results in three groups of two-source tasks having the same underlying matching patterns.

9.4.2 Experimental Setup

We compare the performance of three active learning methods using the query strategies HeALER [Chen et al., 2019], ALMSERgraph, and ALMSERgroup. The query strategies use different criteria for assessing the informativeness of the record pairs of the unlabeled pool. We remind the reader that HeALER uses a committee of heterogeneous classifiers and selects record pairs with the highest disagreement among the predictions of the committee. We use the following five classification models for building the committee: logistic regression, linear SVM, decision tree, XGBoost, and random forest. ALMSERgraph exploits the signals of the correspondence graph constructed from the predictions of the learner and selects record pairs for labeling that are likely erroneously predicted by the learner. ALMSERgroup uses the same strategy as ALMSERgraph, but further exploits grouping

[3] http://millionsongdataset.com/lastfm/

signals for restricting the number of record pairs of the unlabeled pool. A detailed description of the ALMSERgraph and ALMSERgroup query strategies is provided in Section 8.1.5. In our experiments, we only modify the query selection component while keeping the rest of the active learning setting the same. For the sake of convenience, we will use the abbreviations HeALER, ALMSERgraph and ALMSERgroup to denote the methods that use the corresponding query selection strategies. To avoid confusion, we remind the reader that these were abbreviated as QHC, ALMSERgraph_qs and ALMSERgroup_qs, respectively, in the evaluation Section 8.2 of the previous chapter.

We allow 200 iterations for each experiment and use single queries. In the case that the query strategy assigns the maximum informativeness score to more than one record pair of the pool, one of them is randomly selected for annotation. We use a random forest classifier[4] as a learner and measure the F1 score of its predictions on the test set in each iteration. No graph or grouping signals are exploited for model learning. We conduct three experimental runs for each multi-source task and each method. Finally, we use the area under the mean F1 curve, which we abbreviate with F1-AUC, for comparing the active learning methods along all iterations. A higher F1-AUC score signifies overall better results in terms of F1 score.

9.4.3 Results and Analysis of ALMSERgen Tasks

In this section, we compare the results of the three active learning methods on the continuum of the curated multi-source entity resolution tasks and identify which signals are helpful to the query selection component given the profiling characteristics of the tasks. For our analysis, we use 2D and 3D scatter plots with the color of each circle representing the winning active learning method. The varying circle sizes indicate the difference of the outperforming method to the second-best method in terms of F1-AUC for iterations 50 to 200. Large dots signify clear winners, while smaller dots represent winning methods that are only slightly better than the runner-up methods.

Figure 9.4 presents the overall comparison results of the three active learning methods on the continuum of the 252 multi-source tasks along the three dimensions described in Section 9.2: value heterogeneity (VH), value pattern overlap (VPO), and entity overlap (EO). In 41.6% of the tasks, HeALER is the winning active learning query strategy, while ALMSERgraph and ALMSERgroup outperform in 25.8% and 33% of the tasks, respectively. Looking at the 3D plot of Figure 9.4, we observe four main patterns, which we indicate with the dotted circles and the abbreviations P1-P4. In the following, we discuss in detail each pattern. Additionally, we report the best performing active learning methods for the tasks of every pattern by relating their results to the runner-up methods as well as to *passive F1*, i.e. the upper bound F1 scores achieved in a passive learning setting with a random forest classifier being trained on the complete pool of records pairs.

[4]With default parameters as defined by the scikit-learn library version 0.24.2

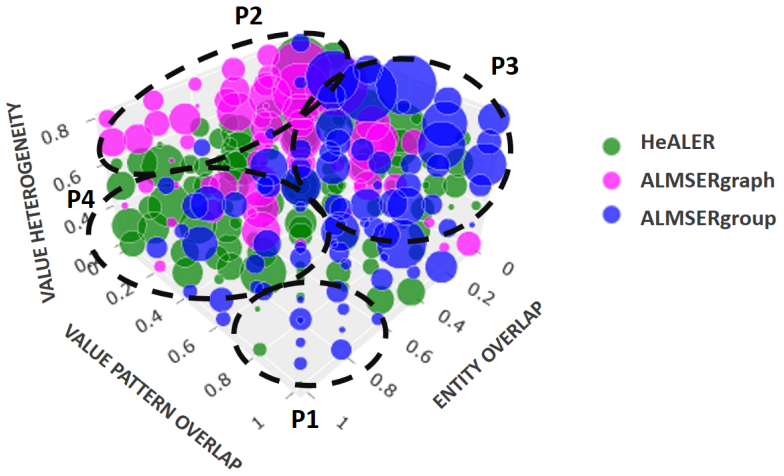
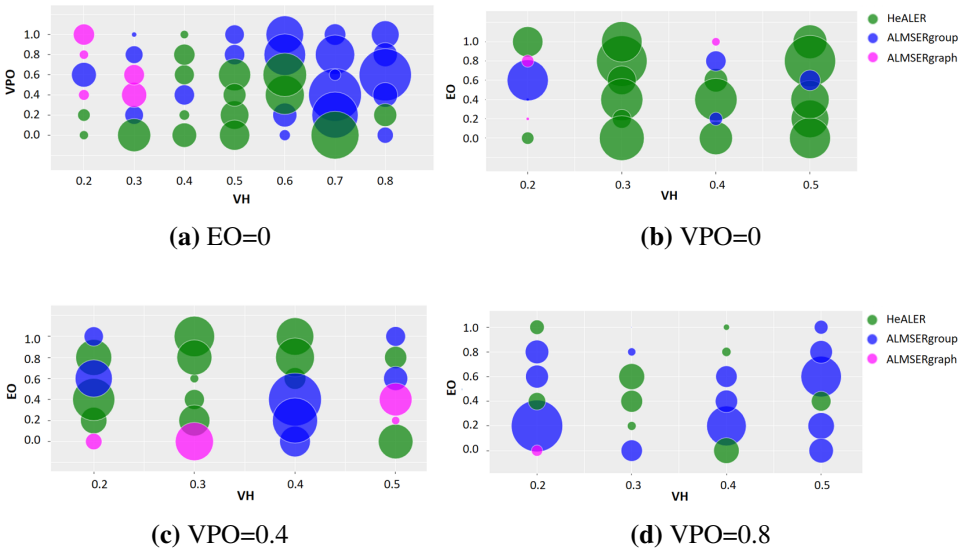


Figure 9.4: Overall task continuum results.
Outperforming AL methods per task along the three profiling dimensions. The circle size indicates the F1-AUC difference to the runner-up method.

P1 - No clear winner for easy tasks. For all of the tasks with entity and value pattern overlap larger than 0.6 as well as a minimal value heterogeneity of 0.3 or less, HeALER and ALMSERgroup outperform ALMSERgraph, as shown in the P1 circled settings of Figure 9.4. However, the mean F1-AUC difference to the runner-up methods is only 0.51 for the settings in which HeALER outperforms and 0.68 for the settings in which ALMSERgroup outperforms. This indicates that the best-performing methods are not clear winners as they only marginally outperform the second best method. The mean passive F1 score for all tasks adhering to this pattern is 0.983, while the mean F1 of the best performing active learning methods at the final 200th iteration is 0.961. We consider such multi-source entity resolution tasks relatively easy to solve as the high overlap of mostly homogeneous entity records eases the discovery of the few distinct matching patterns, i.e. selecting one matching record pair for annotation can help the classifier to learn the underlying pattern of many other record pairs at once.

P2 - Graph signals are helpful for tasks with high value heterogeneity. In 71.6% of the tasks with a value heterogeneity level larger than 0.5, ALMSERgraph is the winning strategy with a mean F1-AUC difference of 3.41, given that the value pattern overlap level is 0.6 or below. The mean passive F1 score for all tasks of this pattern is 0.888, while the mean F1 of the best performing active learning methods at the final 200th iteration is 0.828. Such tasks are harder to solve as they contain heterogeneous value representations for a large number of entities. Furthermore, the low value pattern overlap level signifies that there exist many different underlying matching patterns. Exploiting the signals from the correspondence graph



(a) EO=0 **(b)** VPO=0

(c) VPO=0.4 **(d)** VPO=0.8

Figure 9.5: Outperforming AL methods per task for specific settings. The circle size indicates the F1-AUC difference to the runner-up method.

leads to the faster discovery of all underlying matching patterns in comparison to committee-based query strategies. However, this observation only holds when there exists a minimum entity overlap, i.e. EO>0.0. For multi-source tasks with an entity overlap level of 0, i.e. when all entities are represented by one record in a maximum of two data sources, the correspondence graph does not have a rich structure as the maximum component size is 2. Therefore exploiting graph signals cannot lead to the selection of informative query candidates, which causes the ALMSERgraph query strategy to fail in most cases. Figure 9.5a compares the three query strategies along all tasks with EO=0. In 88% of these tasks, either HeALER or ALMSERgroup outperform ALMSERgraph with a mean F1-AUC difference of 3.67 to the runner-up methods.

P3: Grouping signals are helpful for tasks with low value heterogeneity and high pattern overlap. In 55.5% of the tasks with a value heterogeneity lower than 0.5 and a value pattern overlap larger than 0.5, ALMSERgroup is the winning active learning strategy with a mean F1-AUC difference to the runner-up method of 1.52. However, ALMSERgroup does not deliver better results over HeALER in the case of multi-source tasks with low pattern overlap. We illustrate and further analyze this observation with Figures 5b-5d. The figures depict the winning strategies for tasks with a value heterogeneity level of 0.5 or lower and three different value pattern overlap levels: 0.0, 0.4, and 0.8. We see that for the multi-source entity resolution tasks, where the value pattern overlap is 0 (Figure 5b), i.e. different underlying matching patterns exist in each two-source task of the setting, HeALER outperforms ALMSERgroup in 66% of the settings. The mean F1-AUC

difference of HeALER's results to the runner-up method is 3.30, while for the tasks where ALMSERgroup outperforms, the mean F1-AUC difference to the runner-up method is 1.30. With the increase of the value pattern overlap level, we can observe that the grouping signal starts contributing to the selection of more informative record pairs for labeling. For VPO=0.4 (Figure 5c), HeALER outperforms in 54% of the tasks with a mean F1-AUC difference to the runner-up method of 1.93, while the mean F1-AUC difference for the settings where ALMSERgroup is the best performing query strategy is 2.31. Finally, ALMSERgroup performs the best in 58.3% of the tasks when the value pattern overlap level is 0.8 (Figure 5d) with a mean F1-AUC difference to the second-best method of 2.26, while HeALER outperforms in 37% of the tasks with a smaller F1-AUC difference of 0.98.

P4: Graph and grouping signals are not needed for tasks with low value heterogeneity and low pattern overlap In 89.5% of the tasks with a value heterogeneity of 0.5 or lower, the HeALER and ALMSERgroup query strategies outperform ALMSERgraph independently from the other two dimensions. The F1-AUC difference to the runner-up methods is 2.26 and 1.71 for HeALER and ALMSERgroup, respectively. In terms of F1 scores, the tasks of this pattern lie between the results of the tasks in P1 and P2. More concretely, the mean passive F1 is 0.941 and the mean F1 of the best performing active learning methods at the 200th iteration is 0.91. For such tasks, solely exploiting graph signals for picking record pairs for annotation does not lead to the faster discovery of matching patterns.

As already introduced in the analysis of P3, the contribution of grouping signals is positively related to the value pattern overlap level, i.e. grouping signals contribute less for tasks with a low value pattern overlap level. More concretely, we observe that for 67% of the tasks with a value heterogeneity and a value pattern overlap of 0.5 or lower ALMSERgroup underperforms compared to the other two methods. In the following paragraph, we further investigate this observation.

Interpretation of the Contribution of Grouping Signals In order to investigate the reasons why grouping signals do not contribute to tasks with a low value pattern overlap, we perform a two-step analysis: First, we evaluate how representative the metric of task relatedness is for finding groups of two-source tasks with similar patterns. We remind the reader that ALMSERgroup uses the metric of task relatedness in order to select a subset of representative two-source tasks and reduce the size of the unlabeled pool. Second, we evaluate to what extent ALMSERgroup identifies all groups of two-source entity resolution tasks of each setting.

For the first part of our analysis, we calculate the cosine similarity of the naive transfer learning (NTL) and the task relatedness (RLTD) for each combination of two-source tasks in the continuum of multi-source tasks. A high naive transfer learning score between two tasks A and B indicates that the tasks have the same underlying matching patterns, as a model trained on the record pairs of task A performs well when applied to task B. A high similarity between the NTL scores

Table 9.2: Evaluation of using task relatedness for identifying groups of tasks with similar patterns.

VPO	Sim. NTL-RLTD	Settings with sufficient subset of tasks	# Distribution of settings over range			
			[2,4]	[5,6]	[7,8]	[9,15]
1	0.81	42 (100%)	38	3	0	1
0.8	0.76	37 (88.0%)	29	9	4	0
0.6	0.75	18 (42.8%)	9	25	5	3
0.4	0.71	12 (28.5%)	6	14	13	9
0.2	0.69	6 (14.2%)	4	9	11	18
0	0.70	7 (16.6%)	8	8	2	24

and the RLTD scores for the complete multi-source setting indicates that the second is a good unsupervised approximation of the first and can therefore lead to the discovery of groups of entity resolution tasks.

For the second part of our analysis, we evaluate for how many of the multi-source tasks ALMSERgroup selects a sufficient subset of two-source tasks to query from. A sufficient subset contains at least one two-source task per group of tasks with similar matching patterns. In addition, we report the size of the selected subsets of the two-source tasks in order to evaluate to what extent ALMSERgroup manages to reduce the number of query candidates to select from.

Table 9.2 presents the results of our evaluation on using task relatedness for grouping tasks for different VPO levels. We observe that higher VPO levels lead to a higher similarity of NTL and RLTD scores. Therefore, we can conclude that task relatedness can better approximate the naive transfer learning scores in multi-source entity resolution tasks with a high value pattern overlap level, i.e. a small number of groups of two-source tasks share the same underlying patterns. Additionally, we observe that ALMSERgroup better identifies sufficient subsets of two source tasks to query from for higher VPO levels. In addition, we see that ALMSERgroup achieves a large candidate reduction for high VPO levels: for VPO 1.0, ALMSERgroup only selects candidates from a maximum of 4 out of the 15 two-source tasks in 90% of the settings. In contrast, we see that the more distinct matching patterns exist in the multi-source setting, the harder it is for ALMSERgroup to select a sufficient subset of two-source tasks to query from, e.g. when the VPO is 0, ALMSERgroup can only identify a sufficient subset of two-source tasks in 16.6% of the cases.

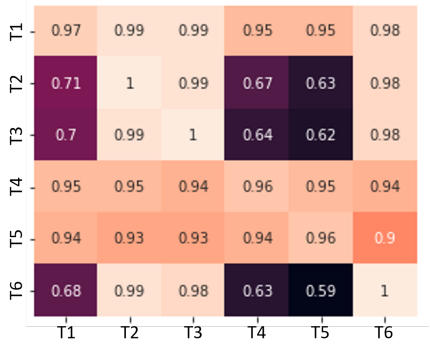


Figure 9.6: Naive transfer learning scores per two-source task for the computers multi-source entity resolution benchmark task.

9.4.4 Results and Analysis of Benchmark Tasks

In this section, we explore the impact of the profiling characteristics of the five benchmark multi-source entity resolution tasks introduced and used in the evaluation Section 8.2, i.e. computers, computers_mut, MusicBrainz, MusicBrainz_mut, and restaurants, on the performance of HeALER, ALMSERgraph, and ALMSERgroup. Thus, we aim to verify whether the patterns observed with the multi-source tasks generated with ALMSERgen are confirmed with the benchmark tasks. In the following, we describe how we compute the profiling dimensions of the benchmark tasks. Next, we present the active learning results and discuss them in relation to the patterns analyzed in Section 9.4.3.

Profiling of Benchmark Multi-Source Tasks

We compute the value heterogeneity and the entity overlap levels of all benchmark tasks, as described in Section 9.2. As it is not possible to know the exact number of groups of data sources adhering to the same value patterns, we approximate the value pattern overlap level by estimating the number of groups of two-source tasks of the multi-source setting having the same underlying matching patterns. To do so, we use the naive transfer learning scores of a random forest classifier for all pairwise combinations of two-source tasks and extract the subsequence of tasks that best generalizes over all tasks, using Algorithm 2 (part B) presented in Section 8.1.6. We illustrate this approximation with the example of Figure 9.6, presenting the naive transfer learning scores for each pairwise two-source task combination for the computers task in a heatmap view. The computers task contains four data sources, which, when combined pairwise, produce six two-source entity resolution tasks T1-T6. A random forest classifier trained on T1 performs 0.99 on F1 score when applied on T3 and 0.98 on F1 score when applied on T6, as shown in Figure 9.6. By applying Algorithm 2 (part B), we identify that T1 generalizes that best over all tasks and therefore the VPO level is 1.0, i.e. there exists only one

two-source task whose underlying matching pattern can cover all two-source tasks of the multi-source setting. Table 9.3 (left) summarizes the profiling information of the five benchmark tasks, which we discuss in detail below.

Computer Products Tasks The computers task and its mutated version computers_mut contain computer product records from four e-commerce websites and therefore comprise six two-source tasks T1-T6. For the computers task, the underlying matching patterns of the two-source tasks T2, T3, and T6 focus on the Jaccard similarity of the *part number* attribute, while the patterns of the tasks T1, T4, and T5 rely additionally on the containment similarity of the *title* attribute. Therefore, a model trained on the tasks T1, T4, and T5 can generalize well on all two-source tasks of the computers multi-source task, as indicated in the heatmap of Figure 9.6. Given that only one of those tasks is needed to cover the patterns of the complete setting, we calculate the VPO level to be 1.0. Similarly, we calculate the VPO level of the computers_mut task to be 0.8, as there exist two distinct matching patterns shared within two groups of two-source tasks. The value heterogeneity is calculated to be 0.40 for the computers task and 0.43 for the computers_mut, while the entity overlap level is 0.44 for both tasks.

Music Records Tasks The music records tasks are based on song records from the MusicBrainz dataset and comprise ten two-source tasks T1-T10. The underlying matching patterns of the MusicBrainz task rely mostly on the combination of the Levenshtein similarity of the *album* attribute and the *song length* similarity for five out of the ten two-source tasks. The rest of the tasks have distinct patterns, thus leading to 3 groups, i.e. a VPO level of 0.6. The MusicBrainz_mut task is a mutated version of the MusicBrainz task and contains more complex matching patterns, which are shared between only a few data source pair combinations. For this task, the VPO level is 0.4.

Restaurant Task The restaurant task derives from the pairwise combination of four data sources and comprises six two-source tasks T1-T6. Five of the six two-source tasks share the same matching pattern, which relies on the *zip* number equality and the relaxed Jaccard similarity of the *address*. However, one of the six two-source tasks does not adhere to this matching pattern. Therefore groups of two-source tasks with similar matching patterns emerge, i.e. the VPO level is 0.8.

Active Learning Results for Benchmark Tasks

We present the results of HeALER, ALMSERgraph, and ALMSERgroup on the five benchmark multi-source tasks in Table 9.3. We report the F1-AUC, i.e. the area under the F1 curve between iterations 50-200, and the F1-AUC difference of the outperforming to the runner-up method, similarly to our analysis in Section 9.4.3. Additionally, we report the mean F1 scores of the three experimental runs at the 85th, 150th, and final 200th active learning iteration.

Table 9.3: Profiling information and active learning results for benchmark multi-source entity resolution tasks.

Task	VH	EO	VPO	Method	F1-AUC	F1-AUC diff.	F1@85	F1@150	F1@200
computers	0.40	0.44	1.0	HeALER	135.62		0.893	0.912	0.918
				ALMSERgraph	**138.96**	1.32	0.921	0.932	0.938
				ALMSERgroup	137.64		0.904	0.931	0.932
computers_mut	0.43	0.44	0.8	HeALER	127.66		0.841	0.850	0.866
				ALMSERgraph	128.95	1.24	0.824	0.879	0.878
				ALMSERgroup	**130.19**		0.864	0.877	0.883
MusicBrainz	0.14	0.50	0.6	**HeALER**	**140.07**		0.931	0.941	0.945
				ALMSERgraph	138.37	1.70	0.913	0.930	0.934
				ALMSERgroup	137.02		0.888	0.926	0.918
MusicBrainz_mut	0.19	0.50	0.4	**HeALER**	**132.43**		0.857	0.895	0.908
				ALMSERgraph	131.38	1.05	0.868	0.889	0.896
				ALMSERgroup	127.19		0.820	0.879	0.888
restaurants	0.14	0.35	0.8	**HeALER**	**138.51**		0.921	0.927	0.927
				ALMSERgraph	138.21	0.30	0.918	0.923	0.926
				ALMSERgroup	137.48		0.913	0.920	0.921

We observe that the graph and grouping signals improve the active learning results over the HeALER baseline for both computer product tasks. The profiling dimensions of these tasks lie between patterns P2 and P4, i.e. graph signals contribute due to the rather high value heterogeneity while grouping signals contribute due to the high value pattern overlap level. Compared to HeALER, we observe that graph and grouping signals contribute until the last 200th iteration. In contrast, the differences in F1 score between ALMSERgraph and ALMSERgroup are only marginal after the 150th iteration. The music records tasks verify the pattern P3 of our analysis. Given the low value heterogeneity and value pattern overlap levels of the tasks, graph and grouping signals are not helpful for improving the active learning results over HeALER. Finally, pattern P1 of our analysis is confirmed by the results of the restaurants task, which has a low value heterogeneity and a high value pattern overlap. Although HeALER outperforms the other two methods for this task in terms of F1-AUC, the F1-AUC difference to the runner-up method is only 0.30, indicating that there is no clear winner for the task.

9.5 Discussion and Conclusion

In this chapter, we explored the impact of the characteristics of multi-source entity resolution tasks on the performance of three active learning methods with different query strategies: HeALER, ALMSERgraph, and ALMSERgroup. The query strategies utilize different types of signals for selecting record pairs for labeling. To the best of our knowledge, our work is the first benchmark study on active learning methods for multi-source entity resolution.

Given that the dimensions for profiling entity resolution tasks proposed so far, and discussed in Chapter 3, focus solely on the two-source setting, we established three dimensions for profiling multi-source entity resolution tasks: entity over-

lap, value heterogeneity, and value pattern overlap. The proposed dimensions are unique to the multi-source setting and relevant to the active learning query strategy.

To enable our analysis, we developed ALMSERgen, a multi-source task generator, and curated a continuum of 252 multi-source entity resolution tasks. ALMSERgen constructs multi-source entity resolution tasks given an input data source and a pre-defined configuration along the three dimensions. ALMSERgen is the first data generator that considers multi-source entity resolution task-related desiderata. In comparison to existing entity resolution task generators developed for evaluating link discovery frameworks [Ferrara et al., 2011; Ioannou et al., 2013; Saveta et al., 2015], ALMSERgen applies a similar set of transformations on value level but does not consider any transformations on schema level. In comparison to the DaPo entity resolution task generator [Hildebrandt et al., 2020], ALMSERgen is less scalable, as it only supports parallel execution on multiple cores of a single machine (vertical scalability) but no parallel execution on multiple machines (horizontal scalability).

We evaluated three active learning methods using the HeALER, ALMSERgraph, and ALMSERgroup query strategies on the 252 generated tasks as well as on five benchmark tasks. Our findings showed that all methods perform equally well for easy multi-source entity resolution tasks in which a high overlap of entities described with rather homogeneous record values exists. With the increase of the value heterogeneity of records describing the same entity, grouping signals were shown to improve the active learning performance, given that there exist a few groups of two-source tasks sharing the same underlying matching patterns. Exploiting graph signals as part of the query strategy was shown to improve the active learning performance for tasks containing large amounts of matching records with heterogeneous attribute values.

Chapter 10

Conclusion

Entity resolution aims to identify records from one or more data sources that refer to the same real-world object [Christen, 2012]. It has been studied for decades [Fellegi and Sunter, 1969] and continues to play an important role in many different application areas, such as public health [Jaro, 1995], crime detection [Phua et al., 2010] and e-commerce [Peeters et al., 2020b].

An important challenge in entity resolution is the absence of common object identifiers among different data sources [Christen, 2012]. To circumvent this challenge and reconcile the heterogeneity of records referring to the same real-world object, entity resolution methods either rely on heuristics developed by experts or use learning-based approaches that apply machine learning techniques [Papadakis et al., 2021]. Supervised entity resolution methods constitute a family of learning-based approaches that use labeled data, typically matching and non-matching record pairs, to train a binary classification model [Christophides et al., 2020; Papadakis et al., 2021]. The trained model can then predict matching and non-matching relations between new record pairs.

The predictive performance of an entity resolution method can vary given the characteristics of the task at hand [Köpcke et al., 2010; Mudgal et al., 2018]. Therefore, uncovering the specific challenges associated with different entity resolution tasks is essential for understanding the strengths and weaknesses of different entity resolution methods. In this thesis, we proposed a set of dimensions for profiling entity resolution tasks and used them for comparing and grouping existing benchmark tasks.

Acquiring labeled data for training machine learning models is expensive and time-consuming, and it is thus considered a major limitation of supervised entity resolution methods [Papadakis et al., 2021]. In this thesis, we researched two approaches for reducing the labeling effort involved in constructing training sets for entity resolution tasks with different profiling characteristics. The first approach explored the potential of using the Semantic Web as a source of distant supervision for entity resolution. The second approach focused on active learning techniques, which rely on the fundamental idea that a machine learning model can achieve bet-

ter predictive performance if trained on a small but carefully chosen set of labeled instances [Settles, 2012].

This chapter summarizes the three parts of the thesis, outlines our main contributions, and discusses open issues and future work. In the last section of this chapter, we discuss the research impact of our work.

10.1 Part I: Entity Resolution

10.1.1 Summary and Contributions

Profiling Entity Resolution Benchmark Tasks In the first part of the thesis, we introduced the reader to the fundamental concepts of entity resolution, discussed how symbolic and subsymbolic methods target the individual steps of the entity resolution workflow, and provided a systematic understanding of the challenges of entity resolution tasks. To achieve the latter, we proposed a heuristic for extracting attributes relevant to solving the task and defined the following dimensions for profiling entity resolution tasks: sparsity, schema complexity, textuality, development size, and corner cases. We used the proposed heuristic and dimensions to profile and group 21 benchmark tasks into five groups entailing similar challenges. Prior to profiling, we complemented the correspondence sets of 17 out of the 21 benchmark tasks by adding a fixed set of non-matching record pairs and/or split them into fixed train, validation, and test sets, to support the reproducibility and comparability of the results of different entity resolution methods. Finally, we evaluated the difficulty of each group of benchmark tasks by establishing baseline evaluation results.

The proposed profiling dimensions go beyond the dimensions for profiling entity resolution tasks proposed so far in related work, which solely focus on the records of the data sources to be matched [Mudgal et al., 2018], as they also consider properties of the correspondence set of record pairs. Furthermore, our grouping scheme complements the categorization of entity resolution tasks proposed by Mudgal et al. [2018] into *structured*, *textual*, and *dirty*. This is necessary as we identified that a more fine-grained categorization is required for both textual and structured tasks. With regard to textual tasks for which subsymbolic methods are known to excel [Mudgal et al., 2018; Papadakis et al., 2021], we found that they can be equally well solved using symbolic entity resolution methods if there exists a small number of corner cases in the correspondence set. With regard to structured tasks, we observed that structuredness alone is not enough to assess the difficulty of a task, and additional dimensions, such as schema complexity and density, need to be considered.

10.1.2 Open Issues and Future Research

Our analysis, grouping, and evaluation of baseline entity resolution methods were conducted against 21 entity resolution benchmark tasks. We selected the tasks with

the goal of including in our study diverse tasks, in terms of profiling characteristics, that have been widely used for benchmarking [Köpcke et al., 2010; Li et al., 2020; Mudgal et al., 2018]. Arguably, the list of selected tasks is not extensive, as not all publicly available entity resolution benchmark tasks have been considered in our study, including some of the tasks of the MDedup repository[1] as well as the tasks from the OAEI instance matching track.[2] Future research works on entity resolution using benchmark tasks not covered in our analysis would need to calculate the values of the five profiling dimensions in order to uncover the specific challenges associated with the tasks. However, this does not entail high efforts as we have made the code for profiling entity resolution tasks publicly available.

One limitation of our profiling analysis is that the calculation of the sparsity, textuality and corner cases dimensions relies on the extracted relevant attributes. The latter are heuristically approximated using a wrapper feature extraction method [Khalid et al., 2014] optimized for a random forest classifier. Future work may consider an ensemble of different classification models for extracting relevant attributes. An additional interesting direction for future research would be to analyze the differences in the values of the profiling dimensions, which result when different classification models are applied for the calculation of the relevant attributes.

10.2 Part II: The Semantic Web as Distant Supervision for Entity Resolution

10.2.1 Summary and Contributions

In the second part of the thesis, we explored the potential of using semantic annotations, which are one realization form of the Semantic Web, as a source of distant supervision for entity resolution.

Adoption of Semantic Annotations In the first chapter of this part, we presented how semantic annotations can be embedded in HTML pages and gave an overview of the four main markup formats as well as the schema.org vocabulary. Next, we profiled the growth of semantic annotations using the four markup formats and the schema.org vocabulary from 2012 to 2020. For our profiling analysis, we used the adoption statistics of semantic annotations published yearly by the Web Data Commons project. The project was initiated by Prof. Dr. Christian Bizer and Dr. Hannes Mühlheisen in 2012 with the focus to extract, profile, and publicly provide semantically annotated data from Common Crawl,[3] the largest web corpus available to the public. The contribution of this thesis to the Web Data Commons

[1] https://hpi.de/naumann/projects/repeatability/duplicate-detection/mdedup.html
[2] http://oaei.ontologymatching.org
[3] http://commoncrawl.org/

project was the yearly extraction, profiling, and publication of semantically annotated web data during the period 2016 to 2021. By analyzing the adoption trends of different markup formats, we identified that although there is a constantly growing amount of websites using semantic annotations, not all markup formats show the same growth trends. We showed that Microdata and JSON-LD have dominated in terms of adoption over the hCard and RDFa markup formats. Additionally, we found that the Microdata and JSON-LD markup formats are mostly used together with the schema.org vocabulary, as in 2020, at least 94.4% of the entities markedup with either format were annotated with at least one schema.org class or property.

Estimating the Potential of Semantic Annotations as Distant Supervision for Entity Resolution Given the adoption statistics, we conducted a use-case analysis in order to explore the potential of using semantically annotated schema.org product and local business annotations as distant supervision for domain-specific entity resolution tasks. To do so, we profiled the adoption of identifying schema.org properties for the selected classes and grouped the entities considering the overlap of the annotated identifiers. We observed that although schema.org provides terms for both classes, which can be used for annotating unique entity identifiers, there exist considerable differences concerning the adoption of such properties among the analyzed classes. These differences are mainly attributed to the annotation recommendations from large search engines such as Google and Yahoo. Despite those differences, we identified that it is possible to obtain more than 50 thousand matching record pairs for each of the analyzed classes, given the grouped entities. This indicates that semantic annotations show a big potential for generating large training sets for entity resolution tasks related to the product and business domains. Although semantic annotations have been exploited as distant supervision for other downstream applications, such as information extraction [Foley et al., 2015; Meusel and Paulheim, 2014], to the best of our knowledge, we are the first to explore their potential as a source of distant supervision and thus as a means of reducing the labeling effort for entity resolution.

The WDC Product Corpus for Entity Resolution In the second chapter of this part, we focused on the product-specific use-case, investigated different errors related to schema.org product annotations, and developed a pipeline for cleansing and grouping semantically annotated product offers describing the same real-world product. The output of this pipeline was the WDC Product Corpus, comprising 26.5 million records of product offers, grouped in 16.3 million clusters representing the same real-world product. We profiled the WDC Product Corpus along different dimensions and showed that the offers of the corpus describe products of different categories and derive from more than 79 thousand websites. The distantly labeled set, derived from combining pairwise the intra- and inter-cluster offers, can be used for training entity resolution models targeting the product entity resolution task and is the largest, in terms of the number of matching pairs, and most

heterogeneous, in terms of amount of data sources, publicly available training set for entity resolution. Finally, we evaluated the cleanliness and training quality of the WDC Product Corpus. Based on a manually verified sample of pairs of offers, we evaluated the level of noise, i.e. intra-cluster offer pairs that refer to different real-world products, to be 6%. Using large amounts of product category-specific matching and non-matching offer pairs (> 40 thousand) for training baseline entity resolution models, we showed that F1 scores in the range of 89.2% to 94.4% can be achieved when applying the deep learning-based DeepMatcher model [Mudgal et al., 2018]. The F1 scores were calculated against a manually verified gold standard of matching and non-matching pairs of product offers with no identifiers. The relatively high cleanliness level of the clusters of the corpus, in combination with the high performance that can be achieved when training entity resolution models with training subsets derived from the corpus clusters, clearly demonstrated the utility of the Semantic Web as a valuable source of distant supervision for product-related entity resolution tasks.

10.2.2 Open Issues and Future Research

One limitation of the profiling of the adoption growth of semantic annotations is the omission of the crawling strategy applied by Common Crawl. Different crawling strategies have been shown to influence the profiling results of the corpora published by the Web Data Commons project [Stolz and Hepp, 2015]. Nonetheless, Meusel [2017] has shown that the Common Crawl data used by the Web Data Commons project are representative of the public Web. We relied on this finding for conducting our profiling analysis.

Regarding the analysis of the potential of using semantically annotated schema.org/LocalBusiness annotations as source of distant supervision for entity resolution, it shall be noted that the findings constitute an initial estimate. This is due to the fact that our analysis relied on markedup entities grouped together, considering solely the overlap of their annotated identifiers. As we showed for the semantically annotated product-related data, annotation errors commonly occur, and a series of cleansing steps is required to circumvent them. Therefore, an interesting direction for future work would be to develop cleansing pipelines tailored to business-related semantic annotations.

Finally, it is worth noting that for evaluating the training quality of distantly labeled sets derived from the WDC Product Corpus, we considered a closed-world entity resolution scenario. This means that the distantly labeled offer pairs used for training entity resolution models and the manually labeled offer pairs used for evaluation represent the same real-world products as they derive from the same subset of clusters of the WDC English Product Corpus. Therefore, an interesting direction for future research would be to investigate to which extent the training subsets derived from the WDC Product Corpus can be used in open-world entity resolution scenarios. The goal in such scenarios is to predict matching and non-matching relations between offers describing unseen product entities.

10.3 Part III: Active Learning for Entity Resolution

10.3.1 Summary and Contributions

In the third part of the thesis, we researched active learning methods as a means of reducing the labeling effort for entity resolution. Active learning has been widely applied in entity resolution tasks. Yet, we identified two gaps in related work, which we addressed in our research: first, the expensive initialization of the active learning workflow in terms of labeling effort, and second, the lack of active learning methods for multi-source entity resolution tasks.

Unsupervised Bootstrapping of Active Learning for Entity Resolution In the first chapter of this part, we addressed the first identified gap and developed an unsupervised method for bootstrapping, i.e. initializing and assisting, the complete active learning workflow. The proposed method relies on a thresholding heuristic that considers pre-calculated similarity scores and assigns labels with some degree of noise to the record pairs of the unlabeled pool. The noisy labels are used for initializing the active learning process and throughout the whole active learning cycle for the training of the learner and query selection. In comparison to existing approaches for initializing active learning, which rely on a randomly selected and manually labeled set [Meduri et al., 2020; Nafa et al., 2020; Qian et al., 2017], our method comes at no additional labeling effort. In comparison to initialization approaches that do not increase the labeling effort by randomly initializing the learner, which is typically a set of linkage rules evolving with genetic programming [Isele and Bizer, 2013; Ngomo and Lyko, 2012; Ngomo et al., 2013], our method uses a random forest classifier as a learner. Tree-based models, such as random forests, have been shown to perform better than rule-based models in active learning settings for entity resolution [Meduri et al., 2020].

We compared our proposed thresholding heuristic against commonly used thresholding methods on six entity resolution tasks of different profiling groups, as defined in Chapter 3. We identified that our thresholding heuristic achieves overall more stable results independently from the underlying similarity score distribution. We compared our unsupervised bootstrapped active learning method to symbolic active learning baselines, which are initialized either with a randomly selected set of matching and non-matching pairs or with transfer learning, and use the committee-based query strategy of HeALER [Chen et al., 2019]. Our experimental results showed that our method outperforms the symbolic baselines, which use random initialization by up to 48% in F1 score in the first active learning iterations. Within a labeling budget of 100 record pairs, our method outperforms the symbolic baselines by up to 3% when random sampling is used for initialization and up to 4.2% when transfer learning from non-highly related entity resolution tasks is applied for initializing active learning. Furthermore, we compared our method to the active learning method of Kasai et al. [2019], which uses transfer learning for initialization, an uncertainty-based query strategy, and a DeepMatcher model

as learner. Given that our unsupervised initialization method can be used for initializing any active learning method and is not bound to a specific query strategy or learner, we combined it with the active learning method of Kasai et al. [2019], thus designing the second subsymbolic active learning baseline. We showed that the subsymbolic baselines entail significantly larger waiting times within each active learning iteration in comparison to our symbolic method. Finally, with respect to the learner's performance, we showed that our method consistently outperforms the two subsymbolic baselines within an annotation budget of 500 record pairs by a minimum of 1.8% and a maximum of 32% in F1 score, depending on the task.

Multi-Source Active Learning For Entity Resolution In the second and third chapters of this part, we focused on active learning as a means of reducing the labeling effort for multi-source entity resolution tasks. Towards this research direction, we developed ALMSER, an active learning algorithm for multi-source entity resolution. ALMSER uses graph and grouping signals which are inherent to multi-source entity resolution tasks for query selection and training of the learner. ALMSER comes with the query strategies ALMSERgraph and ALMSERgroup, which to the best of our knowledge, are the first query strategies specially tailored to multi-source entity resolution tasks. We evaluated ALMSER with its two query strategies on five multi-source entity resolution tasks containing up to five data sources. We identified that after 200 iterations ALMSER reaches close to passive learning results, i.e. a maximum difference of 3.2 percentage points. Additionally, we compared ALMSER to two baseline active learning methods using a committee-based and a margin-based query strategy and no graph or grouping signals. The comparison results indicated that ALMSER consistently outperforms the two baselines in all tasks. Finally, by performing an ablation study to evaluate the distinct components of ALMSER, we observed that the positive contribution of graph and grouping signals to the performance of the learner when used only as part of the active learning query strategy is subject to the task at hand.

To further investigate this finding, we studied the impact of the profiling characteristics of multi-source entity resolution tasks on active learning methods exploiting different signals for query selection. Towards this goal, we proposed three profiling dimensions for describing multi-source entity resolution tasks. To enable our analysis, we developed ALMSERgen, the first multi-source entity resolution task generator, and curated 252 tasks. We evaluated three active learning methods using the HeALER, ALMSERgraph, and ALMSERgroup query strategies on the 252 generated tasks as well as on five benchmark tasks. By analyzing the results, we identified four patterns for explaining the contribution of graph and grouping signals with respect to the profile of the tasks.

10.3.2 Open Issues and Future Research

We identify two possible directions for future work concerning our work on unsupervised bootstrapping of active learning for entity resolution. First, clustering

methods such as the ones evaluated in the work of Saeedi et al. [2017] can be used for assigning labels on the pool record pairs in an unsupervised fashion and initializing active learning. To do so, two assumptions considered in the work of Saeedi et al. [2017] need to be surpassed: (i) The records derive from multiple, de-duplicated data sources. The de-duplication assumption and the number of data sources are key to the cleansing step of some of the applied clustering techniques. (ii) Manually defined and domain-specific matching rules are available for assigning the weighted edges between the records. The latter can be replaced by our domain-independent score aggregation function.

A second possible research direction is the adaptation of the deep active learning method of Kasai et al. [2019], which our method consistently outperforms given a small labeling budget, so that transformer-based models are used. Li et al. [2020] have shown that DITTO, a deep learning entity resolution system based on pre-trained transformer-based language models, outperforms DeepMatcher on benchmark tasks with less training data. Although DITTO has only been tested in a passive learning setting, its promising results, when trained with small training sets, show that it can be a good candidate for active learning settings as well.

With regard to our work on active learning for multi-source entity resolution, it must be highlighted that no scalability issues were considered. In our work, we assumed as computationally possible the blocking of record pairs between all pairwise data source combinations. Given a large number of data sources and limited resources, this can be prohibitive. To circumvent this, future work can define a matching execution plan prior to active learning. This can be achieved by applying related works on defining an efficient order of executing pairwise entity resolution tasks [Hertling and Paulheim, 2021; Shen et al., 2007].

Concerning the ALMSER query strategies, it should be noted that a single weighting scheme is applied to the record pairs of the unlabeled pool. The applied weighting scheme aims to balance the selection of likely false positives and likely false negatives for querying. Additionally, ALMSERgroup calculates the subset of most representative tasks and implicitly assigns a selection weight of zero to the record pairs that do not belong to this subset. However, in some entity resolution settings, it might be desired to favor the record pairs deriving from specific tasks over others, e.g. given their size. In this case, the weighting scheme needs to be adjusted given the desired goal, e.g. by assigning a larger weight to record pairs deriving from larger tasks.

An additional open issue, which motivates future work on active learning for multi-source entity resolution, evolves around the suggested profiling dimensions. The proposed profiling dimensions for multi-source entity resolution tasks rely on the ground truth, i.e. we need to know the labels of the record pairs in order to exactly calculate them. This was necessary for conducting our analysis and explaining the impact of the profile of multi-source entity resolution tasks on the performance of different active learning methods. However, it does not solve the problem of selecting an active learning method for solving a specific task. In this case, an unsupervised approximation of the profiling dimensions is required. For

example, entity overlap and value heterogeneity could be estimated using unsupervised entity resolution methods, while value pattern overlap could be estimated using the unsupervised metric of task relatedness [Thirumuruganathan et al., 2018].

10.4 Research Impact

In this final section of the thesis, we discuss the impact of our contributions to other published research works. Hereby, we split the discussion into three parts, similar to the structure of the thesis.

Research Impact of Part I: Entity Resolution - Profiling Entity Resolution Benchmark Tasks The dimensions for profiling entity resolution tasks, introduced in the first part of the thesis, help researchers to understand the specific challenges of each task, as well as the strengths and weaknesses of different entity resolution methods. Foxcroft et al. [2021] use the five dimensions proposed in our work, i.e. schema complexity, sparsity, textuality, development set size, and corner cases, to profile four product-related tasks and understand the strengths of different symbolic and subsymbolic methods. Three of the tasks used in their evaluation have already been introduced in Chapter 3, while the fourth task is proprietary and has a lower schema complexity, lower sparsity, and textuality in comparison to the three public tasks. In their experimental evaluation, the authors compare the results of four symbolic and five subsymbolic methods. Their results confirm our assumption that symbolic methods can achieve good performance on entity resolution tasks belonging to the profiling *Group 1: Dense Data, Simple Schema*, while subsymbolic methods are more performant for tasks with an increased level of textuality and amount of corner cases.

Graf et al. [2021] develop Frost, a benchmarking platform that offers entity resolution benchmark tasks, quality metrics, profiling dimensions, and visualization tools for systematically analyzing and interpreting the results of different entity resolution methods. Two of our profiling dimensions, namely sparsity and textuality, have been implemented as part of Frost. Additionally, the profiling dimension capturing the level of corner cases in a task is considered for the selection of benchmark tasks included in the benchmarking platform. Finally, Wang et al. [2021a] cite our work on profiling entity resolution tasks. In their work, they construct seven new entity resolution benchmark tasks by increasing the textuality and sparsity of existing tasks.

Research Impact of Part II: The Semantic Web as Distant Supervision for Entity Resolution Our work on profiling the adoption of Semantic Web annotations contributed to the continuation of the Web Data Commons project, initiated by Prof. Dr. Christian Bizer and Dr. Hannes Mühlheisen in 2012, for the period 2016-2021. The data and results published within the Web Data Commons Project have been referenced in multiple research works. The *RDFa, Microdata,*

and Microformat-Data Sets page[4] has been cited in 7 research papers, while our joint work with Dr. Robert Meusel on exploiting Microdata annotations for the specific downstream task of categorization [Meusel et al., 2015] has been cited 17 times. Furthermore, in the period 2016-2021, the Web Data Commons-RDFa, Microdata, and Microformat-Data Sets page was visited by more than 20 thousand unique users.[5]

The WDC Product Corpus for entity resolution, presented in Chapter 5, has been used in multiple research works and enabled the evaluation and comparison of deep learning-based entity resolution methods. Since its publication in 2019, our paper presenting the corpus [Primpeli et al., 2019] has been cited from 25 research papers. In 10 of those, it has been used as part of the experimental evaluation. Additionally, the training data extracted from the WDC Product Corpus were used as part of the Semantic Web Challenge on Mining the Web of HTML-embedded Product Data co-located with the International Semantic Web Conference in 2020 [Zhang et al., 2020b]. Finally, both the WDC Product corpus as well as our analysis on exploiting the Semantic Web as a source of supervision for entity resolution have been referenced in the exploratory analysis of Zhang and Song [2021] on utilizing the Web of Linked Data for the specific application of product data mining.

Research Impact of Part III: Active Learning for Entity Resolution Our work on unsupervised bootstrapping of active learning for entity resolution has been cited in 5 research works. In two of those, from Papadakis et al. [2021] and Agarwal et al. [2021], our work is mentioned for explicitly tackling the cold start problem, which occurs in active learning settings. Different from our method, Agarwal et al. [2021] tackle multi-class classification problems with active learning. The authors apply a clustering method and select the centroids of each cluster as seeding instances. The selected data points are manually inspected and labeled. Papadakis et al. [2021] provide an overview of different methods for initializing active learning, while our approach is referenced as a ”more principled approach that requires no human intervention“ in comparison to existing ones.

Finally, we are the first to develop an active learning method that exploits multi-source-related signals with the aim of reducing the labeling effort required for tackling efficiently multi-source entity resolution tasks. While Huang et al. [2018] have also proposed an active learning method for multi-source entity resolution, their approach relies on different workflow desiderata in comparison to our work. More specifically, in their work, the human annotators are considered to be error-prone. Additionally, the main focus of their work lies on improving the interaction with and user satisfaction of the human annotator.

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

[5] Information extracted using Google Data Analytics to measure the unique page views for the period 01.01.2016-31.12.2021

List of Figures

List of Tables

Bibliography

Abedjan, Z., Golab, L., and Naumann, F. (2015). Profiling relational data: A survey. *The VLDB Journal*, 24(4):557–581.

Achichi, M., Cheatham, M., Dragisic, Z., Euzenat, J., Faria, D., Ferrara, A., Flouris, G., Fundulaki, I., Harrow, I., Ivanova, V., Jiménez-Ruiz, E., Kolthoff, K., Kuss, E., Lambrix, P., Leopold, H., Li, H., Meilicke, C., Mohammadi, M., Montanelli, S., Pesquita, C., Saveta, T., Shvaiko, P., Splendiani, A., Stuckenschmidt, H., Thiéblin, E., Todorov, K., Trojahn, C., and Zamazal, O. (2017). Results of the ontology alignment evaluation initiative 2017. In *Proceedings of the 12th International Workshop on Ontology Matching*, OM '17, pages 61–113, Vienna, Austria. CEUR-WS.org.

Adelberg, B. (1998). NoDoSE - a tool for semi-automatically extracting structured and semistructured data from text documents. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 283–294, New York, NY, United States. ACM.

Agarwal, A., Garg, R., and Chaudhury, S. (2013). Greedy search for active learning of OCR. In *Proceedings of the 12th International Conference on Document Analysis and Recognition*, ICDAR '13, pages 837–841, Washington, DC, USA. IEEE Computer Society.

Agarwal, D., Srivastava, P., Martin-del Campo, S., Natarajan, B., and Srinivasan, B. (2021). Addressing practical challenges in active learning via a hybrid query strategy. *arXiv:2110.03785*.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, United States. Morgan Kaufmann.

Angluin, D. (1988). Queries and concept learning. *Machine learning*, 2(4):319–342.

Anthony, M. H. G. and Biggs, N. (1997). *Computational learning theory*. Cambridge University Press.

Arasu, A., Götz, M., and Kaushik, R. (2010). On active learning of record matching packages. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 783–794, New York, NY, United States. ACM.

Barlaug, N. and Gulla, J. A. (2021). Neural networks for entity matching: A survey. *ACM Transactions on Knowledge Discovery from Data*, 15(3):1–37.

Baum, E. B. and Lang, K. (1992). Query learning can work poorly when a human oracle is used. In *Proceedings of the International Joint Conference on Neural Networks*, IJCNN '92, pages 335–340, Washington, DC, United States. IEEE Computer Society.

Bechtold, H. (2019). Multi-class classification of e-commerce product offers. Master's thesis, University of Mannheim, Mannheim, Germany.

Bellare, K., Curino, C., Machanavajihala, A., Mika, P., Rahurkar, M., and Sane, A. (2013). Woo: A scalable and multi-tenant platform for continuous knowledge base synthesis. *Proceedings of the VLDB Endowment*, 6(11):1114–1125.

Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. (2018). The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '18, pages 9368–9377, Washington, DC, United States. IEEE Computer Society.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Berenzweig, A., Logan, B., Ellis, D. P., and Whitman, B. (2004). A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, pages 63–76.

Berners-Lee, T. (2009). Semantic Web and Linked Data (2009). http://www.w3.org/2009/Talks/0204-campus-party-tbl/.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):34–43.

Bilenko, M., Kamath, B., and Mooney, R. J. (2006). Adaptive blocking: Learning to scale up record linkage. In *Proceedings of the 6th International Conference on Data Mining*, ICDM '06, pages 87–96, Los Alamitos, CA, United States. IEEE Computer Society.

Bilenko, M. and Mooney, R. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 39–48, New York, NY, United States. ACM.

Bilke, A. and Naumann, F. (2005). Schema matching using duplicates. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 69–80, Washington, DC, United States. IEEE Computer Society.

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, J. and Naumann, F. (2009). Data fusion. *ACM Computing Surveys*, 41(1):1–41.

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*, ICDE '21, pages 1272–1283, Los Alamitos, CA, United States. IEEE Computer Society.

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.

Borkar, V., Deshmukh, K., and Sarawagi, S. (2001). Automatic segmentation of text into structured records. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD/PODS '01, pages 175–186, New York, NY, United States. ACM.

Bouguelia, M.-R., Belaïd, Y., and Belaïd, A. (2015). Identifying and mitigating labelling errors in active learning. In *Proceedings of the 4th International Conference on Pattern Recognition Applications and Methods*, ICPRAM '15, pages 35–51, Setubal, Portugal. SCITEPRESS.

Bouguelia, M.-R., Nowaczyk, S., Santosh, K., and Verikas, A. (2018). Agreeing to disagree: Active learning with noisy labels without crowdsourcing. *International Journal of Machine Learning and Cybernetics*, 9(8):1307–1319.

Brinkmann, A. and Bizer, C. (2021). Improving hierarchical product classification using domain-specific language modelling. *Bulletin of the Technical Committee on Data Engineering*, 44(2):14–25.

Brunner, U. and Stockinger, K. (2019). Entity matching on unstructured data: an active learning approach. In *Proceedings of the 6th Swiss Conference on Data Science*, SDS '19, pages 97–102, Los Alamitos, CA, United States. IEEE Computer Society.

Brunner, U. and Stockinger, K. (2020). Entity matching with transformer architectures-a step forward in data integration. In *Proceedings of 23rd International Conference on Extending Database Technology*, EDBT '20, pages 463–473. OpenProceedings.org.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

Caruana, R., Karampatziakis, N., and Yessenalina, A. (2008). An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine learning*, ICML '08, pages 96–103, New York, NY, United States. ACM.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, United States. ACM.

Chen, X., Xu, Y., Broneske, D., Durand, G. C., Zoun, R., and Saake, G. (2019). Heterogeneous committee-based active learning for entity resolution (HeALER). In *Proceedings of the 23rd European Conference on Advances in Databases and Information Systems*, ADBIS '19, Cham, Switzerland. Springer.

Christen, P. (2008). Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 151–159, New York, NY, United States. ACM.

Christen, P. (2009). Development and user experiences of an open source data cleaning, deduplication and record linkage system. *ACM SIGKDD Explorations Newsletter*, 11(1):39–48.

Christen, P. (2012). *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Data-centric systems and applications. Springer, Heidelberg Germany.

Christen, P. and Goiser, K. (2007). Quality and complexity measures for data linkage and deduplication. In *Quality measures in data mining*, Studies in Computational Intelligence, pages 127–151. Springer, Heidelberg, Germany.

Christen, P., Vatsalan, D., and Wang, Q. (2015). Efficient entity resolution with adaptive and interactive training data selection. In *2015 IEEE International Conference on Data Mining*, ICDM '15, pages 727–732, Washington, DC, United States. IEEE Computer Society.

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):1–42.

Christophides, V., Efthymiou, V., and Stefanidis, K. (2015). Entity resolution in the web of data. *Synthesis Lectures on the Semantic Web*, 5(3):1–122.

Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web*, IIWEB '03, pages 73–78, Palo Alto, CA, United States. AAAI Press.

Cohn, D., Atlas, L., and Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

Collins, M. and Singer, Y. (1999). Unsupervised models for named entity classification. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, EMNLP '99, New York, NY, United States. ACM.

Crescenzi, V., De Angelis, A., Firmani, D., Mazzei, M., Merialdo, P., Piai, F., and Srivastava, D. (2021). Alaska: A flexible benchmark for data integration tasks. *arXiv:2101.11259*.

Crescenzi, V., Mecca, G., and Merialdo, P. (2001). Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 109–118, San Francisco, CA, United States. Morgan Kaufmann.

Dagan, I. and Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. In *Proceedings of the 12th International Conference on Machine Learning*, ICML '95, pages 150–157. Morgan Kaufmann, San Francisco, CA, United States.

Dantzig, G. and Fulkerson, D. (1956). On the max-flow min-cut theorem of networks. In *Linear inequalities and related systems*, Annals of Mathematic Studies. Princeton University Press, Princeton, NJ, United States.

Daskalaki, E., Flouris, G., Fundulaki, I., and Saveta, T. (2016). Instance matching benchmarks in the era of Linked Data. *Journal of Web Semantics*, 39:1 – 14.

de Freitas, J., Pappa, G. L., da Silva, A. S., Gonc¸alves, M. A., Moura, E., Veloso, A., Laender, A. H., and de Carvalho, M. G. (2010). Active learning genetic programming for record deduplication. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, pages 1–8, Washington, DC, United States. IEEE Computer Society.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). 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*, NAACL '19, Stroudsburg, PA, USA. ACL.

Doan, A., Halevy, A., and Ives, Z. (2012). *Principles of data integration*. Morgan Kaufmann, San Francisco, CA, United States.

Dong, X. L. and Srivastava, D. (2015). *Big data integration*. Morgan & Claypool.

Dorneles, C. F., Gonçalves, R., and dos Santos Mello, R. (2011). Approximate data instance matching: a survey. *Knowledge and Information Systems*, 27(1):1–21.

Draisbach, U. and Naumann, F. (2010). DuDe: The duplicate detection toolkit. In *Proceedings of the 8th International Workshop on Quality in Databases*, QDB '10, New York, NY, United States. ACM.

Dunn, H. L. (1946). Record linkage. *American Journal of Public Health and the Nations Health*, 36(12):1412–1416.

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, M. G., Verykios, V. S., and Elmagarmid, A. K. (2002). Tailor: A record linkage toolbox. In *Proceedings 18th International Conference on Data Engineering*, ICDE '02, pages 17–28, Los Alamitos, CA, United States. IEEE Computer Society.

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.

Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. (2004). Web-scale information extraction in knowitall. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, United States. ACM.

Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210.

Ferrara, A., Montanelli, S., Noessner, J., and Stuckenschmidt, H. (2011). Benchmarking matching applications on the Semantic Web. In *Proceedings of the 8th Extended Semantic Web Conference*, ESWC '11, pages 108–122, Heidelberg, Germany. Springer.

Foley, J., Bendersky, M., and Josifovski, V. (2015). Learning to extract local events from the web. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 423–432, New York, NY, United States. ACM.

Foxcroft, J., Chen, T., Padmanabhan, K., Keng, B., and Antonie, L. (2021). Product matching lessons and recommendations from a real world application. In *Proceedings of the 34th Canadian Conference on Artificial Intelligence*. Pubpub.org.

Fu, C., Han, X., Sun, L., Chen, B., Zhang, W., Wu, S., and Kong, H. (2019). End-to-end multi-perspective matching for entity resolution. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI '19, Palo Alto, CA, United States. AAAI Press.

Fujii, A., Tokunaga, T., Inui, K., and Tanaka, H. (1998). Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, 24(4):573–597.

Getoor, L. and Machanavajjhala, A. (2012). Entity resolution: Theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019.

Gokhale, C., Das, S., Doan, A., Naughton, J. F., Rampalli, N., Shavlik, J., and Zhu, X. (2014). Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 601–612, New York, NY, United States. ACM.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Graf, M., Laskowski, L., Papsdorf, F., Sold, F., Gremmelspacher, R., Naumann, F., and Panse, F. (2021). Frost: Benchmarking and exploring data matching results. *arXiv:2107.10590*.

Guha, R. V., Brickley, D., and Macbeth, S. (2015). Schema. org: evolution of structured data on the web. *ACM Queue*, 59(2):44–51.

Halevy, A., Rajaraman, A., and Ordille, J. (2006). Data integration: The teenage years. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, VLDB '06, pages 9–16. VLDB Endowment.

Halevy, A. Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., and Sikka, V. (2005). Enterprise information integration: successes, challenges and controversies. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 778–787.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Morgan Kaufmann, San Francisco, CA, United States.

Haugeland, J. (1989). *Artificial intelligence: The very idea*. MIT press.

Hernández, M. A. and Stolfo, S. J. (1998). Real-world data is dirty: Data cleansing and the merge/purge problem. *Data mining and knowledge discovery*, 2(1):9–37.

Hertling, S. and Paulheim, H. (2021). Order matters: Matching multiple knowledge graphs. In *Proceedings of the 11th International Conference on Knowledge Capture Conference*, K-CAP '21, pages 113–120, New York, NY, United States. ACM.

Herzog, T. N., Scheuren, F. J., and Winkler, W. E. (2007). *Data quality and record linkage techniques*. Springer Science & Business Media.

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.

Hoi, S. C., Jin, R., and Lyu, M. R. (2006). Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 633–642, New York, NY, United States. ACM.

Hou, B., Chen, Q., Wang, Y., Nafa, Y., and Li, Z. (2019). Gradual machine learning for entity resolution. In *Proceedings of the World Wide Web Conference 2019*, WWW '19, pages 3526–3530, New York, NY, United States. ACM.

Huang, J., Hu, W., Li, H., and Qu, Y. (2018). Automated comparative table generation for facilitating human intervention in multi-entity resolution. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 585–594, New York, NY, United States. ACM.

Ioannou, E., Rassadko, N., and Velegrakis, Y. (2013). On generating benchmark data for entity matching. *Journal on Data Semantics*, 2(1):37–56.

Ipeirotis, P. G., Provost, F., Sheng, V. S., and Wang, J. (2014). Repeated labeling using multiple noisy labelers. *Data Mining and Knowledge Discovery*, 28(2):402–441.

Isele, R. and Bizer, C. (2012). Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, 5(11):1638–1649.

Isele, R. and Bizer, C. (2013). Active learning of expressive linkage rules using genetic programming. *Web Semantics*, 23:2–15.

Jain, A., Sarawagi, S., and Sen, P. (2021). Deep indexed active learning for matching heterogeneous entity representations. *Proceedings of the VLDB Endowment*, 15(1):31–45.

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.

Jaro, M. A. (1995). Probabilistic linkage of large public health data files. *Statistics in medicine*, 14(5-7):491–498.

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):465–477.

Johnson, T. (2009). Data profiling. In *Encyclopedia of Database Systems*. Springer, Boston, MA, United States.

Kaisler, S., Armour, F., Espinosa, J. A., and Money, W. (2013). Big data: Issues and challenges moving forward. In *Proceedings of the 46th Hawaii International*

Conference on System Sciences, HICSS '13, pages 995–1004, Los Alamitos, CA, United States. IEEE Computer Society.

Kannan, A., Givoni, I. E., Agrawal, R., and Fuxman, A. (2011). Matching unstructured product offers to structured product specifications. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 404–412, New York, NY, United States. ACM.

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*, ACL '19, Stroudsburg, PA, USA. ACL.

Kejriwal, M. and Miranker, D. P. (2015). An unsupervised instance matcher for schema-free RDF data. *Web Semantics*, 35:102–123.

Khalid, S., Khalil, T., and Nasreen, S. (2014). A survey of feature selection and feature extraction techniques in machine learning. In *Proceedings of the 2014 Science and Information Conference*, SAI '14, pages 372–378, Queens, NY, United States. The Science and Information Organization.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations*, ICLR '14.

Klyne, G. and Carroll, J. J. (2004). Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation. http://www.w3.org/TR/rdf-concepts/.

Konda, P., Das, S., C., P. S. G., Doan, A., Ardalan, A., Ballard, J. R., Li, H., Panahi, F., Zhang, H., Naughton, J., Prasad, S., Krishnan, G., Deep, R., and Raghavendra, V. (2016). Magellan: Toward building entity matching management systems over data science stacks. *Proceedings of the VLDB Endowment*, 9(13):1581–1584.

Konyushkova, K., Sznitman, R., and Fua, P. (2017). Learning active learning from data. In *Proceedings of the 31st Annual Conference on Advances in Neural Information Processing Systems*, NIPS '17, pages 4228–4238, Red Hook, NY, United States. Curran Associates.

Kooli, N., Allesiardo, R., and Pigneul, E. (2018). Deep learning based approach for entity resolution in databases. In *Proceedings of the 10th Asian Conference on Intelligent Information and Database Systems*, ACIIDS '18, pages 3–12, Cham, Switzerland. Springer.

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):484–493.

Koudas, N., Sarawagi, S., and Srivastava, D. (2006). Record linkage: similarity measures and algorithms. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD/PODS '06, pages 802–803, New York, NY, United States. ACM.

Koumarelas, I., Kroschk, A., Mosley, C., and Naumann, F. (2018). Experience: Enhancing address matching with geocoding and similarity measure selection. *Journal of Data and Information Quality*, 10(2):1–16.

Köpcke, H. and Rahm, E. (2008). Training selection for tuning entity matching. In *Proceedings of the International Workshop on Quality in Databases and Management of Uncertain Data*, QDB/MUD, pages 3–12, New York, NY, United States. ACM.

Laney, D. (2001). 3D data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6(70):1.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

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.

Lim, E.-P., Srivastava, J., Prabhakar, S., and Richardson, J. (1996). Entity identification in database integration. *Information Sciences*, 89(1-2):1–38.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692*.

McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 43–52, New York, NY, United States. ACM.

McCallum, A., Nigam, K., and Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 169–178, New York, NY, United States. ACM.

McFee, B., Bertin-Mahieux, T., Ellis, D. P., and Lanckriet, G. R. (2012). The million song dataset challenge. In *Companion Proceedings of the 21st Annual Conference on World Wide Web*, WWW '12 Companion, pages 909–916, New York, NY, United States. ACM.

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 SIGMOD International Conference on Management of Data, SIGMOD/PODS '20, pages 1133–1147, New York, NY, United States. ACM.

Meusel, R. (2017). *Web-scale profiling of semantic annotations in HTML pages*. PhD thesis, University of Mannheim, Mannheim, Germany.

Meusel, R. and Paulheim, H. (2014). Linked data for information extraction challenge 2014 tasks and results. In *Proceedings of the 2nd International Conference on Linked Data for Information Extraction*, pages 3–8, Aachen, Germany. CEUR-WS.org.

Meusel, R. and Paulheim, H. (2015a). Creating large-scale training and test corpora for extracting structured data from the web. In *Proceedings of the 3rd International Conference on Linked Data for Information Extraction*, pages 2–6, Aachen, Germany. CEUR-WS.org.

Meusel, R. and Paulheim, H. (2015b). Heuristics for fixing common errors in deployed schema. org microdata. In *Proceedings of the 9th European Semantic Web Conference*, ESWC '12, pages 152–168, Cham, Switzerland. Springer.

Meusel, R., Primpeli, A., Meilicke, C., Paulheim, H., and Bizer, C. (2015). Exploiting microdata annotations to consistently categorize product offers at web scale. In *Proceedings of the 16th International Conference on Electronic Commerce and Web Technologies*, EC-Web '15, pages 83–99, Cham, Switzerland. Springer.

Michelson, M. and Knoblock, C. A. (2006). Learning blocking schemes for record linkage. In *Proceedings of the 21st national conference on Artificial intelligence*, AAAI '06, pages 440–445, Palo Alto, CA, United States. AAAI Press.

Mika, P. (2015). On schema.org and why it matters for the web. *IEEE Internet Computing*, 19(4):52–55.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, Red Hook, NY, United States. Curran Associates.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ACL/IJCNLP '09, pages 1003–1011, Stroudsburg, PA, United States. ACL.

Mitchell, H. B. (2010). Image Similarity Measures. In *Image Fusion: Theories, Techniques and Applications*, pages 167–185. Springer Berlin Heidelberg, Berlin, Heidelberg.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. (2018). Never-ending learning. *Communications of the ACM*, 61(5):103–115.

Monge, A. E. and Elkan, C. (1996). The field matching problem: Algorithms and applications. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, KDD '96, pages 267–270, Palo Alto, CA, United States. AAAI Press.

Mozafari, B., Sarkar, P., Franklin, M., Jordan, M., and Madden, S. (2014). Scaling up crowd-sourcing to very large datasets: A case for active learning. *Proceedings of the VLDB Endowment*, 8(2):125–136.

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. (2018). Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 19–34, New York, NY, USA. ACM.

Nafa, Y., Chen, Q., Chen, Z., Lu, X., He, H., Duan, T., and Li, Z. (2020). Active deep learning on entity resolution by risk sampling. *arXiv:2012.12960*.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. (2013). Learning with noisy labels. *Advances in Neural Information Processing Systems*, 26.

Naumann, F. (2014). Data profiling revisited. *ACM SIGMOD Record*, 42(4):40–49.

Naumann, F. and Herschel, M. (2010). An introduction to duplicate detection. *Synthesis Lectures on Data Management*, 2(1):1–87.

Neumaier, S., Savenkov, V., and Vakulenko, S. (2017). Talking open data. In *Proceedings of the 14th Extended Semantic Web Conference*, ESWC '17, pages 132–136, Cham, Switzerland. Springer.

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.

Ng, H.-F. (2006). Automatic thresholding for defect detection. *Pattern Recognition Letters*, 27(14):1644–1649.

Ngomo, A.-C. N., Lehmann, J., Auer, S., and Höffner, K. (2011). Raven–active learning of link specifications. In *Proceedings of the 6th International Conference on Ontology Matching*, OM '11, pages 25–36, Aachen, Germany. CEUR-WS.org.

Ngomo, A.-C. N. and Lyko, K. (2012). Eagle: Efficient active learning of link specifications using genetic programming. In *Proceedings of the 9th Extended Semantic Web Conference*, ESWC '12, pages 149–163. Springer, Heidelberg, Germany.

Ngomo, A.-C. N., Lyko, K., and Christen, V. (2013). Coala–correlation-aware active learning of link specifications. In *Proceedings of the 10th Extended Semantic Web Conference*, ESWC '13, pages 442–456. Springer, Heidelberg, Germany.

Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 625–632, New York, NY, United States. ACM.

Nilsson, N. J. (1998). *Artificial intelligence: a new synthesis*. Morgan Kaufmann, San Francisco, CA, United States.

Olsson, F. (2009). A literature survey of active machine learning in the context of natural language processing. Technical report, Swedish Institute of Computer Science.

Olsson, F. and Tomanek, K. (2009). An intrinsic stopping criterion for committee-based active learning. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 138–146, Stroudsburg, PA, United States. ACL.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

Oulabi, Y. and Bizer, C. (2019). Using weak supervision to identify long-tail entities for knowledge base completion. In *Semantic systems : The power of AI and knowledge graphs in the Proceedings of the 15th International Conference SEMANTiCS*, SEMANTiCS '19, pages 83–98, Cham, Switzerland. Springer.

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, Cham, Switzerland.

Papadakis, G., Alexiou, G., Papastefanatos, G., and Koutrika, G. (2015). Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data. *Proceedings of the VLDB Endowment*, 9(4):312–323.

Papadakis, G., Ioannou, E., Thanos, E., and Palpanas, T. (2021). The four generations of entity resolution. *Synthesis Lectures on Data Management*, 16(2):1–170.

Papadakis, G., Skoutas, D., Thanos, E., and Palpanas, T. (2019). Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys*, 53(2):1–42.

Papadakis, G., Tsekouras, L., Thanos, E., Giannakopoulos, G., Palpanas, T., and Koubarakis, M. (2020). Domain- and structure-agnostic end-to-end entity resolution with JedAI. *ACM SIGMOD Record*, 48(4):30–36.

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, R. and Bizer, C. (2022). Supervised contrastive learning for product matching. In *Companion Proceedings of the World Wide Web Conference 2022*, WWW '22 Companion. (to appear).

Peeters, R., Bizer, C., and Glavaš, G. (2020a). Intermediate training of BERT for product matching. In *Proceedings of the 2nd International Workshop on Challenges and Experiences from Data Integration to Knowledge Graphs*, DI2KG '20, pages 1–2, Aachen, Germany. CEUR-WS.org.

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*, WIMS '20, pages 195–204, New York, NY, United States. ACM.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1532–1543, Stroudsburg, PA, United States. ACL.

Petrovski, P. and Bizer, C. (2017). Extracting attribute-value pairs from product specifications on the web. In *Proceedings of the 2017 International Conference on Web Intelligence*, WI '17, pages 558–565, New York, NY, United States. ACM.

Petrovski, P. and Bizer, C. (2020). Learning expressive linkage rules from sparse data. *Semantic Web*, (11):549–567.

Petrovski, P., Bryl, V., and Bizer, C. (2014). Integrating product data from websites offering microdata markup. In *Companion Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 1299–1304, New York, NY, United States. ACM.

Phua, C., Smith-Miles, K., Lee, V., and Gayler, R. (2010). Resilient identity crime detection. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):533–546.

Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Porter, E. H. and Winkler, W. E. (1997). Approximate string comparison and its effect on an advanced record linkage system. Technical report, US Bureau of the Census.

Primpeli, A. and Bizer, C. (2019). Robust active learning of expressive linkage rules. In *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics*, WIMS '19, pages 1–7, New York, NY, United States. ACM.

Primpeli, A. and Bizer, C. (2020). Profiling entity matching benchmark tasks. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, CIKM '20, pages 3101–3108, New York, NY, United States. ACM.

Primpeli, A. and Bizer, C. (2021). Graph-boosted active learning for multi-source entity resolution. In *Proceedings of the 25th International Semantic Web Conference*, ISWC '21, pages 182–199, Cham, Switzerland. Springer.

Primpeli, A. and Bizer, C. (2022). Impact of the characteristics of multi-source entity matching tasks on the performance of active learning methods. In *Proceedings of the 19th Extended Semantic Web Conference*, ESWC '22, pages 113–129, Cham, Switzerland. Springer.

Primpeli, A., Bizer, C., and Keuper, M. (2020). Unsupervised bootstrapping of active learning for entity resolution. In *Proceedings of the 17th Extended Semantic Web Conference*, ESWC '20, pages 215–231, Cham, Switzerland. Springer.

Primpeli, A., Meusel, R., Bizer, C., and Stuckenschmidt, H. (2017). The Web Data Commons structured data extraction. In *E-Science-Tage 2017: Forschungsdaten managen*, page 1, Heidelberg. Heidelberg University.

Primpeli, A., Peeters, R., and Bizer, C. (2019). The WDC training dataset and gold standard for large-scale product matching. In *Companion Proceedings of the World Wide Web Conference 2019*, WWW '19 Companion, pages 381–386, New York, NY, United States. ACM.

Prud'hommeaux, E. and Seaborne, A. (2008). Sparql query language for RDF. W3C recommendation. https://www.w3.org/TR/rdf-sparql-query/.

Qian, K., Popa, L., and Sen, P. (2017). Active learning for large-scale entity resolution. In *Proceedings of the 31st on Conference on Information and Knowledge Management*, CIKM '17, pages 1379–1388, New York, NY, United States. ACM.

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 VLDB Endowment*, 8(13):2194–2205.

Rahm, E. and Do, H. H. (2000). Data cleaning: Problems and current approaches. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 23(4):3–13.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. (2021). A survey of deep active learning. *ACM Computing Surveys*, 54(9):1–40.

Ristoski, P. and Mika, P. (2016). Enriching product ads with metadata from HTML annotations. In *Proceedings of the 13th Extended Semantic Web Conference*, ESWC '16, pages 151–167, Cham, Switzerland. Springer.

Roh, Y., Heo, G., and Whang, S. E. (2019). A survey on data collection for machine learning: A Big Data-AI integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1328–1347.

Saeedi, A., Peukert, E., and Rahm, E. (2017). Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In *Proceedings of the 21st European Conference on Advances in Databases and Information Systems*, ADBIS '17, pages 278–293, Cham, Switzerland. Springer.

Saeedi, A., Peukert, E., and Rahm, E. (2018). Using link features for entity clustering in knowledge graphs. In *Proceedings of the 15th Extended Semantic Web Conference*, ESWC '18, pages 576–592, Cham, Switzerland. Springer.

Saeedi, A., Peukert, E., and Rahm, E. (2020). Incremental multi-source entity resolution for knowledge graph completion. In *Proceedings of the 17th Extended Semantic Web Conference*, ESWC '20, pages 393–408, Cham, Switzerland. Springer.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv:1910.01108*.

Sarawagi, S. (2008). *Information extraction*. Now Publishers, Norwell, MA, United States.

Sarawagi, S. and Bhamidipaty, A. (2002). Interactive deduplication using active learning. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 269–278, New York, NY, United States. ACM.

Satopaa, V., Albrecht, J., Irwin, D., and Raghavan, B. (2011). Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *Proceedings*

of the 31st International Conference on Distributed Computing Systems Workshops, ICDCSW '11, pages 166–171, Los Alamitos, CA, United States. IEEE Computer Society.

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*, ISWC '15, pages 375–391, Cham, Switzerland. Springer.

Settles, B. (2012). Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Sezgin, M. and Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–165.

Shao, J., Wang, Q., and Lin, Y. (2019). Skyblocking for entity resolution. *Information Systems*, 85:30–43.

Shen, D., Zhang, J., Su, J., Zhou, G., and Tan, C. L. (2004). Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, ACL '04, pages 589–596, Stroudsburg, PA, USA. ACL.

Shen, W., DeRose, P., Vu, L., Doan, A., and Ramakrishnan, R. (2007). Source-aware entity matching: A compositional approach. In *Proceedings of the 23rd International Conference on Data Engineering*, ICDE '07, pages 196–205, Los Alamitos, CA, United States. IEEE Computer Society.

Sherif, M. A., Dreßler, K., and Ngomo, A.-C. N. (2020). LIGON-link discovery with noisy oracles. In *Proceedings of the 15th ISWC workshop on Ontology Matching*, OM '20, pages 48–59, Aachen, Germany. CEUR-WS.org.

Song, H., Kim, M., Park, D., Shin, Y., and Lee, J.-G. (2022). Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Steorts, R. C., Ventura, S. L., Sadinle, M., and Fienberg, S. E. (2014). A comparison of blocking methods for record linkage. In *Proceedings of the 2014 International Conference on Privacy in Statistical Databases*, PSD '14, pages 253–268, Cham, Switzerland. Springer.

Stolz, A. and Hepp, M. (2015). Towards crawling the web for structured data: Pitfalls of common crawl for e-commerce. In *Proceedings of the 6th International Workshop on Consuming Linked Data*, COLD '15, Aachen, Germany. CEUR-ws.org.

Tao, Y. (2018). Entity matching with active monotone classification. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, SIGMOD/PODS '18, pages 49–62, New York, NY, United States. ACM.

Tejada, S., Knoblock, C. A., and Minton, S. (2001). Learning object identification rules for information integration. *Information Systems*, 26(8):607–633.

Thirumuruganathan, S., Li, H., Tang, N., Ouzzani, M., Govind, Y., Paulsen, D., Fung, G., and Doan, A. (2021). Deep learning for blocking in entity matching: A design space exploration. *Proceedings of the VLDB Endowment*, 14(11):2459–2472.

Thirumuruganathan, S., Parambath, S. A. P., Ouzzani, M., Tang, N., and Joty, S. (2018). Reuse and adaptation for entity resolution through transfer learning. *arXiv:1809.11084*.

Tran, K.-N., Vatsalan, D., and Christen, P. (2013). Geco: An online personal data generator and corruptor. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM '13, pages 2473–2476, New York, NY, United States. ACM.

Tu, J., Fan, J., Tang, N., Wang, P., Chai, C., Li, G., Fan, R., and Du, X. (2022). Domain adaptation for deep entity resolution. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD/PODS '22, New York, NY, United States. ACM.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS '17, pages 5998–6008, Red Hook, NY, United States. Curran Associates.

Vlachos, A. (2008). A stopping criterion for active learning. *Computer Speech and Language*, 22(3):295–312.

Waitz, S. (2021). Combining deep learning and active learning for entity resolution. Master's thesis, University of Mannheim, Mannheim, Germany.

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, J., Li, Y., and Hirota, W. (2021a). Machamp: A generalized entity matching benchmark. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, CIKM '21, pages 4633–4642, New York, NY, United States. ACM.

Wang, P., Zheng, W., Wang, J., and Pei, J. (2021b). Automating entity matching model development. In *Proceedings of the 37th International Conference*

on Data Engineering, ICDE '21, pages 1296–1307, Los Alamitos, CA, United States. IEEE Computer Society.

Wang, Z., Sisman, B., Wei, H., Dong, X. L., and Ji, S. (2020). Cordel: A contrastive deep learning approach for entity linkage. In *Proceedings of the 20th International Conference on Data Mining*, ICDM '20, pages 1322–1327, Los Alamitos, CA, United States. IEEE Computer Society.

Wilke, M. and Rahm, E. (2021). Towards multi-modal entity resolution for product matching. In *Proceedings of the 32rd GI Workshop on Foundations of Database Systems*, GVDB '21, Aachen, Germany. CEUR-WS.org.

Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods*, pages 354–359, London, England. American Statistical Association.

Wolcott, L., Clements, W., and Saripalli, P. (2018). Scalable record linkage. In *Proceedings of the 2018 IEEE International Conference on Big Data*, Big Data '18, pages 4268–4275, Los Alamitos, CA, United States. IEEE Computer Society.

Wu, R., Chaba, S., Sawlani, S., Chu, X., and Thirumuruganathan, S. (2020). Zeroer: Entity resolution using zero labeled examples. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD-/PODS '20, pages 1149–1164, New York, NY, United States. ACM.

Zadrozny, B. and Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, pages 609–616, San FranciscoCAUnited States. Morgan Kaufmann Publishers Inc.

Zhang, C. and Chen, T. (2002). An active learning framework for content-based information retrieval. *IEEE transactions on multimedia*, 4(2):260–268.

Zhang, W., Wei, H., Sisman, B., Dong, X. L., Faloutsos, C., and Page, D. (2020a). Autoblock: A hands-off blocking framework for entity matching. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, pages 744–752, New York, NY, United States. ACM.

Zhang, X.-Y., Wang, S., and Yun, X. (2015). Bidirectional active learning: A two-way exploration into unlabeled and labeled data set. *IEEE Transactions on Neural Networks and Learning Systems*, 26(12):3034–3044.

Zhang, Z., Bizer, C., Peeters, R., and Primpeli, A. (2020b). MWPD2020: Semantic web challenge on mining the web of HTML-embedded product data. In *Proceedings of the Semantic Web Challenge on Mining the Web of HTML-embedded Product Data co-located with the 19th International Semantic Web Conference*, ISWC '20, Aachen, Germany. CEUR-WS.org.

Zhang, Z. and Song, X. (2021). An exploratory study on utilising the web of linked data for product data mining. *arXiv:2109.01411*.

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 World Wide Web Conference 2019*, WWW '19, pages 2413–2424, New York, NY, United States. ACM.

Zhuang, Y., Li, G., Xue, W., and Zhu, F. (2020). An active learning based hybrid neural network for joint information extraction. In *International Conference on Web Information Systems Engineering*, WISE '20, pages 84–100, Cham, Switzerland. Springer.