AUTOMATED KNOWLEDGE BASE EXTENSION USING OPEN INFORMATION

by

ARNAB KUMAR DUTTA

DURGAPUR, INDIA

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
Fakultät für Wirtschaftsinformatik und Wirtschaftsmathematik
der Universität Mannheim

Mannheim, Deutschland, 2015

Dekan: Prof. Dr. Heinz Jürgen Müller, Universität Mannheim, Deutschland

Referent: Prof. Dr. Heiner Stuckenschmidt, Universität Mannheim, Deutschland

Korreferent: Ao. Prof. Dr. Fabian Suchanek, Télécom ParisTech University, France

Tag der mündlichen Prüfung: 4 Februar 2016

Your work is going to fill a large part of your life,
and the only way to be truly satisfied is to do what you believe is great work.
And the only way to do great work is to love what you do.
If you haven't found it yet, keep looking.
Don't settle. As with all matters of the heart, you'll know when you find it.
And, like any great relationship, it just gets better and better as the years roll on.
So keep looking until you find it. Don't settle.

- Steve Jobs (1955 - 2011)

ABSTRACT

Open Information Extractions (OIE) (like NELL, REVERB) frameworks provide us with domain independent facts in natural language forms containing knowledge from varied sources. Extraction mechanisms for structured knowledge bases (KB) (like DBPEDIA, YAGO) often fail to retrieve such facts due to its resource specific extraction schemes. Hence, the structured KBs can extend themselves by augmenting their coverage with the facts discovered by OIE systems. This possibility motivates us to integrate these two genres of extractions into one interactive framework. In this work, we present a complete, ontology independent, generalized architecture for achieving this integration. Our proposed solution is modularized which solves a specific set of tasks: (1) mapping subject and object terms from OIE facts to KB instances (2) mapping the OIE relational phrases to object properties defined in the KB. Furthermore, in an open extraction setting identical semantic relationships can be represented by different surface forms, making it necessary to group them together. To solve this problem, (3) we propose the use of markov clustering to cluster OIE relations. Key to our approach lies in exploiting the inherent dependancies between relations and its arguments. This makes our approach completely context agnostic and generally applicable. We evaluated our method on the two state of the art extraction systems, achieving over 85% precision on instance mappings and over 90% for the relation mappings. We also created a distant supervision based gold standard for the purpose and the data has been released as part of this work. Furthermore, we analyze the effect of clustering and empirically show its effectiveness as a relation mapping technique over other techniques. Overall, our work positions itself on the intersection of information extraction, ontology mapping and reasoning.

ZUSAMMENFASSUNG

Offene Systeme zur Informationsextraktions, „Open Information Extraction" (OIE) Systeme, wie z.B. NELL oder REVERB, sind in der Lage, aus verschiedenen textuellen Quellen domänen-unabhängigen Fakten zu extrahieren, und diese als Fragmente natürlicher Sprache in Form von Subjekt-Prädikat-Objekt Tripeln, d.h. letztlich semi-strukturiert, auszugeben. In häufig genutzten strukturierten Wissensbasen, „Knowledge Bases" (KBs), wie z.B. DBPEDIA oder YAGO, ist eine Vielzahl dieser Fakten jedoch nicht enthalten, da diese KBs durch spezifische, und somit in ihrer Breite beschränkten, Extraktionsschemata erzeugt werden. Aus diesem Gegensatz eröffnet sich die Möglichkeit, strukturierte KBs mit Fakten aus OIE Systeme zu komplementieren. In dieser Arbeit werden Methoden untersucht und ein System entwickelt, um diese Integration zu ermöglichen. Der Hauptbeitrag ist hierbei der Entwurf und die Implementierung einer umfassenden, Ontologie unabhängigen und allgemeine Architektur zur Informationsintegration. Die vorgeschlagene Lösung besteht aus folgende Modulen: (1) Abbilden von Subjekt- und Objekt-Termen aus OIE Fakten auf KB Instanzen. (2) Abbilden von relationalen Termen auf die in der KB definierten Relationen. Da verschiedene OIE Relation, mit entsprechend verschiedenen Oberflächenformen, u.U. auf dieselbe, normalisierte KB Relation abgebildet werden, müssen (3) die OIE Terme zunächst in Clustern gruppiert werden, was mit Hilfe von Markov Clustering erreicht wird. Im Kern basiert diese Arbeit auf der Ausnutzung der Beziehungen zwischen Relationen und ihren Argumenten, d.h. Subjekt- und Objekt-Termen, wobei dies in einer kontextunabhängig Weise geschieht, welche die allgemein Anwendbarkeit der Methoden auf beliebige KBs sicherstellt. Die Leistungsfähigkeit dieses Ansatzes wird durch eine experimentelle Evaluation mit zwei OIE Systeme auf einem selbst erstellen Gold Standard untersucht, in welcher eine Präzision von 85% in Bezug auf Instanzen und 90% in Bezug auf Relationen, sowie des Weiteren ein positiver Effekt des Markov Clusterings gegenüber andere Ansätze, aufgezeigt werden kann. Insgesamt leistet diese Arbeit damit einen relevanten methodischen Beitrag zur Integration von semi-strukturierten OIE Fakten aus Texten und strukturierten Fakten aus KBs dar.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. Dr. Heiner Stuckenschmidt for a wonderful learning opportunity he provided to me. I feel deeply indebted to the way he patiently let me learn from my own mistakes. With his excellent guidance, I have learnt and improved a lot about the process of doing scientific research. I consider myself very fortunate to have worked with him. I am highly grateful to Dr. Christian Meilicke for his selfless support and close interest in my work. His constant motivation helped me to overcome some of the critical phrases during the PhD. He taught me some of the good practices for scientific writing. His way of looking into the problems is highly commendable and I always kept learning something new from him in these three years of my life. I also take this opportunity to thank Prof. Dr. Simone Paolo Ponzetto for being a constant motivation with his excitement about research and cordial personality. I would also like to thank Dr. Mathias Niepert for some of his ideas which he shared during the inception phases of the PhD and for bagging the Google Faculty Research Award (funding for this work).

The whole work would not have been possible without the lovely group of people I was working with. It includes each and every one with whom I was fortunate to have been in close contact with. Thank you Stephanie for helping me through all the hassles of official documents, Sebastian for your immediate resolution of the infrastructure related issues, Cilli for being such a lovely friend (I feel lucky to have known her), Jörg for being always so helpful in and out of office, Micha for always being my "German" friend (I am his "Indian" friend and we won the "couple of the month" title) and everyone else in the DWS lab.

Finally, I would like to thank Silpi, my loving wife for convincing me to pursue a scientific career. If not for you, I would have been still an undergraduate engineer working as a software developer. Special thanks goes to my family back home in India (my parents, my father-in-law) for their continued love, support and encouragement. Thanks for learning to use a touch screen device (and Skype, Watsapp, Viber) and making me always believe "..whatever happens in life, always happens for good".

PUBLICATIONS

Some of the ideas, figures, tables and numbers have appeared in the following publications:

- Arnab Dutta, Christian Meilicke and Heiner Stuckenschmidt. **Enriching Structured Knowledge with Open Information**. In: *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*; 267-277. , Geneva, 2015.

- Arnab Dutta, Christian Meilicke and Simone Paolo Ponzetto. **A Probabilistic Approach for Integrating Heterogeneous Knowledge Sources**. In: *Lecture Notes in Computer Science The Semantic Web: Semantics and Big Data : 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014, Proceedings*; 286-301. Springer International Publ., Cham, 2014.

- Arnab Dutta, Christian Meilicke and Heiner Stuckenschmidt. **Semantifying Triples from Open Information Extraction Systems**. In: *Frontiers in Artificial Intelligence and Applications STAIRS 2014 : Proceedings of the 7th European Starting AI Researcher Symposium*; 111-120. IOS Press, Clifton, Va. [u.a.], 2014.

- Arnab Dutta, Mathias Niepert, Christian Meilicke and Simone Paolo Ponzetto. **Integrating Open and Closed Information Extraction: Challenges and First Steps**. In: *CEUR workshop proceedings NLP-DBPEDIA 2013 : Proceedings of the NLP & DBpedia workshop co-located with the 12th International Semantic Web Conference (ISWC 2013) Sydney, Australia, October 22, 2013*; . RWTH, Aachen, 2013.

- Arnab Dutta and Michael Schuhmacher. **Entity Linking for Open Information Extraction**. In: *Lecture Notes in Computer Science Natural Language Processing and Information Systems : 19th International Conference on Applications of Natural Language to Information Systems, NLDB 2014, Montpellier, France, June 18-20, 2014. Proceedings*; 75-80. Springer Internat. Publ., Cham, 2014.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

KB Knowledge Base

OIE Open Information Extraction

NELL Never Ending Language Learner

MLN Markov Logic Network

YAGO Yet Another Great Ontology

IM Instance Matching

RM Relation Matching

db: http://dbpedia.org/resource/

dbo: http://dbpedia.org/resource/ontology

Part I

INTRODUCTION

1

MOTIVATION

Over the last few decades, unsupervised and semi-supervised techniques for extracting knowledge from heterogeneous sources, have marked the beginning of different genres of information extraction systems. To begin with, some of the state of the art extraction systems like DBpedia [ABK+07], Yago [SKW07], Freebase [BEP+08] are well known in the community. This class of extraction systems define a genre of IE which are confined to some encyclopedic knowledge sources, like Wikipedia for instance. The knowledge extracted by these sources maintain a structure in the form of hierarchies of concepts and relations. Additionally, the resources from these systems are well defined with unique identifiers. All of these systems are not necessarily unsupervised but involves a bit of manual interference.

In the more recent years, the focus shifted to methods involving lesser human intervention and hence unsupervised techniques. This new genre of IE systems introduced the term *open* Information extraction (OIE) [BCS+07] as an effort to highlight the schema independent approach. These systems do not adhere to any particular domain or schema, unlike its former kins, but looked for information from every possible domain covered by textual contents available in the web. Systems like TextRunner [BCS+07], Reverb [FSE11], Nell [CBK+10], Ollie [MSB+12], were some of the prominent names classified as OIE systems. Although highly scalable, typically to the size of the web, OIE systems are often plagued with ambiguity due to the lack of any unique identifiers for the resources or relations. In contrast to the structured knowledge bases, OIE systems are usually schema less or schema poor in structure.

This thesis work has been perceived as an integration task between these two broad strains of IE systems. Figure 1, illustrates the scenario as two closely related, yet complementary systems. It marks the advantages and drawbacks of each of the systems and distinctly states the scope of integrating them. For instance, the ambiguity inherent in the OIE extracts can be counteracted with accurate iden-

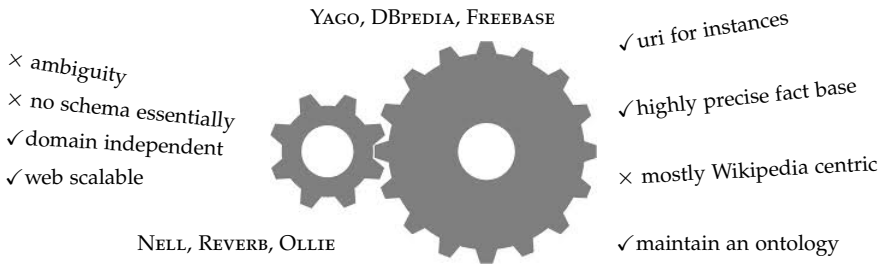

Figure 1: The gear box: Scope of integrating structured knowledge bases with open extraction systems. '×' denotes demerit of the systems, '✓' denotes merits.

tifiers for instances and relations in a structured KBs. While simultaneously, the Wikipedia centric extractions of the structured KBs can hugely benefit from the typical web scale coverage of the OIE systems. In this work, we integrate the best of the two systems: exploit the scalability of the OIE systems by harnessing the schema provided by its counterpart. We envision this integration task as a symbiotic process: on one hand the data produced by OIE systems can be made useful for reasoning and analysis and can also result in a new generation of search engines [Etz11], while on the other hand the classical IE systems can extend its less focused or poorly covered domains with additional facts. In the rest of this paper, we would refer to the classical IE systems as the target knowledge base (KB) and the other simply as OIE systems. In a relaxed notion, every IE system is a KB but in this context, by referring to the schema-oriented systems as "target KB" we want to clarify our general approach: starting from schema-less systems and moving towards the schema-oriented ones.

In this work, we adhere to our underlying goal of integrating unstructured, ambiguous knowledge sources (in the form of open information extracts) with the structured KBs. In particular, we illustrate our methodologies by integrating REVERB and NELL outputs to DBPEDIA. Now, referring to the Figure 1, a successful integration task is the one where the demerits of one are eliminated or overcome by the merits induced by the other. This has to be a bidirectional step. One possible and tangible way of achieving this is to augment the structured KB with OIE data sources. Augment in the sense, generating additional KB assertions which were missing from the KB, but learnt from the OIE sources. In this way, the target KB extends itself, thereby breaking the shackles of being just Wikipedia centric. On the other extreme, the OIE triples are semantified, every instance and relation

is expressed in the KB vocabulary, thereby overcoming vagueness and ambiguity. This is the core motivation for this thesis.

In the process we identify and solve different sub problems. Our proposed solution towards the broader goal of integrating the two IE systems is a combination of each of the solutions to these sub problems.

- given an open domain extracted fact is it possible to semantify the relationship and its arguments (instances)?
- If so, what is the most likely reference to the instances in terms of a target KB?
- what is the most likely KB relation which captures the same semantic sense as expressed in the extracted fact?
- does the semantification step generate some new knowledge for the target KB?

These are some of the broad questions we tried to answer in this work. Now we must note that the solution to each one of these is necessarily not a discrete solution but rather inter-twined. This is much due to the closely connected nature of the problem itself. A semantification step is all about removing ambiguity and this is achieved automatically if the instances and relations have accurate target KB counterparts. An accurate instance disambiguation is possible, if the semantics of the OIE relationship is correctly deciphered and inversely, a better understanding of the semantics of an OIE relation would benefit towards a better instance disambiguation task. Last, but not the least, assuming a semantification is possible then we can regenerate each one of the OIE triple as a KB assertion. This brings us to the final block of the puzzle: do they all generate some new KB triples? This is specially interesting since, it answers the claim that a domain independent open extractions can indeed augment a structured KB.

We propose, a general framework for this entire pipeline which is implemented keeping in mind the modular and interdependent nature of the problem. Thus in this work, we have focussed on the some of the key research areas and have made the following contributions.

- a context free technique for instance matching. We have devised a probabilistic way to find accurate references of ambiguous terms from OIE extractions to a target KB.
- a rule mining based scheme to find the most likely KB counterparts for OIE relational phrases.
- a vector based and a graph based method for clustering natural language phrases and its effectiveness as a relation matching scheme.
- a generalized open-source framework for extending the target KB with additional assertions learnt from the open extractions.
- the well documented data sets generated within the scope of this work. Especially, gold standard data sets for NELL instance matching, a REVERB relation phrases matching data set.

Our empirical results show over 90% precision values on the evaluation sample for both REVERB and NELL data sets. The instance matching has been compared against a strong baseline using the Wikipedia most frequent sense. While for the relation matching techniques we compared against a system proposed by Liu et al., [LLZ+13]. We recreate the exact evaluation setup as performed by the authors and achieved better precision scores. In the following, Chapter 2 introduce the various IE systems, especially the major differences they have with one another and also discuss how they fit into our entire work. We present the proposed framework in Chapter 3 and briefly describe the functionality of each. This highlights the individual components and their role towards the solving the sub problems mentioned.

This thesis has been divided into some major parts. Each of these are dedicated to one concrete sub-problem and can be considered as a complete analysis on the problem in themselves:
In particular, the current **Part I** is an introduction for the thesis. It presents some background on the information extraction systems, introduces the general paradigm of *open* information extraction with strict differences it bears with domain-specific classical IE systems. The major contribution of this part is the introduction to the knowledge generation task as a mode of knowledge base integration. We have adopted a top-down approach in presenting this thesis, where we describe in the beginning the broad objective. We present the concept of knowledge generation as

perceived in the context of out work. The detailed empirical analysis are avoided here but presented later. Here, we primarily aim at making the core ideas transparent with exhaustive examples.

Part II presents in depth of the instance matching technique. Chapter 4 introduces the general problem area and formalizes the task in our context. The two subsequent chapters present the baseline approach (Chapter 5) and the probabilistic approach (Chapter 6) to perform instance matching. The empirical analysis are reported in Chapter 7.

Part III presents the technique for relation matching. It follows a similar structure as seen with instance matching. The rule based and cluster based methods are detailed in Chapters 9 and 10 respectively. We report on the empirical results in Chapter 11. The results for the two approaches have been explicitly presented under separate sections(11.1 and 11.2).

Part IV aggregates the major two sections and presents a way to perform knowledge generation. This part also presents a generic algorithm for generating new knowledge from OIE triples. We introduce the semi-supervised gold standard creation scheme in Chapter 13. Our framework is capable of handling different kinds of OIE systems and under varying settings. We evaluate these different *workflows* against the gold standard and present detailed analysis for them.

Part VI is the concluding part where, we summarize the major aspects of this work and try to introspect our method under a different lens. In particular, we report the merits of our framework and also highlighting scenarios where it fails to achieve the best results. We also discuss some of the areas of future research, where this work can potentially serve as a precursor.

2

INFORMATION EXTRACTION SYSTEMS

According to Piskorski et al., [PY13], "*Information Extraction is about deriving structured factual information from unstructured text*". The information source can be web pages, texts, articles, structured tables or even event descriptions. To correctly structure this information, it is often necessary to have a *domain* knowledge. As the extractor processes a piece of text, it tries to *slot-fill* its predefined set of templates for the entities and relations that might occur in the text. These are more commonly referred to as domain-specific IE systems. In the following two sections (Section 2.1 and Section 2.2), we present the two state of the art classical IE systems belonging to this category.

However, in the more recent years, the focus of IE has shifted to more domain independent methods designed to work on large and diverse corpora of texts (typically web scale), without any predefined knowledge. Aptly, termed as *open* IE, this genre of extraction systems employ a multitude of semi-supervised machine learning based models to automatically extract relations and arguments. In Section 2.3 and Section 2.4 we present two OIE systems which we work with in this work. Also in the end of this section, we mention the other IE systems which are available but not used in the context of this thesis work.

2.1 DBPEDIA

This is one of the largest projects that has positioned itself as the central hub of the linked data cloud. DBpedia particularly aims at unsupervised methods of acquiring large amounts of structured information from Wikipedia. It mainly extracts the content from Wikipedia infobox templates, categories, geo-coordinates, etc.. However, it does not employ any relation learning approach from the Wikipedia categories. The extracted template information is mapped to an ontology: specifically to its own fixed set of classes and relations. Moreover, the ontology is with more than 1000 different relations much broader than other existing ontologies like YAGO [SKW07] or semantic lexicons like BabelNet [NP12]. It consists around

4 million entities, around 500+ million facts and 115K concepts. DBpedia represents its data in accordance with the best-practices of publishing linked open data. The term *linked data* describes an assortment of best practices for publishing, sharing, and connecting structured data and knowledge over the web [BLK+09].

The DBpedia ontology models the relationships using the resource description framework (RDF), a generic graph-based data model for describing objects and their relationships. Each entity has an unique identifier in the form of URI. This is also valid for the relations which have a well defined URI. The entities are often called instances since they belong to a particular concept class. For instance, `dbo:OfficeHolder` is a concept class and `db:George_Washington` is an instance of the class. These kinds of class assertions are represented with the relation `rdf:type`, and hence the RDF representation simply takes the form `rdf:type(db:George_Washington, dbo:OfficeHolder)`. A particular concept is often in a subsumption relation with its super class. For instance, `dbo:OfficeHolder` $\subset$ `dbo:Person` $\subset$ `dbo:Agent`. Hence, the class membership of an instance automatically makes it an instance of its parent classes (`db:George_Washington` is also a `dbo:Person` and `dbo:Agent`). Such hierarchical structures within the concepts are invaluable for us. We exploit exactly this structure in particular to discover domain and range restrictions for open domain relations. The large number of relations and instances make it an appropriate choice as the target/reference knowledge base to which we can link the resources from Nell and Reverb.

2.2 YAGO

A popular acronym for **Y**et **A**nother **G**reat **O**ntology is a semantic knowledge base created from Wikipedia, Wordnet [Mil95] and GeoNames[1]. It is an exhaustive source of about 286K concepts, 10 million entities and over 120 million facts. However, the number of relations maintained by Yago is less, around 100 compared to DBpedia. Yago is known for its high accuracy of over 95% which can be attributed to its approach of exploiting the Wikipedia category system. Hence, Yago has more fine grained concepts in its ontology capable of encoding a lot more information than simple concept names in DBpedia (for instance, the Yago class "*MultinationalCompaniesHeadquarteredInTheNetherlands*"). The extractions by Yago are not infobox based unlike as done in DBpedia. In particular, Yago links

1 http://www.geonames.org/

the leaf node of Wikipedia categories into the Wordnet hierarchy.

In our work, we do not use YAGO as the final target KB, rather use it as an additional KB to complement DBPEDIA. Often, some instances from DBPEDIA have missing type information. The type information is very important for our methodology and it provides us an invaluable source of implicit information. We use the mappings between DBPEDIA and YAGO classes to our advantage. In particular, there are instances typed with YAGO class names instead of DBPEDIA concepts. We use the range of mapping files, the YAGO taxonomy, and subsumption relationships to infer the DBPEDIA class[2]. Hence, in the context of our whole framework, we are using the structured information from the two largest knowledge sources.

2.3 REVERB

To begin with, we must mention the mother project of KNOWITALL [ECD+05] whose primary intention was to change the whole way of performing web search: reading the web instead of retrieving web pages [BE08]. TEXTRUNNER was the first generation of *open* information extraction system. Although it proved effective, it soon ran into problems like, relational tuples set full of non-informative and incoherent extractions [FSE11]. REVERB was the next generation of OIE system devised specifically to tackle the problems with its earlier version. It identifies and extracts binary relations between entities in a text. The feature which sets REVERB apart from its peers is that it is a "relations first" approach rather than a "instance first approach". The later approach is problematic since often a compound relation phrase may contain noun phrases and "instance first" approach will confuse them as the entities (relation arguments) and not identify them as part of a complex relational phrase. For example, in the given text pattern *X was attending high-school in Y*, an instance first approach will identify, "*X*" and "high-school" as the arguments for the relation "*was attending*". While with relation first approach, "*was attending high-school in*" will be identified as the correct and complex relational phrase. Thus a lot of incoherent extractions were avoided by REVERB.

REVERB provides us with a valuable source of ambiguous facts. The instances do not have URIs and the extracts are from varied domains. Additionally, the presence of large number of relational phrases and often some of them semantically

2 available from the downloads page of YAGO

similar ones makes our problem setting bit harder compared to NELL. Unlike NELL it has no hierarchical structures and adheres to the more general idea of schema independent knowledge source.

2.4 NELL

The **N**ever **E**nding **L**anguage **L**earning [CBK+10] (NELL) project's objective is the creation and maintenance of a large-scale machine learning system that continuously *learns* and extracts structured information from unstructured web pages. It operates on a large corpus of more than 500 million web pages[3]. This marks the new genre of information extraction systems, unlike the resource specific extractions observed with YAGO or DBPEDIA. NELL employs a set of seed classes and relations and for each sets 10-15 positive and negative instances. The core idea for NELL is to build several semi-supervised machine learning [CSZ10] components that accumulate instances of the classes and relations, re-train the machine learning algorithms with these instances as training data, and re-apply the algorithms to extract novel instances. These steps are repeated over and again and are aptly called *iterations*. We see in the later section (Section 7.1.1) that the publicly available NELL datasets are released under various iteration numbers. Since numerous extraction components work in parallel and extract facts with different degrees of confidence in their correctness, one of the most important aspects of NELL is its ability to combine these different extraction algorithms into one coherent model. This is also accomplished with relatively simple linear machine learning algorithms that weigh the different components based on their past accuracy. NELL has been running since 2010, initially fully automated and without any human supervision. Since it has experienced concepts drift for some of its relations and classes, that is, an increasingly worse extraction performance over time, NELL now is given some corrections by humans to avoid this long-term behavior. NELL does not adhere to any of the semantic web standards such as RDF or description logic [DNMP13].

We use the extracts from NELL and use it as our source data. These are open domain extracts and not uniquely identified. This makes it suitable for the purpose of semantifying ambiguous facts. However, NELL maintains a concept and

3 http://lemurproject.org/clueweb09/

relation hierarchy of its own[4]. We design a framework which avoids the use of this hierarchy or of such kinds since, our goal is to achieve KB extension by using the least amount of information from the open domain. This makes our approach very general and requires no additional requisition for the sources.

OTHERS

The list of IE systems are not just limited to these. We however present the brief detail only for the ones relevant to our work. The choice of DBPEDIA and YAGO as the target KBs was solely based on their robust ontologies, exhaustive datasets, publicly available query endpoint, size and availability. While, the reason for NELL or REVERB as sources was comparatively simpler since, these are the two state of the art systems with fairly large number of extractions. We mention here the other different IE systems (both the classical ones and open domain ones).

Freebase is also a well known knowledge base, which maintains a tuple database to structure human knowledge [BEP+08]. They provide read-write access through HTTP based graph query API. It consists of around 125 million tuples, over 4000 concepts types, and more than 7000 properties. This could have also qualified as a target KB for our work. However, due to easier access methods of DBPEDIA over FREEBASE we opted for the former.

WOE [WW10] system is an improvement over TEXTRUNNER in terms of both precision and recall. The work uses heuristics to match the Wikipedia infobox attributes to sentences for constructing the training data. WOE is unique due to its ability to run under two modes: a feature based one which is as fast as that of TEXTRUNNER; and also a dependency parse based feature.

Ollie is another OIE project from the same KNOWITALL project. The purpose of yet another system (OLLIE) was to remove some of the problems in REVERB: first, relation extraction mediated by verbs and second, lack of use of context [MSB+12]. However, it is a built up on top of REVERB since, internally it uses REVERB to have a set of seed inputs. Furthermore, OLLIE employs a faster dependance parsers compared to its predecessors. In our opinion, working with REVERB was a more rudimentary choice over its slightly modified siblings.

4 http://rtw.ml.cmu.edu/rtw/kbbrowser/

OpenIE 4.0[5] is focussed on n-ary relations in general. It is a successor of OLLIE. The extracts do not exactly fit our goal since the extracted facts are more helpful to determine if a particular assertion is correct or incorrect. Hence, the extracts from OpenIE4.0 are not just some ground facts, but often a clause annotated with it. For instance, from the sentence, "*Some people say Barack Obama was born in Kenya.*", OpenIE extracts an n-ary tuple as "*Some people say: (Barack Obama, was born in, Kenya)*" instead of simply "*Barack Obama, was born in, Kenya)*"[6] .

Exemplar [dSMSB13] is another system following the paradigm of open IE specializing in extracting n-ary relations. This is similar as compared to the general idea of OpenIE 4.0 but involves lesser complex NLP machinery. Internally, it also uses dependency parse trees to accurately find the connections between a given relation and its arguments.

In this section, we mentioned the major IE systems while there also exists others (Kylin [WW07], StatSnowball [ZNL+09]). However, it is an interesting observation that the general research trend is now directed more towards domain independent extraction methods than structured domain specific extractions. This is evident with the fast paced development and releases of different OIE systems with only minor optimizations over its predecessors.

5 http://openie.allenai.org/
6 The example has been cited from https://github.com/knowitall/openie

3

FRAMEWORK OVERVIEW

The primary contribution in this thesis is a context agnostic methodology for integrating heterogeneous knowledge sources. We are essentially exploiting the best of both the worlds in our work: the domain independent coverage of OIE to enhance some poorly covered domain of the target KB, while simultaneously resolving ambiguity on the OIE terms and relations using the rich ontological structure of the structured KB. This idea of integration has been exemplified in this work as a knowledge generation task. Essentially, the source of newly generated knowledge is the input from OIE. Thus the task of generation can be referenced under multiple nomenclatures as,

1. **semantification** of the OIE triples. Since the input triples from the OIE systems undergo a transformation into a fully semantified triple in terms of the KB vocabulary. The vague and ambiguous terms (and relations) are replaced with well defined URIs for each of its resources (instances and relations).

2. **extension** of the target knowledge base. The new triples generated from the OIE inputs are unknown to the KB hence, the generation step adds a set of new triples to the KB.

In the process we designed a framework to perform the task of knowledge generation. The goal in this chapter is to present a complete architectural overview of our proposed framework. While presenting the framework, we refrain from intrinsic details since our aim is to present the abstract idea of the workflow at this stage of the thesis and get familiarized with the individual modules along with their contributions in the pipeline. We must note the modular architecture in the framework, where each of the modules can be replaced by other algorithms of choice and still the architecture would maintain to be functional. Additionally, we chose one example OIE triple and present relevant values for it as as we visit each of the modules. Some of the ideas and figures have been already published in our previous work [DMS15].

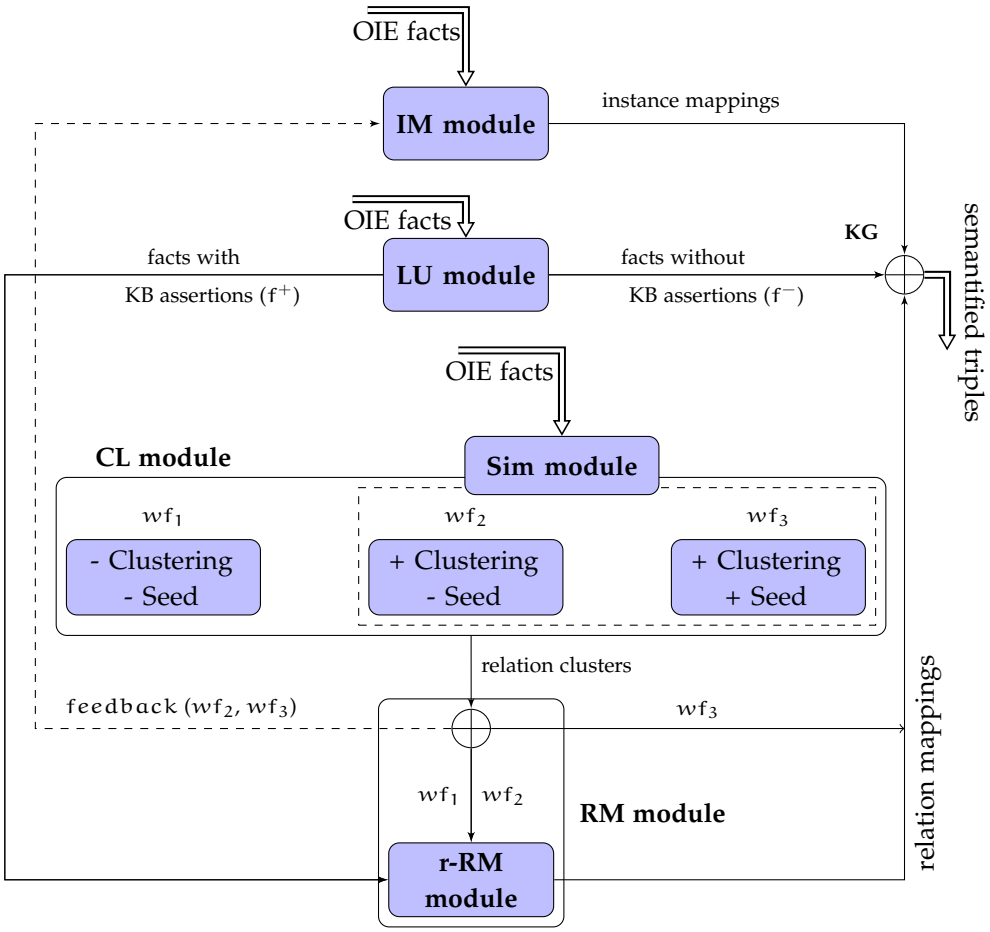

Figure 2: The system architecture. IM: Instance matching, LU: Look up, CL: Clustering, RM: Relation matching, r-RM: rule based RM, Sim: Similarity Computation, KG: Knowledge Generator, *wf*: workflow

3.1 MODULES

In Figure 2, we depict the framework for translating OIE facts to assertions in the target knowledge base. It consists of six primary interacting components or modules. We consider an example REVERB triple and present the primary modules and state the exact transformations the triple undergoes. Let us consider the REVERB triple

$$\texttt{is headquartered in}(\text{EMBL}, \text{ Heidelberg})$$

Knowledge Base Look Up

This purpose of this module is to search for facts that are already stated in the target KB, DBPEDIA in this case. This is marked as "LU module" in the figure. Generally, for every fact of the form $p(s, o)$, we search for assertions in DBPEDIA that

relate s and o. To do so, we require the DBpedia instance mappings for s and o. We obtain the top-k mappings for both the subject and object using Wikipedia as the background knowledge. If S and O respectively denote the top-k DBpedia candidate mappings, then we essentially check for all the |S| * |O| combinations. Out of these, some of the assertions might correspond to the given fact in the target KB. This is an indication that the original OIE triple has a counter fact in the KB. This class of OIE facts are called f^+; facts with KB assertions and are fed into the later relation mapping module (RM module) as evidences for relation mapping. If there is no assertion existing in the KB, the facts are classified as f^-; facts without KB assertions. These facts are, at the end of the overall workflow, translated into the DBpedia vocabulary. The idea can be illustrated with the Example 1.

Example 1. *For instance, consider the* Reverb *triple,*
is headquartered in(AEGON, The Hague).
The top-k candidates for the subject will be
S = *{db:Aegon, db:Aegon_UK}.*
O = *{db:The_Hague, db:Den_Haag_Centraal_railway_station, ...}.*
The LU module looks for any relation in DBpedia *for all the S × O pairs of KB instances. And there exists the assertion in* DBpedia *as the following*
dbo:location (db:Aegon, db:The_Hague)
The above subject-object combination is one of the pairs from the above set of possible S × O combinations. Our example triple also undergoes undergoes a similar look up and due to absence of any existing KB assertion, we mark it as f^-.

It might also happen that s and o may appear in the KB as several different property assertions (for instance, dbo:location(db:London, db:United_Kingdom) and dbo:capital(db:London, db:United_Kingdom)). Hence, the output of this module with respect to our example triple is,

$$\begin{aligned} &\dots \\ f^+ &: \text{is headquartered in}(\text{AEGON, The Hague}) \\ f^- &: \text{is headquartered in}(\text{EMBL, Heidelberg}) \end{aligned}$$

"..." denote the presence of other OIE triples from the input which are similarly annotated as f^+ or f^-. This module essentially performs a fact separation over a

KB look up.

Instance Matching
This is the module responsible for finding the most probable KB reference of the terms occurring as subject or object within an OIE triple. This is marked as "IM module" in the figure. This module takes as input facts from OIE systems. The outcome of the IM module is a set consisting of *at most one* mapping of the subject and object terms to DBpedia instances. Here the set of input triples undergoes a probabilistic matching algorithm. Observe the emphasis on the aspect that every term can have maximum of one mapping. This is a realistic modeling since a particular term in an OIE fact usually refers to only one real world entity and not to a multitude of those. Every OIE instance term (not the relational phrases) can be considered as a surface form representation of the actual entity in context. For instance, "*london*" might refer to a range of real world entities: the cricket stadium "The Oval", the capital city of UK, or even the poem by William Blake. Hence, there is a considerable amount of ambiguity involved in finding the correct entity reference. We exploited the English Wikipedia as entity tagged corpus and formulated the problem of finding exact KB references as an inference problem in a graphical model (markov logic networks). We start with a prior hypothesis of top-k candidates about the OIE terms and employ reasoning techniques to eliminate candidates which cannot be realistic. For instance, in the Nell triple *subpartof(city:heathrow, city:london)*, the likelihood of *subpartof* having place as a range, enhances the chances of *london* being a place and not a poem or a person. And similarly, the more we *observe* places as ranges for the relation *subpartof*, it is more likely for the relation to have place as range. Thus, there is a deep rooted dependency between the OIE subject/object terms and the OIE relation connecting them. This module considers these dependancies and models the solution around it. It starts with a prior assumption of the instances/OIE terms, eliminates the implausible ones, and refines the final set of likely KB matches. This is set in an iterative bootstrapped fashion to refine and reason on the final set of instance mappings.

Referring back to our original example, the f^- OIE triple is splited up and the top-1 mappings of each subject and object are analyzed. The type information of them hints that, the domain of *is headquartered in* may possibly be `dbo:Organisation` and the range `dbo:Town`. This is an evidence but not strong enough. We do the

same for the other OIE triples with the same associated relations. This includes looking for the instance types of all the terms (as subjects and as objects) for the particular relation, in this case, all the facts of the form *is headquartered in(*,*)*. Additionally, it may also accept collections of OIE facts, grouped together based on their relational phrases (for instance, facts of the form *is headquartered in(*,*)*, *has its headquarters in(*,*)* and so on). Based on the types the IM step maps the subject and object as follows[1],

$$\text{EMBL} \rightarrow \texttt{db} : \text{European\_Molecular\_Biology\_Laboratory}$$
$$\text{Heidelberg} \rightarrow \texttt{db} : \text{Heidelberg}$$

It is a very interesting to note that both f^+ and f^- facts contribute towards type evidence generation. We apply the algorithm on the entire set of input facts. We need to exploit as much of evidence possible from the input instances. Hence, it is immaterial if a given OIE triple has a KB counterpart assertion. It is useful to restate at this point that an OIE triple is marked f^- because of a missing relation assertion in KB, even though a perfect mapping might be possible on the instances. The IM step heavily benefits if there is a mapping for an OIE term, it does not require both the subject or the object terms to be mapped, either one of them helps in this type generation step.

Clustering Module

The clustering (represented as "CL module" in the figure) module generates as output clusters of relational phrases having similar meaning, i.e., that can be correctly mapped to the same target KB relation under the context of having identical semantic meanings. From Figure 2 we observe that, there is no direct dependency of instance matching module on clustering and the workflow is still complete without the clustering mechanism. However, this module is incorporated into the workflow to handle scenarios where we require multiple OIE relations to be reasoned together. This was the case with REVERB data set. The design of this module was made in a way such that, it can work either ways: with relations requiring clustering or ones without the need.

1 We refer to `db:European_Molecular_Biology_Laboratory` simply as `db:EMBL` for concise representations in the text

We implemented three different clustering methods which differentiates three distinct workflow modes our framework can handle. We present the detailed explanations of the following workflows in Section 3.2.

- wf_1: the trivial case for which we treat each relational phrase as an one element cluster. This is similar to the case of no clustering.
- wf_2: clustering only the OIE input relations without any KB relations as input seeds.
- wf_3: clustering the OIE input relations along with the KB relations as input seeds.

Clustering details are explained in detail in Chapter 10. As illustrated in Figure 2, these three different workflows (wf_1, wf_2 and wf_3) have been marked along the figure. The clusters generated by wf_2 and wf_3 are forwarded to the IM module (dashed arrow in Figure 2), which is executed again to improve the instance mapping due to better statistical coverage of clusters compared to the coverage of individual relational phrases.

Referring to our current example, we run the clustering algorithm on the input relations and generate clusters. We present the clusters generated for both the workflows.

$$\begin{aligned} wf_1 &: \{\ldots, \{\text{is headquartered in},\}, \ldots\} \\ wf_2 &: \{\ldots, \{\text{is headquartered in}, \text{ headquartered in}\}, \ldots\} \\ wf_3 &: \{\ldots, \{\text{has its headquarters in}, \text{ is headquartered in}, \\ &\qquad \text{headquartered in}, \text{ dbo : headquarter}\}, \ldots\} \end{aligned}$$

As seen that, the given relation of interest is grouped with a set of other related phrases. For completeness, we also explicitly show the default case with wf_1, which consists of only one element clusters. With wf_3 we have the cluster with the DBpedia relation `dbo:headquarter`.

Similarity Computation

This is an additional block represented in the figure as the "Sim module" . This is one of the other major modules in our workflow and is relevant in the context of clustering, particularly for the two workflows: wf_2 and wf_3. Our underlying clustering technique is a graph based one which requires a pairwise affinity score

between any two pairs of OIE relations. As seen in the figure, the similarity computation is performed only in wf_2 and wf_3 (shown with the enclosing dotted rectangle). In this module, we consider two relations in pairs and define a similarity score between them. This score indicates the degree of semantic similarity between the two relations. In defining this score, we consider two major influences: first, how the two relation instances are distributed in the input data set (referred to as ov in Section 10.2.1); second, how much the relations are semantically related in general (referred to as wn in Section 10.2.1). The later uses external data sources like Wordnet. In Section 10.2.1 we present detailed ways of using these scores to assign the pairs of relations with an unique score. Using our running example,

$$ov(\texttt{is headquartered in}, \texttt{headquartered in}) = 0.012$$
$$wn(\texttt{is headquartered in}, \texttt{headquartered in}) = 1.0$$

Relation Mapping

Represented as "RM module" in the figure, this module tries to map a relation phrase or clusters of such phrases to a target KB object property. The relation mapping can be done by either mapping each relation phrase or clusters of such phrases to a KB or letting the KB relations make logical groups with the OIE relations. The final result is a set of mapping of an OIE relation to a KB relation.

It must be noted, the RM module tries to map OIE properties to KB object properties (i.e which are defined between entities in the KB, for instance, `dbo:author` which is defined between entities `dbo:Work` and `dbo:Person`) and not to data type properties (i.e. between literals like `dbo:foundationDate` defined between `dbo:PopulatedPlace` and `xsd:date`). As shown in the figure, it consists of one major sub block labelled "r-RM" which denotes rule based relation mapping. The underlying mechanism for this module is an association rule mining based approach, which attempts to mine for the frequent rule pattern of the form $\texttt{rel} \rightarrow (\texttt{domain}, \texttt{range})$. Observe that, the output f^+ from the LU module is also an input for this relation mapping task. Every OIE fact which is also "observed" in the target KB in some relational form, can be considered to be a strong evidence for a likely mapping. This is influenced by the general paradigm of distant supervision [AMC14, MBSJ09] which states that,

"*If two entities participate in a relation, any sentence that contains those two entities might express that relation*"

For the OIE fact "'*offers a wide range of(altec lansing, speaker systems)*" the analogous fact we observed in DBpedia was "`dbo:product(db:Altec_Lansing, db:Loudspeaker)`". This provides a possibility that "*offers a wide range of*" might be mapped to "`dbo:product`" [DMS15]. In general, the whole set of f^+ facts provides evidences for a possible mapping. This module exploits the domain and range of a DBpedia property in the stated assertion. The first workflow wf_1 involves a direct application of this technique on Reverb and Nell. For workflow wf_2 to treat clusters of relational phrases as well. Eventually, this module outputs a set of relation mappings. Note that clustering with DBpedia properties as seeds (wf_3) implicitly solves the relation mapping problem, since each relational phrase is mapped to the DBpedia seed in that cluster. Thus, the r-RM module is not used by wf_3. And so, in the figure we do not see wf_3 directed towards "r-RM", but output directly.

Here, the importance of the fact separation module is evident especially for the rule based RM. The f^+ triples essentially gather the likely relations that might hold true. As seen in the example above, it is possible that *is headquartered in* maps to `dbo:location`. We estimate the likelihoods of such a relation mapping from all the possibilities learnt from the f^+ triples. Hence,

$$wf_1 : \text{is headquartered in} \rightarrow \text{dbo} : \text{headquarter}$$
$$wf_2 : \text{is headquartered in}, \ \text{headquartered in}, \rightarrow$$
$$\text{dbo} : \text{headquarter}, \ \text{dbo} : \text{foundationPlace}$$

We observe that the mapping is no more on single relations but on clusters for wf_2. This allows to map to other KB relations which were previously not possible.

Knowledge Generator

The final block of the framework is combining all the information we gathered across in the previous modules. This is shown as "KG" in the Figure 2. Given the instance and relation mappings for a certain fact, each component of the fact can be translated directly to an assertion formulated in the vocabulary of the target knowledge base. The semantification process is applied only to f^- facts. The terms occurring as subject/object are instance mapped to the KB, DBpedia in this case, the relation is mapped to KB object property. This generates a new fact in the KB, since this was originated from the f^- facts. For instance, we started from the OIE fact "*originated in(Japanese honeysuckle, Japan)*", and generated the new assertion "`dbo:origin (db:Lonicera_japonica, db:Japan)`", with the guarantee that

there are no pre-existing assertion in DBpedia with any other object property connecting the two entities.

Hence, collating all the separate pieces, we have individual mappings for the subject, object and the relation (predicate) in the Reverb triple we originally started with. This generates a new triple which does not exist in DBpedia as a SPO triple.

$$\text{is headquartered in}(\text{EMBL},\ \text{Heidelberg}) \rightarrow$$
$$\text{dbo} : \text{headquarter}(\text{db} : \text{EMBL},\ \text{db} : \text{Heidelberg})$$

Hence, we presented the entire integration process with the help of a single Reverb relation instance and discussed the various workflows and their effects.

3.2 WORKFLOWS

The proposed framework is capable of working under various different configurations. Depending on the type of the input OIE data set, one can opt for multiple *workflows*. In this work, we categorically differentiate between three workflows namely: wf_1, wf_2 and wf_3. In this section, we sketch the sequence of steps for each of these different workflows and illustrate them with the help of Figure 3. The direction of flows has been represented by directed edges in the figures.

- wf_1 : The basic mode where the input OIE relations are *not* clustered explicitly, and the IM module runs on the input set considering only one relation and its instances at a time. For instance, it considers *only* the instances of the Reverb relation "*originated in*" together and tries to find the instance matches for the OIE terms in those relation instances. The LU module runs to separate the input facts. The f^+ facts thus skimmed from the original input data set, is used to learn association rules which helps in relation matching. In this case, RM module is driven by the "r-RM" sub module. Although theoretically wf_1 is the trivial clustering case, in reality there is no clustering performed.

- wf_2: This is the next mode where we employ clustering, but the clustered relations are plugged into the rule based algorithm. As a matter of fact, we run the KB look up step and the clustering step in parallel and we feed them into the RM step. We use the clustered relations and their instances as input

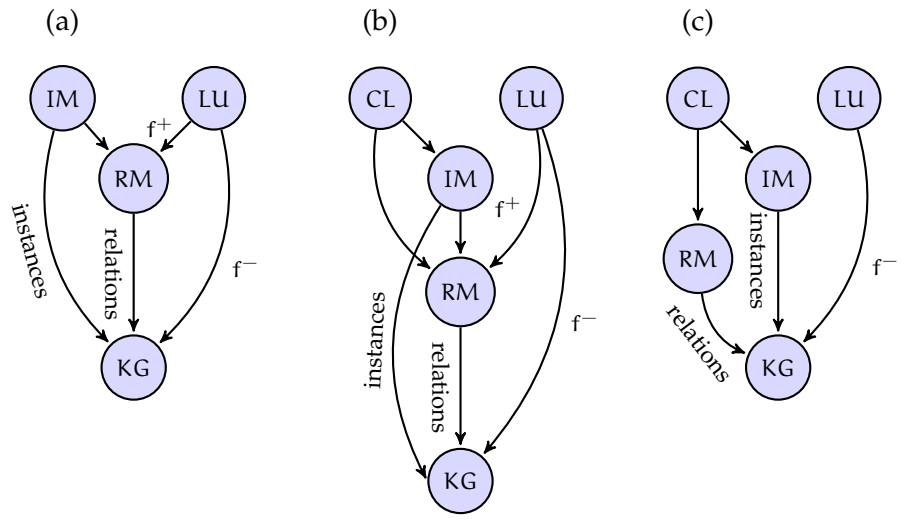

Figure 3: Schematic representation of (a) wf_1 (b) wf_2 (c) wf_3. (IM: Instance matching, RM: Relation matching, CL: Clustering, LU: KB Look up.)

to the IM module. The joint statistics over a group of synonymous relations is expected to contribute to a better instance refinement than the solitary relations in themselves can do. Observed closely, removing the "CL" module and its associated directed edges from Figure 3(b) reduces it to Figure 3(a). Thus, clustering is the only addition in wf_2. Here, we are employing the actual non-trivial clustering of relations. Reiterating, in this case also, RM module is essentially the "r-RM" sub module.

- wf_3: The final workflow where we *only* use the clustering based method for relation mapping. Hence, here no rule based module is required. The CL module and LU module runs independently as in wf_2. The result of clustering is a relation matching in this case and hence an explicit r-RM module is not required. So the RM module used here is already done via the clustering with DBpedia seed relations. Also note that in this mode, we do not explicitly use the f^+ triples.

This chapter provides a broad overview of our proposed system for generating knowledge from OIE systems. It highlights the prime modules and more importantly the interactions between them. We introduce the idea of different workflows, which are used in our set of experiments in the later sections. In the following chapters, we present detailed discussions on each modules. Each of the parts detail the three major contributions.

Part II

INSTANCE MATCHING

4

INTRODUCTION

The task of instance matching gained a lot of prominence with the advent of the Semantic web research. In this chapter, we introduce the general problem of instance matching and draw relevant correspondences with similar variants of the problem. In particular, we discuss some of the state of the art approaches for solving the task of instance matching and also briefly argue the differences we have with our task. Since, our work fits well into this research area, we would like to adhere to the problem of instance matching as conceived by the semantic web community. Although, this may seem to be new problem, but this is known to have its existence for some decades and have been extensively studied under various nomenclatures in different branches of computer sciences. In the scope of this work, we would briefly mention some of these close research areas.

4.1 RELATED WORK

4.1.1 *Record Linkage*

The aim of this is to find multiple records in multiple data sources, that are associated with the same entity. The survey paper from Gu et al., [GBVR03] defines it as: "*Record linkage is the task of quickly and accurately identifying records corresponding to the same entity from one or more data sources*". One of the pioneer works in this topic was done by Winkler [Win95] who devised an algorithmic approach of solving the task; a big improvement over the pre-existing manual solutions. Ever since, better and smarter methods have been proposed which ranges widely from single column similarity metrics [Coh98] and distance based multi-field matching [ACG02] to machine learning classifiers using support vector machines [BMC+03]. In the more recent years, Jin et al., [JLM03] have achieved remarkable results by projecting each tuple over the n-dimensional attribute space on which they used FastMap as the mapping algorithm.

In our context, the task of instance matching does not have any direct correspondence to the record linkage task. However, we dedicated a section on this due to the inherent similarity with the nature of the problem. Generally speaking, record linkage is concerned about matching correct instances across multiple sources, which is in a way relevant to the broader aspect of our task. Furthermore, we also want to emphasize that this genre of problem is not relevant only to the natural language or the semantic web community but exists for decades in other communities like databases.

4.1.2 *Instance Matching*

According to Rong et al., [RNX+12], "*The problem of discovering owl:sameAs links between pairwise data sources is called instance matching*". The definition mentions of "*owl:sameAs*"[1] link, which is the semantics for expressing equivalence of two different entities in the linked data/semantic web context and interestingly the definition itself classifies its application area. We must note the high similarity between the definitions of the entity linking task and instance matching task. The former is a more general area and suits well for textual resources. While the later speaks of a source and target data source. These data sources are published under the Resource Description Framework (RDF) [RNX+12]. And ideally each contains several millions of RDF triples as subject-predicate-object. There is a wide array of works on instance matching in the recent past. Works of Rong et al.[RNX+12] adopts a more sophisticated technique by transforming the problem into the binary classification problem and solving it by machine learning algorithms. Some have tried to enrich unstructured data in form of text with Wikipedia entities [MW08]. PARIS [SAS11] takes a probabilistic approach to align ontologies utilizes the interdependence of instances and schema to compute probabilities for the instance matches. The Silk framework [VBGK09] discovers missing links between entities across linked data sources by employing similarity metrics between pairs of instances. The wide range of works are not limited to the ones mentioned but extend well beyond. However, we had attempted to mention the pioneering works in this area and provided some conceptual background about the problem.

1 http://www.w3.org/TR/2004/REC-owl-semantics-20040210/#owl_sameAs/

4.1.3 *Entity Linking*

The task of instance matching has been discussed also under the notion of entity linking and mostly in the broader area of Natural Language Processing. Larson [Lar10] defines the problem statement as, "*entity linking describes the task of matching references to named entities found in natural language texts to a unique identifier, denoting a specific entity*". For over quite some time, researchers have made considerable efforts in solving the tasks of Entity Linking (EL) [JG11] and Word Sense Disambiguation (WSD) [Nav09]. This is a very broad area and it is impractical to mention every work. However, seminal work in entity linking includes contributions by Bunescu and Paşca [BP06] and Cucerzan [Cuc07], who focused on the usage of Wikipedia categories and global contexts, respectively. Lin et al. [LME12a] provide a novel approach to link entities across million documents. They take web extracted facts and link the entities to Wikipedia by means of information from Wikipedia itself, as well as additional features like string similarity, and most importantly context information of the extracted facts. Another work of Lin et al., [LME12b] deals with trying to link unlinkable entities in a corpus to Wikipedia. They use the evidence from the linkable entities to detect the fine grained type information of the unlinkable ones. This work is more aligned towards classical NLP and exploits the features from Google Books ngrams. Dredze et al. [DMR+10] achieved remarkable results using supervised approaches, in which they were able to link entities with missing knowledge base entries. Similarly for WSD, supervised systems have been shown to achieve the highest performance, although questions remain on whether these approaches perform well when applied to domain-specific data [AdS09]. Besides, recent work indicates that knowledge-based methods can perform equally well when fed with high-quality and wide-coverage knowledge [PN10, NP12]. The GROUNDER system by Fader et al., [FSEC] is yet another linker system, trying to find accurate references to entities in a text. This system clubs together two major signals, first, the prior information about the mention (some occurrence of *clinton* is more likely to refer to Bill Clinton than Clinton county) and second the contextual evidence. The priori information is extracted using the Wikipedia interlinked structure, which is much similar to our frequent sense model. They use cosine contextual similarity which can be defined as follows: if a document contains an entity mention then that document is highly likely to be similar to the actual Wikipedia article about that entity mention. For instance, a political document where clinton is mentioned

will have a higher cosine similarity with the wiki-page of Bill Clinton than the page on Clinton county. This is an efficient model to combine prior and context to efficiently disambiguate named entities. For us, we do not have the source page with every OIE extracts, for instance NELL does not have one. Hence, the above technique will have only the prior component functional in our problem setting.

Other entity linking systems like DBPEDIA Spotlight [MJGSB11], AIDA [HYB+11], exploit mainly context of the entities. Context is usually missing within the facts generated by OIE systems[2], making the task bit harder. In contrast to these approaches, we apply a method that uses Wikipedia anchor text as surface forms as introduced by Bunescu and Paşca [BP06]. This consists of a simple, yet high-performing baseline that provides us with high-quality seeds for our probabilistic approach. which is presented in the following chapters. In the survey paper by Shen et al., [SWH15], a three module structure of a typical entity linking system is outlined. These include (i) candidate entity generation, (ii) candidate entity ranking and finally (iii) unlink-able mention prediction. Our task adopts the first two approaches. The goal of this thesis is not to create another entity linking system, but use the general guidelines for designing one and use it as a part of our broader aim of knowledge base extension.

It is important to understand the fine line of difference between entity linking and instance matching. The former is more appropriate in the context of natural language texts with the named entities occurring in conjunction with other parts of speech. The task is to find (entity spotting) and link them to a reference KB (Wikipedia/FREEBASE etc). While the later is more relevant in the semantic web context, where there can be multiple data sources (not necessarily unstructured) and a bridge needs to be created between the identical entities. For instance in FREEBASE, `https://www.freebase.com/m/0jcx` refers to Albert Einstein and in DBPEDIA it is `http://dbpedia.org/page/Albert_Einstein`. A instance matcher would find that these two are same the entity.

4.2 PROBLEM STATEMENT

We formally introduce the problem of instance matching as relevant in this work. Note that this is the general formalism of the problem and is valid in the rest of this thesis. Let $\mathcal{S}$ denote the set of terms occurring as subjects and objects in

2 Usually this is the case, NELL gives extraction patterns, while REVERB presents the source URLs

the source data, i.e. the OIE input tuples (NELL and REVERB). And $\mathcal{T}$ denote the set of instances (subjects and objects) involved in some semantic relation in the target data source, i.e. the structured knowledge base (DBPEDIA). Now let $s \in \mathcal{S}$ and $t \in \mathcal{T}$, are two elements from these two data sets respectively. We denote they are same by associating them together with a partial function fn (which maps some elements of $\mathcal{S}$ to the elements $\mathcal{T}$). Intuitively, the function signifies that they both represent the same real world entity, or speaking in terms of the definition of an instance matching task, s is *sameAs* t. The underlying function $fn : \mathcal{S} \to \mathcal{T}$ dictates that each element s can be mapped to at most one element from $\mathcal{T}$, which is given by, $fn(s_1) \neq fn(s_2) \implies s_1 \neq s_2, \quad \forall s_1, s_2 \in \mathcal{S}$. Hence, the problem is about finding a set of pair wise elements one from each source, denoted as $\mathcal{M}$ and defined as,

$$\mathcal{M} = \{ (s, fn(s)) : (s, fn(s)) \in \mathcal{S} \times \mathcal{T} \} \tag{1}$$

Essentially, the pair element (s, t) is an instance from the set $\mathcal{S} \times \mathcal{T}$, since t is the mapping of s. However, we impose an additional restriction on the number of simultaneous pairings a particular s can have, which is set to at most 1. If there are no suitable element t, then s is essentially not paired and left blank ("-"). The pair $(s, -)$ is not explicitly added into the set $\mathcal{M}$. As a side note, this is one of the most important notations, which will be referred to repeatedly through out this work. For instance, $\mathcal{S}$ may have elements like {*usa*, *love*, *Bear cubs*, ...} while $\mathcal{T}$ may have the elements like {... `db:United_States`, `db:Purple_Heart`, `db:Love`, ...}. Our goal is to find the set $\mathcal{M}$ as {(*love*, `db:Love`), (*usa*, `db:United_States`), ...}. Intuitively, the pairing denotes that in some tuple where *usa* occurred as a subject, the exact knowledge base instance it referred to is `db:United_States`.

However, in reality, there can be some tricky situations. Often the same term occurs multiple times across the whole data set. For instance, a quick look up in the REVERB data would give us the following triples.

$$\begin{aligned} \text{was born in}(\text{Edward}, \text{England}) \\ \text{was offered the role of}(\text{RobertRedford}, \text{Edward}) \\ \text{is a Web Developer for}(\text{Edward}, \text{Yahoo!}) \end{aligned}$$

It is clear that the term *Edward* occurring both as subject and object across these different triples are probably not talking of the same person. According to the formulation above, we cannot have multiple mappings for the term *Edward*. We

employed a simple work around to this problem by assigning an unique identifier to every term that has multiple occurrences across the data. This unique identifier was an auto incremented numerical value concatenated to the end of the term text. Hence we had triples like

$$\begin{aligned} \text{was born in}(\text{Edward}_1, \text{England}) \\ \text{was offered the role of}(\text{RobertRedford}, \text{Edward}_2) \\ \text{is a Web Developer for}(\text{Edward}_3, \text{Yahoo!}) \end{aligned}$$

This alphanumeric representation of the terms allowed to uniquely determine the references to the KB by making explicit mapping pairs as (Edward_1, `db:Edward_VI_of_England`) and so on. Hence, the exact term sense in different triples can be easily determined.

4.3 DISCUSSION

It is quiet evident from the range of related works that there are a lot of context aware and supervised techniques available. Here, we briefly outline our work and draw essential similarities and differences with these state-of-the-art approaches as presented in the previous section. As a major part of our work, we perform instance matching across two data sources. The source is Open Information Extraction system outputs (NELL and REVERB in the form of non-urified triples) and the target being DBPEDIA. These triples often contain ambiguous and vague terms as subject/object. Hence, we focus on matching these terms to a structured knowledge base. We should note that our work falls in the intersection zone of the classical definitions of instance matching and entity linking task. We indeed try to define *owl:sameAs* links between entities across data sources (similar with instance matching task) but the entities are from OIE sources which are ambiguous (without URI) as in the case of natural language texts (similar with entity linking task). We do not use any new terminology for our work, but stick to instance matching. We adopt the modular design guidelines as defined by Shen et al., [SWH15] namely, entity generation and entity ranking. Once we do that, we try to find similar entities in a knowledge base. This part exactly corresponds to finding "*owl:sameAs*" links between two data sources.

However, there is a major difference with any of these prior works mentioned. Our initial goal was to achieve a context free translation from open domain extractions to a structured vocabulary. All the entity linking strategies use the context of the entity mentions to improve the quality. Context can be the surrounding text of the entity, the whole paragraph or even the outgoing links from the source page. This is very natural that such additional information will definitely enhance the linking procedure since they enhance the likelihood of the entity of being what it should be. But, with Open Information extracts, often such contexts are missing. This is not a general statement but for NELL it is the case while with REVERB the source URLs are provided. We wanted to design a complete and general framework which can work with bare minimum inputs; just in the form of the subject-predicate-object. We require no additional context but use the implicit context latent in the triple. For instance the REVERB triple *was also chairman of(Baer, the Centennial Committee)* already gives a hint that *Baer* is more likely to be a person. The key to our complete solution is the the extraction and exploitation of such latent/implicit contexts. This makes our solution very general.

5

BASELINE APPROACH

We employ most frequent sense of words in the Wikipedia corpus to design a baseline approach. This employs using Wikipedia anchors as surface forms of entities and the idea was introduced in the earlier works of Bunescu et al., [BP06]. For the purpose, we provide a very brief description of the anchor-link structure of Wikipedia in Section 5.1. In the subsequent section (Section 5.2), we exploit this intra-page links to present the details of our baseline algorithm. Finally in Section 5.3 we present an application use case where the base line methodology was applied to improve the instance matching task.

5.1 WIKIPEDIA: THE KNOWLEDGE SOURCE

In this initial approach, we use Wikipedia as an entity-tagged corpus [BP06] in order to bridge knowledge encoded in NELL or REVERB or any other OIE system with DBPEDIA. The Wikipedia has been a valuable source of rich information content for a wide range of scientific, academic and professional tasks. It is an exhaustive source of unstructured data which has been extensively used to enrich machines with knowledge [HNP13]. It consists of a huge collection articles covering various categories. Each article, often called as a wiki-page, consists of a clear and unambiguous title followed by a relevant textual content. The text within such pages may also contain "mentions" of other articles which are hyper-linked to other Wikipedia page. These mentions are called *anchor texts*. Hence, the complete Wikipedia can be considered as an inter-connected mesh of pages where every page has some outgoing links via the anchors to some other pages. And also, most of the pages have some incoming links from anchor texts from other wiki-pages.

The anchor-links structure of Wikipedia encode useful information about the page titles and the anchor texts leading to the respective pages. In particular, we can extract two essential pieces of information: first, which all anchor texts point to a particular page article. This has been illustrated in Figure 4. Each of the rect-

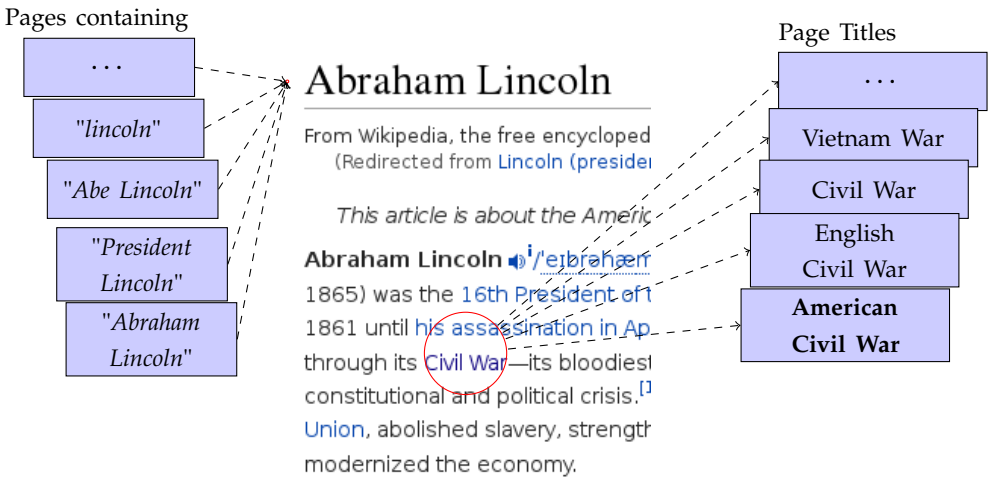

Figure 4: A representation of hyper links within Wikipedia, representing incoming links to an article via different anchor texts and outgoing links from anchor texts within a page to different Wiki-articles.

angular structures denote individual Wikipedia pages. The ones on the left are some arbitrary pages with the respective mentions of entities within their individual text contents. These mentions are essentially anchor texts and hyperlink to the actual page of Abraham Lincoln. And second, we are more interested in finding to which all wiki pages can a particular mention (anchor text) can point to. This is represented with the bunch of pages on the right of the figure. As observed in the figure, in the given context, the anchor *Civil War* links to the wiki-page on American Civil War (shown in bold). But, it is interesting to observe that the anchor can also refer to a set of other articles depending on the context in which *Civil War* is used. The other possible pages are listed in the figure but definitely not limited to only these.

5.2 METHODOLOGY

5.2.1 *Links Extraction*

This can be considered as pre-processing step where we get the exact counts of the number of outgoing links from an anchor text and likewise the number of incoming links to a wiki page. Wikipedia provides regular data dumps and there are off-the-shelf preprocessing tools to parse those dumps. We used WikiPrep[GM07, GM06] for our purpose. WikiPrep removes redundant information from the original dumps and adds more statistical information to it. It creates

anchor	Article	Link count
civil war	American Civil War	7220
civil war	Civil War	911
...	...	...
civil war	Second Barons' War	1
lincoln	Lincoln, England	1844
lincoln	Lincoln, Nebraska	920
...	...	...
lincoln	List of Archdeacons of Lincoln	1

Table 1: A sample output from the tool Wikiprep, showing the number of outgoing links from a given anchor text to the possible Wikipedia articles. Shown for the anchors *civil war* and *lincoln*.

its own custom XML data dump with additional information like number of categories, number outgoing links and URLs the list of categories, list of outgoing links, list of URLs and more. More exhaustive details can be found on the tool download page[1]. In our work, we are primarily interested in the link counts, namely the frequency of anchor text labels pointing to the same Wikipedia page. In Table 1, we present our example anchor texts along with the total links connecting them to the respective articles. It is interesting to observe that both have as low as 1 link connecting some articles. This gives us an impression that the possible list of articles can be quiet extensive.

Referring to the table, an easy interpretation of this data is as follows: out of all the outgoing links from the anchor text *civil war*, 7220 of them pointed to the Wikipedia page on "American Civil War", 911 to the page "Civil War" and so on. Intuitively, every hyperlink from an anchor to a respective article signifies that the anchor text is a way of expressing the article referenced to. This is called as *surface form* representation. For instance, *civil war* is a surface form of the *American Civil War*. It is also a surface form for "English Civil War" or "Vietnam War" or others. This ability of the surface forms to possibly refer to multiple articles suits our problem setting perfectly. We find an one-to-one correspondence to our mapping task with OIE terms. These terms exhibit polysemous nature i.e. the ability to refer to multiple real world entities under different contexts. Hence, we can safely assume that the OIE terms behave exactly like the Wikipedia anchor texts and our extracted links data set can be very well used for further uses.

1 http://www.cs.technion.ac.il/~gabr/resources/code/wikiprep/

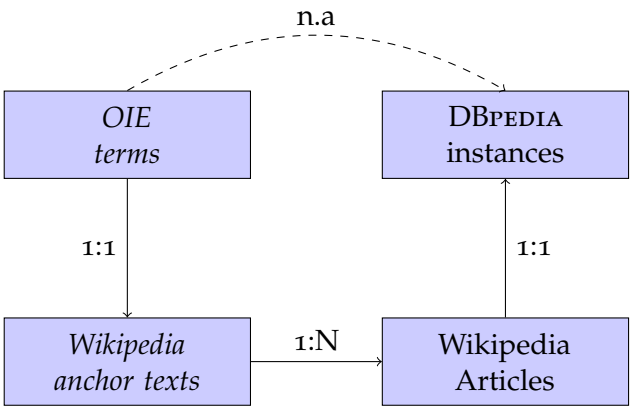

Figure 5: Illustration of the Wikipedia-based most frequent sense baseline method for the instance matching task.

5.2.2 *Candidate Generation and Ranking*

Our aim is to generate a set of possible DBpedia instances for the terms occurring as subject and object in every OIE triple. To do so, we exploit the Wikipedia link structure. The broad idea can be very well illustrated with the Figure 5 where the final target is to find the ranked candidate list of DBpedia instances for a given OIE term. This is depicted as the dashed arrow with a 'n.a' (meaning not available). As seen from the figure, since a direct mapping is not available, we take a detour using Wikipedia statistics to achieve our goal. It is important to observe the connecting arrows and their directions. For instance, the OIE terms bear a direct correspondence with the anchor texts we extracted from the Wikipedia corpus. This is bidirectional with 1:1 mapping degree which means every OIE term will have a single anchor text which is essentially the exact text. The mapping degrees have been marked along the edges of the figure. Similarly, there is also a corresponding DBpedia entity for each Wikipedia article [BLK+09], and hence we observe the similar 1:1 degree of mapping between Wikipedia articles and DBpedia instances. Finally, the only different mapping is from anchor text to page articles, where an anchor can refer to a multitude of pages, denoted as a 1:N mapping and already seen in the example anchor texts in Table 1.

With the exact count of links at our disposal, we only need to solve the candidate ranking for every anchor text to a set of possible Wikipedia articles. As seen in Table 1, the output from WikiPrep can often be a long list of anchor-article pairs and some of them having as low as just one link count. As an initial step,

we define a likelihood measure for each surface form texts to refer to a particular Wiki page article or likewise a DBpedia instance. For any given anchor text, the fraction of articles the links points to is proportional to the probability that the anchor term refers to the particular article [SC12]. Formally, suppose some anchor e refers to N articles ranging from $A_1 \ldots$ to A_N with $n_1, \ldots, n_N$ respective links counts, then the conditional probability P of e referring to A_j is given by,

$$P(A_j|e) = n_j / \sum_{i=1}^{N} n_i \tag{2}$$

where n_j is the link count connecting A_j and e [DNMP13]. It is intuitive that, higher the number of inter connecting hyper links between the page and the anchor text, higher would be the likelihood of the anchor text to be a surface form representation of the the article. Based on these estimated values, we have a top-k ranked list of all possible articles an anchor text might point to. For each OIE triple, we take the terms occurring as subject and object, and apply the procedure above. This gives a ranked list of possible candidate DBpedia instances for the OIE terms. We perform extensive experiments with this base line approach and in Section 7.4, we present some of its performance statistics on the publicly available data sets of Nell and Reverb. Especially, we validate our choice of k with both the data sets and use the value for the rest of the experiments in this thesis.

5.3 USE CASE

We discuss an application scenario where the most frequent Wikipedia sense has been used for disambiguation purpose. We were interested to gauge the effectiveness of the frequency based simplistic baseline approach against a graph based entity disambiguation method. Formally, given an OIE triple of the form $p(s, o)$ we want to find the matching set $\mathcal{M}$ consisting of elements $(s, f(s))$ and $(o, f(o))$ where $f(s)$ and $f(o)$ are DBpedia instances and mappings of the terms s and o respectively (refer Expression 1). The notations used here has been already introduced as a formal definition of the instance matching problem statement in Section 4.2. Furthermore, this example also gives an impression of the baseline method. In particular, we compare and contrast to three methodologies: first, the frequency based entity linking approach; second, we introduce a graph based approach which exploits DBpedia as an exploratory knowledge graph; and third, we propose a Combined approach, which incorporates the frequency-based and

with the graph-based approach. We present them in brief in the following. The definitions and some of the ideas have been already published in our earlier work [DS14]

Frequency based Entity Linking

This is a minor extension of the frequency based approach we presented in Section 5.2. The broader idea was to employ the frequency based baseline both on the subject and object terms independently and evaluate the OIE translation to the DBpedia vocabulary. Note that in this section of our work, we do not focus on mapping the OIE relations to corresponding DBpedia properties. Since, this use case primarily focuses on the effectiveness of the baseline method without considering the joint distribution of entities or entity-relation dependencies that usually occurs in triples. If E_{sub} denotes the top-k candidates for subject s and analogously E_{obj} for object, then for every combination of $|E_{sub}| * |E_{obj}|$ we get DBpedia candidate pairs. Now, since the frequency based method does not consider any joint distribution of the subject/object, hence the mapping of subject is independent of that of the object. This allows us to apply the independence rule in computing the probability of the mapping a subject and object to DBpedia instances. Hence, for every candidate pair, a joint probability P_{freq} can be defined as

$$P_{freq} = P_{sub} * P_{obj} \tag{3}$$

where, both P_{sub} and P_{obj} define the probability of the subject and object mapping to a DBpedia instance respectively and obtained from Equation 2. And the subscript $freq$ denotes the frequency based approach.

Graph-based Entity Linking

This exploits the DBpedia ontology itself and employs graph exploration method. This technique is not the main contribution of this work, but has been introduced by Schuhmacher et al., [SP14]. We use this technique in our experimental setup to perform a comparative study. The details of this approach is beyond the scope of this thesis. However in a nutshell this approach exploits the latent contextual connectivity between OIE terms instead of relying just on the most frequent entity. For instance, while trying to disambiguate the Nell triple *actorstarredinmovie(kevin bacon, footloose)*, it tries to look into the adjacent nodes of `db:Kevin_Bacon` in the DBpedia knowledge graph and deduces that it has higher chances of being associated with `db:Footloose_(1984_film)`. It defines an infor-

mation content measure between an entity pair based on their semantic similarity in the graph. It finds the cheapest cost of paths between all candidate pairs out of $|E_{sub}| \times |E_{obj}|$. The path cost between two entities is calculated as the sum of the edge costs along their undirected connecting path and is normalized as probabilities to P_{graph}. The exact expression of the probability is outside the scope of this discussion. We used the implementation of the probability computation from the works of Schuhmacher et al., [SP14].

Combined Entity Linking

This is a combined approach where we aggregate the powers of both the frequency based and knowledge graph based into one model. This approach is motivated by the fact that both the former approaches have individual weakness, but can complement each other. The former exploits the empirically obtained frequency data about common surface-form-to-instance map- pings, however, it cannot incorporated the information that subject and object should most likely be related in a semantic world. But this information is exploited by the graph-based approach which finds this vague relationship between subject and object in DBpedia however, ignoring the important frequency information. Consequently, we opt for a linear combination of the two approaches and select the subject-object combination with the highest combined probability

$$P_{comb} = \lambda P_{graph} + (1 - \lambda) P_{freq} \tag{4}$$

where the weighting factor lambda (λ) is a normalizing constant lying between 0 and 1. Initially we set this value as 0.5, thus giving equal influence to the graph and the frequency information. With this combination, we give preference to those subject-object combinations, having individually high likelihoods and which are also closely semantically related in DBpedia. Later in Section 7.5 we experimentally validate our claim about the choice of lambda. We also present detailed empirical results and compare each of the three approaches. Our conclusion was that a combined approach outperforms the individual effects of either one of frequency based or graph based methods.

6

PROBABILISTIC APPROACH

6.1 INTRODUCTION

Analysis with the frequency based method revealed that the top-1 candidate for each OIE term is often accurate in correctly deciphering its reference. But, often there can be cases where the correct KB reference to an OIE term is not the most frequent sense (top-1) but other senses ranked lower in the list of possible candidates. For instance in the NELL triple *actorstarredinmovie(al pacino, scarface)*, the most frequent sense would refer the object to the DBPEDIA instance `db:Scarface_(rapper)`, but the correct sense is `db:Scarface_(1983_film)` which is the second best sense. It is not possible to find these kind types correct candidates with the most frequent sense approach. This phenomenon motivated us to explore an improved instance matching technique which considers the possible matching candidates beyond just the most frequent senses. However, in designing a better method we set out few primary assumptions:

- the OIE triples are the only input to the system/framework. This immediately restricts us from exploiting any additional context information (in the form of surrounding texts, knowledge graph representations or paragraph/-source from where the OIE triple was extracted from.)
- the OIE triples do not necessarily maintain any fixed schema. This is not so strict assumption since, often the open domain extracts do not maintain any hierarchy of relations or concepts. NELL is an exception in this regard.

These two broad guidelines, implicitly push towards finding a very general solution. One must note that NELL has no source extraction information (at least a few for the Wikipedia based extractions) but has a schema on its own. On the other hand, REVERB has source URLs but no schema. Thus there is no fixed standard that the OIE data sets in general adheres to. Hence, the key to a general solution lies in formulating with the bare minimum i.e. just the assertions. This is the core feature of our instance mapping algorithm; it is capable to work with OIE triples with almost no special input requirements. The primary methodologies and the

idea has been re-used from our previous publication [DMP14].

Key to our method is the synergistic integration of (i) information about the entity types the OIE terms might refer to and (ii) a method to find a global, optimal solution to the mapping problem across multiple extractions on the basis of statistical reasoning techniques. These two phases are highly intertwined, thus, we alternate between them by means of an iterative approach. We present this broad idea using a NELL triple. For instance, the term *tom sawyer* occurring within the triple *bookwriter*(*tom sawyer*, *twain*) can be mapped to a set of DBPEDIA entities: the fictional character Tom Sawyer (*Tom_Sawyer*), the actual book written by Mark Twain (*The_Adventures_of_Tom_Sawyer*), or the many screen adaptions of the book (*Tom_ Sawyer_(1973_film)*). Given the term and its candidate entities, each of the candidates can be a plausible reference to the occurrence of *tom sawyer* in the context of the example triple. Estimating the likely types (DBPEDIA concepts) which can fit as a domain or range of the semantic relation (*bookwriter*) would allow us to further filter out entity candidates which are incompatible with the types. For instance, estimating that *bookwriter* is a relation defined between instances of types `dbo:Book` and `dbo:Writter` would allow us to reduce the search space for the correct mappings for *tom sawyer* in DBPEDIA by concluding that it is probably not a film or a fictional character, but rather an instance of a book. While estimating the likely types (in terms of DBPEDIA concepts) we are faced with two challenges:

- First, it is not enough to determine `dbo:Writer` as range of *bookwriter*, because many entities writing books are not explicitly typed as `dbo:Writer` but are of different types (e.g. `dbo:Athletes` can also write books). Hence, we need a weight distribution indicating that the type `dbo:Writer` is more probable than `dbo:Politician` and `dbo:Politician` is more probable than `dbo:Location`. Since, we start with no additional context information, this concept distribution for domain and range would allow us to set an implicit restriction on the instance mappings.

- Second, a weighing scheme is heavily dependent on the quantification of each of the domain/range values. We require a confidence score (weights) for the domain and range type of an OIE relation term using the DBPEDIA concepts. This has to be normalized and would directly indicate a likelihood of a DBPEDIA concept to be a domain or range of a given OIE relation.

Finally, exploit the above information effectively in determining the actual mapping assignment. Intuitively, a better domain/range restriction would lead to a better instance matching. These refined instance matches can be again used to improve the domain/range likelihood estimates. Thus, acquiring type information and simultaneously producing high-quality mappings, are two highly intertwined problems. One reinforces the other in a positive way. We identified this dependency and accordingly incorporated this into our overall approach. This explains the need for bootstrapping the mapping process, iteratively until saturation point.

6.2 METHODOLOGY

Based on the above general idea of instance mapping, we now discuss the three prime components of our solution. In Section 5.2.2 we generate a set of potential mapping hypotheses for the OIE terms, along with a likelihood score for each mapping. We use these candidate mappings to derive the entity type information (Section 6.2.2). We use the two weights to model the problem as an inference task within Markov Logic Network (Section 6.2.3). Keeping in mind the intertwined nature of the task, we propose a bootstrapping algorithm (Section 6.2.4) that generates better mapping hypotheses and refines the weight distribution for the learned types over a repeated number of iterations.

6.2.1 *Candidate Generation*

Candidate generation refers to the creation of a set of possible DBPEDIA instances for a given OIE term (occurring as subject or object). Hence, in general a particular OIE term would have possible mappings to top-k DBPEDIA instances. We call this a matching hypothesis or simply mappings and denote by $\mathcal{H}$. It is very important to distinguish between this hypothesis set and the final set of instance matches. The later is a set of final mappings ($\mathcal{M}$) which is the end product of the instance matching algorithm. This has been formally defined Equation 1 along with its associated details in Section 4.2. We tend to define the hypothesis set keeping close parity with the former equation. We introduce a strict subset of the set of KB instances ($\mathcal{I}$), as $\mathcal{I}'$ which is defined for every OIE term s as $\mathcal{I}'_s$ and defined as,

$$\forall s \in \mathcal{S}; \quad \mathcal{I}'_s = \{ x : x \in \mathcal{I} \} \ \textit{where} \ |\mathcal{I}'_s| \leqslant k \tag{5}$$

where, k defines the top-k candidates for each OIE term we are interested in. In our case, $k = 5$. We have presented detailed evaluations and examples in the experiments section (Section 7.4 in particular). Hence for each OIE term s we maintain a set of its individual top-5 DBpedia candidates. This allows us to formally define the hypothesis set $\mathcal{H}$ as,

$$\mathcal{H} = \left\{ (s, \mathcal{T}'_s) : s \in \mathcal{S}, \left| \mathcal{T}'_s \right| \leqslant k \right\} \tag{6}$$

This is an important step for the whole design since the hypothesis restricts the set of possibilities to be finite by eliminating noisy candidates. Hence it is called a hypothesis since this serves us a prior belief set about the OIE term and its likely set of mappings.

Our candidate hypothesis set for a term is determined by the top ranked possible candidate. The ranking is done using the likelihood score. The basis for computing this score is based on the probability (presented in Section 5.2) values we estimate from Wikipedia corpus. Hence, we convert the raw probability values to a log-linear form in accordance to the underlying Markov Logic Network which we employ for this probabilistic approach. We did our initial experiments with the actual probability values, however the results were not so promising. For a given probability p, the log-linear representation is denoted by the function $\mathtt{logit}(p)$ and given by,

$$\mathtt{logit}(p) = \log \frac{p}{1-p} \tag{7}$$

The base of the logarithm is not really important as long as it is greater than 1. This function is called the inverse of the sigmoid function[1]. We see that the function $\mathtt{logit}$ is undefined for a probability value of 1. Hence we introduce an infinitesimal smoothing factor ϵ (usually in the order of 10^{-8}) to the above equation which modifies the function as,

$$\mathtt{logit}(p) = \log \frac{p-\epsilon}{1-p+\epsilon} \tag{8}$$

1 If a function f transforms a variable x to the domain $f(x)$, then an inverse function f' is such that it transforms an element from $f(x)$ back to x i.e. $f'(f(x)) = x$. A sigmoid function for a random variable x is denoted by $\mathtt{sig}(x)$, and defined as $\mathtt{sig}(x) = \frac{1}{1+e^{-x}}$. Substituting p with $\frac{1}{1+e^{-x}}$ in $\log \frac{p}{1-p}$ would gives us the value x. Hence the reason to call the $\mathtt{logit}$ function an inverse of sigmoid function

Let us consider a sample REVERB extractions *son of (Apollo, Zeus)*. We have presented just for the subject term its hypothesis set. We maintain a similar collection for the object as well.

$$\begin{aligned}
+0.435 &: && \mathit{apollo} \rightarrow \texttt{db:Apollo} \\
-1.774 &: && \mathit{apollo} \rightarrow \texttt{db:Apollo_program} \\
-3.231 &: \quad \mathit{apollo} \rightarrow && \texttt{db:List_of_Apollo_asteroids} \\
-3.999 &: && \mathit{apollo} \rightarrow \texttt{db:Apollo_(ballet)} \\
-5.598 &: && \mathit{apollo} \rightarrow \texttt{db:Holden_Apollo} \\
\ldots &: && \mathit{zeus} \rightarrow \ldots
\end{aligned} \tag{9}$$

The entire set $\mathcal{H}$ consists at most k candidate mappings for each of the OIE subject and object terms. The cardinality constraint in Equation 6 restricts the maximum mapping per term to be k.

For the actual candidate generation, we employ the exact methodology as described in Section 5.2.2. The rational for the OIE term and anchor text equivalence has been argued in details in Section 5.2.1. Hence, we directly apply a Wikipedia frequency based lookup on these individual subject/object terms. However, a mapping might not be always possible since there can be cases when either of the terms or sometimes both the terms cannot be linked to some KB instance. We must note that we have only top-5 candidates for each OIE term. This is motivated by the fact that there is necessarily no remarkable improvement in recall beyond top-5. We have performed extensive experiments in support for our claim and those have been reported in Section 7.4.

As a final remark for this section, we must emphasize that the candidate generation methodology is actually independent of any particular generation scheme. The prime idea is to have a set of weighted candidate mappings for each OIE term to a KB. One can generate similar mappings with entirely different weighing schemes or techniques. However, the method should generate meaningful weights that can be interpreted in a probabilistic context.

6.2.2 *Probabilistic Type Generation*

This module caters to the task of probabilistically finding the type restrictions for a given OIE relation. The examples and values used in this section are with respect to the NELL data set but this does not affect the generality of the overall approach. This section deals with the first two challenges we mentioned in the introductory section of this chapter (Section 6.1). Our goal here is to find a probable list of DBPEDIA concepts which can fit best as domain or range for each of the OIE relation. We necessarily do not make any assumption on the precedence of some concepts over others, rather consider all to be a possible domain/range. Hence, none of the concepts are eliminated out on an initial assumption basis but allow the type weights to re-rank them. The key component in our approach is a probabilistic formulation where each mapping, each concept has an effect on the final output set of instance matches (i.e the set $\mathcal{M}$). This is a more realistic modeling of the problem, since often in real world we encounter cases where a range of instance types can be a part of a semantic relation. For instance, consider the NELL relation *hasspouse*(*, *), the arguments are instances of type `dbo:Person`. But, this does not restrict some instance to be a part of this relation which is explicitly not typed as `dbo:Person`. Those may be `dbo:Scientist`, `dbo:Architect` or even `dbo:Farmer`. In the process we introduce a hierarchical data structure which we call as *alpha-tree*. This tree enables to re-rank DBPEDIA concepts with higher likelihood of being a domain or range for the given OIE relation.

We already have the initial hypothesis set $\mathcal{H}$ generated for each OIE term in each OIE relation. Let us focus on one the example triple *son of (Apollo, Zeus)*. So, $\mathcal{H}$ will consist of at least the two entries: {*(apollo,* $\mathcal{T}'_{\texttt{apollo}}$*)*, *(zeus,* $\mathcal{T}'_{\texttt{zeus}}$*)*} (Equation 6), where $\mathcal{T}'_{\texttt{apollo}}$ and $\mathcal{T}'_{\texttt{zeus}}$ are the hypothesis set of top-k candidates, as seen in the Example 9 above. At this stage, the best guess we have for the possible types is provided by the best candidate i.e the top-1. This leads us to select a subset of the hypothesis set $\mathcal{H}$ which we refer to as $\mathcal{M}$. Note that, this set is not different from the earlier definitions of $\mathcal{M}$. It follows the exact definition of Equation 1 and consists exactly one-to-one mapping pairs: {(*apollo*, `db:Apollo`), (*zeus*, `db:zeus`)}, where `db:Apollo` essentially is the top-1 (most frequent) element from the set $\mathcal{T}'_{\texttt{apollo}}$ and the similar constraint holds for `db:zeus`. Thus $\mathcal{M}$ forms a subset of $\mathcal{H}$. We exploit this hypothesis set $\mathcal{M}$ in our subsequent steps and perform iterative updates on this set $\mathcal{M}$. This repetitive procedure over-writes the instance

mappings where top-1 is not the correct choice with some other lower ranked candidates in the top-k candidate list.

The class associations or simply the types of A and B are used as markers for the domain and range restrictions of *p*. This is an implicit way of exploiting the context even though we started from a simple SPO triple. These domain/range restrictions for the property p puts an indirect check on the possible set of KB instances that the OIE subject/object may possibly map to. Here, we create a distribution over the possible KB concepts which are likely to be the domain and range for a given OIE relation or a cluster of relations. This is not about finding the exact KB concept (occurring as domain or range) but a likely one, so given the OIE relation "*was the son of*", it should have `dbo:Person` as a more probable range than `dbo:Building`.

In this regard, we distinguish between the direct and indirect type of an instance. Class C is a direct type of instance a, denoted by $C(a)$, if there exists no sub class D of C, denoted by $D \sqsubseteq C$, such that $D(a)$ exists. We count the direct type of each mapped DBpedia instance in $\mathcal{M}$. We use a DBpedia SPARQL endpoint for querying the direct types[2] of the candidates in $\mathcal{M}$. Finally, we obtain a distribution over the direct type counts for the possible concepts, both for the domain and range of *p*. Figure 6 depicts a snippet of the concept hierarchy for the range of the property `bookwriter`, where the nodes represent the concepts and the numbers (in non-bold) denote their direct type counts. The sum of the counts at a particular level do not add up to their parent node's count, since we are only counting the direct types of each instance.

Key to our method is the observation that an appropriate weight distribution helps us establish whether a certain candidate mapping is correct or not, according to the type information. Considering the most frequent class as a hard domain/range restriction could potentially perform well, but this would fail to consider other instances writing books, e.g. philosophers, researchers or even athletes. On the other extreme, it seems rational to also count the indirect types or to propagate the count for a direct type recursively up to the parent nodes. However, this would result in a type restriction that takes only top-level concepts into account

2 We distinguish between the direct and indirect type of an instance. Class C is a direct type of instance a, denoted by C(a), if there exists no sub class D of C, denoted by $D \subset C$, such that D(a) exists.

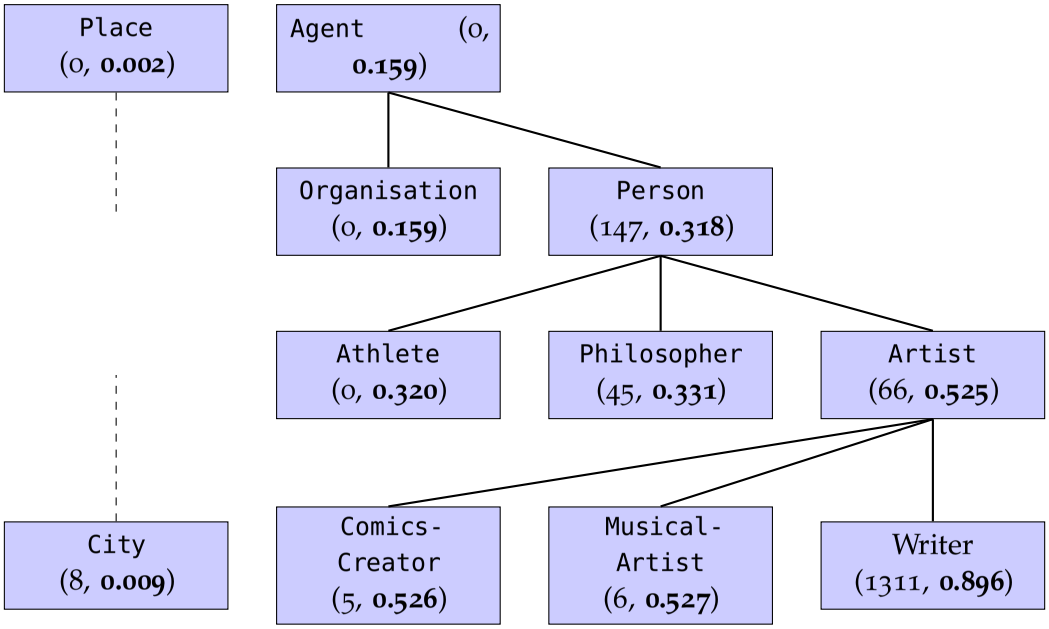

Figure 6: Counting and weighing the range types of the `bookwriter` property. Each concept is accompanied by the counts of the direct types and the normalized S_d score for $\alpha = 0.5$ (shown in bold).

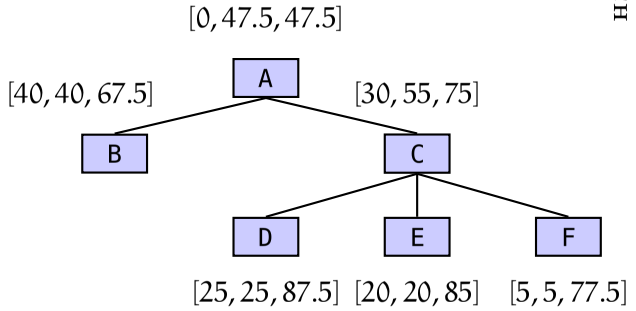

Figure 7: Propagating direct counts in the alpha tree. Shown scores are $[S_o, S_u, S_d]$. This is a working example presenting the score computation mechanism. The node labels are concept names in some hierarchy. Figure 6 presents the final scores

and completely disregards the finer differences expressed in the lower levels of the hierarchy. For example, a writer is more likely to write a book compared to an athlete. Accordingly, we opt for a hierarchical scaling of weights along the levels, such that the most likely class in the hierarchy is determined by the instance distribution of both its children and parent.

Hence, we propose a simple formulation to compute an appropriate score for each concept n. First, we introduce the *up-score*, S_u which is defined as

$$S_u(n) = S_o(n) + \alpha \sum_{c \in child(n)} S_u(c)$$

where $child(n)$ denotes the children of n, $S_o(n)$ refers to the direct type count and α is a constant, which works as a *propagation factor* with $\alpha \in [0, 1]$. The computation of this score starts from the leaf nodes, which are initialized with their direct count $S_o(n)$. S_u is defined recursively and, accordingly, the S_u score for n is computed based on the S_u score for the children of n. Furthermore, we also define a *down-score* S_d as

$$S_d(n) = \begin{cases} S_d(parent(n)) + (1 - \alpha)S_u(n) & ; n \neq \textit{top-concept} \\ S_u(n) & ; n = \textit{top-concept} \end{cases}$$

where $parent(n)$ denotes the parent node of n. We refer to the concept hierarchy annotated with the S_d scores as the so-called *α-tree* in the rest of the work. Figure 6 presents the hierarchical representation of the range concepts for the NELL relation *bookwriter*. The normalized scores in bold represent the final score computed with the α-tree model. The scores in the figure are computed using the same principle.

We present in Figure 7 an example illustrating a simple hierarchy consisting of six concepts. We wanted to present the working methodology of the tree hence we used simplified node names for easy understanding. The relevant scores for $\alpha = 0.5$ are shown adjacent to the nodes as $[S_o, S_u, S_d]$. This example illustrates that the sibling classes D, E and F, eventually, have the highest S_d scores, while the order among them, as defined by S_o, is still preserved in the order defined by S_d. As a final step, the down-scores are normalized by dividing them by the sum of the direct counts S_o for each node. With respect to Figure 7, the sum of S_o is $40 + 30 + 25 + 20 + 5 = 120$ and so the normalized S_d for node D, say,

is estimated as a probability of $87.5/120 = 0.73$. Obviously, the choice of the constant α is critical to achieving the desired result. Setting $\alpha = 0$, neutralizes the effect of child nodes on parent nodes. In this case we have $S_d(n) = S_o(n)$, which means that the type hierarchy is completely ignored. On the other extreme, setting $\alpha = 1$ propagates the scores to the full degree, but always creates the same scores for all concepts in the same branch. With respect to the example shown in Figure 6, we would learn that all concepts in the `Agent` branch have the same weight, while there are no differences between the concepts `Organisation` and `Writer`. In Section 7.6.1 we discuss the choice of the optimal α and report about experimental results related to different α values.

6.2.3 *Formulation with MLN*

Markov Logic Networks (MLN)[RD06] are a framework for combining probability theory and first-order logic. Probabilities allow to capture the uncertainty associated with the problem while first-order logic helps to capture the logical aspects of the problem. Formally, a MLN is a set of weighted first-order logic formulae which are essentially a conjunction of atoms. We use MLN to choose the most probable mapping of individual OIE terms to KB instances, given the uncertain information we have embedded in the α-tree and the weighted mapping hypothesis set, $\mathcal{H}$. Under a set of constants, formulae instantiate into a ground Markov network where every node is a binary random variable resulting from grounding the atoms in the formula and hence called a ground atom. In our task, we use three atoms to build the MLN.

$$\begin{aligned} \mathrm{map}(X, Y),&\ \text{match OIE term X to dbpedia instance Y} \\ \mathrm{hasType}(C, Y),&\ \text{type C of the dbpedia instance Y} \\ \mathrm{pAsst}(P, X_s, X_o),&\ \text{the OIE triples} \end{aligned}$$

The arguments within the above atoms are all variables. They take the values from the set of constants in the form of OIE terms, the set of DBpedia instances that are the potential mapping candidates and DBpedia instance types. For instance, `map`(*Apollo*, `db:Apollo`) is a ground atom and is essentially a binary variable. A conjunction of these formulae, depicts a real world. We can have several ($\approx .2^{\#\mathrm{groundings}}$) network states for different boolean assignments of the ground atoms. Every such state is also called a *world*. In those worlds different

formulae hold true and if some do not, then the world is penalized according to the weights attached with the violating formulae. As a result, that world becomes less likely to hold. Note that unweighted formulae can instead never be violated. According to [RD06], the probability of a world x is defined as

$$P(X = x) = \frac{1}{Z} exp(\sum_i w_i n_i(x))$$

where F_i is a first-order logic formula; w_i is the weight attached to the formula, $n_i(x)$ is the number of true groundings of F_i in x and Z is the normalizing factor (also called *partition function*) given as $\sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$. In Appendix B we presented a brief background on MLN with a simple example showing how the probabilities for a particular world is computed.

While modeling a real world scenario, we want some rules to be always followed strictly hence the concept *hard* formulae. Ideally, these are formulae with infinite weights. Intuitively, their counterpart are called *soft* formulae. In our task, we employ both hard and soft formulae. The hard formula for our model is incorporated in the form of a restriction on the maximum number of mappings a particular OIE term can have. This is formally stated as

$$|\texttt{map}(X, Y)| <= \quad 1$$

which denotes, for all possible instantiations of the `map` atom, that every OIE term X can have at most one mapping to a DBpedia instance Y, i.e. we force the mapping to be functional. Since the mapping predicate is usually weighted, we introduce a wrapper predicate $mapConf(X, Y, w)$ over the actual predicate which states as,

$$w : mapConf(X, Y, w) \rightarrow map(X, Y)$$

where, w is the weight assigned to the mapping. We have encountered these weights in Example 9. The reason to introduce such a predicate is to make the mapping not an absolute truth but a weighted clause which can be violated.

To additionally take type information into account, we extend our model with two soft formulae for each possible combination of OIE relation P and DBpedia

type C. The first formula reflects a weighted domain restriction and the second formula reflects a weighted range restriction.

$$w_d : \texttt{hasType}(C, y_s) \wedge \texttt{pAsst}(P, x_s, x_o) \wedge \texttt{mapConf}(x_s, y_s, f) \rightarrow \texttt{map}(x_s, y_s)$$
$$w_r : \texttt{hasType}(C, y_o) \wedge \texttt{pAsst}(P, x_s, x_o) \wedge \texttt{mapConf}(x_o, y_o, f) \rightarrow \texttt{map}(x_o, y_o)$$

Note that P and C are replaced by constant values, while x_s, x_o, y_s, and y_o are quantified variables. $w_d(P, C)$ and $w_r(P, C)$ are the weights denoting the likelihood of C being the domain or range respectively, and this is obtained from the α-tree computation as discussed in Section 6.2.2. Furthermore, we must remember that, map(X, Y) atoms are weighted and are softly weighted as shown in first order rule 10. The rule formulations shown above are basically weighing 2 weights in particular, the type weights from α-tree; and the mapping weights from the hypothesis candidates. The eventual body of the first order logic rule is the mapping predicate. If the type weight $w_d(P, C)$ is high, it makes the mapping of OIE subject term to its DBpedia counterpart more likely.

Based on our model, we compute the MAP state, i.e., the most probable world which coincides in our scenario with the most probable mapping. In particular, we want to select a better mapping rather than just choosing the top-1 candidate for each OIE term. The MAP inference is conducted with the RockIt system [NNS13]. RockIt computes the most probable world by formulating the task as an Integer Linear Program. The solution of the resulting optimization is the MAP state of the MLN. So far, we used a subset of $\mathcal{H}$, namely $\mathcal{M}$ (introduced in Definition 1), as the input for computing the α-tree. Obviously, the quality of $\mathcal{M}$ directly impacts the quality of the resulting α-tree. At the same time, a better α-tree, can be expected to result in a better MAP state. We present a background of MLN in Appendix B with a simple example and show how our work exploits the strong formalism of MLN.

6.2.4 *Bootstrapping*

The result of running a MAP inference on the MLN is a set of refined mappings. We explore how to use these mappings as input for constructing the α-tree again and to use the resulting α-tree as input to recompute the MAP state. This step tries to improve the mappings in an iterative fashion. We start with $\mathcal{M}$ (having top-1

Algorithm 1 Algorithm for Bootstrapped Instance Matching

1: **procedure** BOOTSTRAP
2: $\mathcal{H} \leftarrow$ set of mapping hypotheses
3: $\mathcal{M}_0 \leftarrow$ *top-1*$(\mathcal{H})$
4: $i \leftarrow 0$
5: **while** $\mathcal{M}_i \neq \mathcal{M}_{i-1}$ **do**
6: $i \leftarrow i + 1$
7: $\mathcal{T}_i \leftarrow \mathtt{alphaTree}(\mathcal{M}_{i-1})$
8: $\mathcal{M}_i \leftarrow \mathtt{computeMAPState}(\mathcal{H}, \mathcal{T}_i)$
9: **return** $\mathcal{M}_i$ ▷ filtered output

mapping for each of the OIE terms) and allow the MLN to decide the MAP state. This eliminates impossible mappings, hence the resulting set of refined mappings are again used to create a weighted concept hierarchy and a new set of improved concept weights. A higher concept weights makes the likely mappings stronger eliminating more incorrect ones and making the type weights even stronger in the next iteration. This is the exact loop-effect we mentioned in the introductory sections. Our algorithm terminates when no more refinement is possible. It is also important to note the repeated update step on the final mapping set $\mathcal{M}$ (line 8 in Algorithm 1), since with each iteration, a newer and refined set of instance matchings are updated on this set. We report about the results for each such iterations in the experiments section (Section 7.6.2). This completes our IM module with a final set of refined mappings. It must be observed that, the method here is general and is independent of any specific mapping generation scheme. Furthermore, it also not specific to DBpedia, any other target KB with a well-defined concept hierarchy should suffice.

7

EXPERIMENTS

In this chapter, we perform extensive evaluation of our proposed probabilistic methodology (as introduced in Chapter 6). In particular, we compare our approach against a baseline method which is essentially the strong Wikipedia most-frequent-sense based method (detailed in Chapter 5). We observe the performance of our proposed method on both the NELL and REVERB data sets. For a principled evaluation, we created a gold standard which has been presented in details in Section 7.2. In the subsequent sections we present an analysis of the gold standard with details on its creation procedure. The rest of this chapter is divided as following: Section 7.1 broadly introduces the two data sets with some statistical analysis on them to get an impression of the underlying patterns within the data sets. Section 7.2 introduces a gold standard for instance matching task, in Section 7.3 we define the evaluation metrics as relevant for our case. Subsequently, Section 7.4 reports respectively on the performance values for the baseline (most frequent sense) approach, the probabilistic approach (Section 7.6) and the combined approach using graph based method (Section 7.5). We perform a complete comparative analysis with all the techniques in Section 7.7. Finally, we make some concluding remarks in Section 7.7.

7.1 DATASET

7.1.1 *Format*

We briefly present the data formats maintained by the two state-of-the-art OIE systems. We first present NELL and then discuss about REVERB data. For better illustration, we present one complete tuple from each of the data sets and show the individual column values for them.

NELL

NELL releases regular data sets[1] which they often term as *Iterations* and every

1 http://rtw.ml.cmu.edu/rtw/resources

data dump at regular intervals are marked with "Iteration#" where # denotes the number of the iteration release. In these experiments we used Iteration#920 for our analysis with NELL. The file contains millions of tuples which contains the details of every extracted fact. Each tuple is tab-separated files with the different column data headers. We consider one complete example tuple, and state its value under the following different headers.

- **Entity**: the subject term in the triple. Often the term is accompanied with a concept prefix, for instance *concept:person:maria_shriver*. This however necessarily does not indicate the category membership (*person*) of the term *maria_shriver*.
- **Relation**: the OIE relationship between the entity (subject) and the value (object). E.g. *concept:hashusband*
- **Value**: the value part of an entity. For easy understanding, this is the object in the relation and hence this can be another entity as well. The concept membership is also not valid for the objects. E.g. *concept:male:arnold_schwarzenegger*
- **Iteration of Promotion**: the iteration at which NELL promoted this fact to be true. This is a non-negative integer value denoting the number of iterations of bootstrapping NELL had undergone. This says a lot about the internal mechanism of NELL. Every extracted fact is not immediately marked as true with high confidence. It requires more iterations to gather enough evidence (extraction patterns) to "promote" it. E.g. *920* in our case
- **Probability**: a score for the fact denoting the degree of truth. E.g. *0.96875*
- **Source**: a summary of the provenance for the belief indicating the set of learning subcomponents (CPL, SEAL, etc.) that had submitted this belief as being potentially true. This is also associated with the time stamp of extraction. E.g. *MBL-Iter:920-2015/04/20-02:23:59-From+ErrorBasedIntegrator(CPL(maria_shriver,arnold_schwarzenegger))*. Usually it is UTF-8 encoded.
- **Entity literalStrings**: The actual textual representations of the subject term that NELL had encountered. Often its a list of values. E.g. *"Maria Shriver", "MARIA SHRIVER", "maria shriver"*
- **Value literalStrings**: similar to the previous one, the actual textual strings for the objects. E.g. *"Arnold Schwarzenegger", "arnold schwarzenegger", "Arnold schwarzenegger", "arnold-schwarzenegger"*

- **Best Entity literalString**: of the set of strings in the Entity literalStrings column, the best way to describe the concept. E.g. *Maria Shriver*
- **Best Value literalString**: similar description as the former, but for Value literalStrings. E.g. *Arnold Schwarzenegger*
- **Categories for Entity**: the set of categories/concepts which NELL believes the entity to belongs to. NELL maintains a hierarchy of concepts and this value informs about the concept type of the subject. In our example this field was blank.
- **Categories for Value**: similar as above but for values. E.g *concept:actor, concept:celebrity, concept:personnorthamerica, concept:male*
- **Candidate Source**: specific provenance information describing the justification as why NELL believes in the fact. E.g. *[CPL-Iter:821-2014/03/09-11:30:25-<token=maria_shriver, arnold_schwarzenegger>arg2+is+married+to+arg1, arg2+met+his+wife+arg1, arg1+and+her+husband+arg2, arg2+and+wife+arg1, arg2+and+his+wife+arg1]*. This example has been reformatted by converting its original UTF-8 encodings to characters.

REVERB

It also maintains a very similar structure. The publicly available data set[2] contains several million tab-separated rows with the following headers. We follow the similar example pattern we did before with NELL.

- **Extraction id**: a numerical value denoting the extraction number, can be considered as a tuple identifier.
- **Argument 1**: this is the subject term occurring in the REVERB fact. Since REVERB extracts binary relations of the form $p(s, o)$, the subject s is the first argument in such a relation p. E.g. *Achilles*
- **Relation**: the semantic relation phrase connecting the subject and object in an extracted fact. Note that, the subject or object necessarily need not be entities, they can be literals as well. E.g. *is the son of*
- **Argument 2**: this is the object term occurring in the REVERB fact, i.e. the object o as mentioned above. E.g. *Peleus and Thetis*
- **Argument 1 - normalized**: argument 1 reduced to its normal form. This is necessarily not stemming or lemmatizing. E.g. *achilles*.

2 http://reverb.cs.washington.edu/reverb_clueweb_tuples-1.1.txt.gz

- **Relation - normalized**: normalizes the relation. In this case it stems and lemmatizes the actual phrase. E.g. *be the son of*
- **Argument 2 - normalized**: same as normalizing argument 1 but for the object terms. E.g. *peleus and thetis*
- **distinct sources**: the number of distinct sentences the extraction was made from. Here it was just 6.
- **confidence**: the confidence value assigned to denote the degree of truth in this extracted fact. E.g. *0.94090*
- **source**: REVERB does not provide the actual sentences, but the URLs of the web-pages it extracted the fact from. If there are multiple sources, then they are separated by "|". There were 6 websites listed for this tuple. E.g. *http://www.classicsunveiled.com/mythnet/html/heroes.html* |...|....

In general, NELL is richer in its data format than REVERB. The former contains detailed extraction patterns and algorithms for each tuple. These are of immense importance for designing an information extraction framework. REVERB on the other hand is bit too coarse grained in providing the URLs. One has to explicitly mine for the patterns in the respective pages. Another striking difference we observe is in the relations. NELL has the relations normalized and are less noisy. But, REVERB relations are non-normalized and in natural language form, as we encounter in any text.

7.1.2 *Pre-Processing*

Both the data sets provide a wide array of values to choose from. However, keeping in mind our broader approach of designing a context agnostic framework, we perform a column based and tuple based pruning.

First, for the column based pruning, we disregard columns which do not contribute to our work process, in particular we considered only the values which gives us a subject-relation-object triple. We chose to ignore the source information for the same reason. With NELL we adhere to the "Best Entity", "Value" representations and "Relation" columns. Although we chose the best literal form, it necessarily did not mean that the other literal representations of the terms are ignored. While implementing, these various case sensitive inputs, are accordingly handled. Also, the information related to the entity-concept association ("Cate-

gories") was disregarded. First, NELL provides a disclaimer on its exactness. Second, this representation is very specific to NELL only and cannot be considered a general pattern for any other OIE system. Furthermore, the relations presented in the data set are also affiliated with a prefix, for instance as observed in the example *concept:agentcreated*. The prefixes are removed as they necessarily do not contribute to anything valuable for our case. With REVERB we select the "normalized Argument 1", "normalized Argument 2", and "normalized Relation". REVERB did not require a lot of term cleaning, since they maintain a data format which is more aligned to natural language text as compared to that of NELL.

Second, for the tuple based pruning, we needed to eliminate triples which we do not require down the pipeline. Some were due to domain constraints, for instance, triples related to medical domain were eliminated from NELL since these contain entities not covered by DBPEDIA. Since instance matching process tries to refer OIE entities to target KB vocabulary, the term related to the uncommon domains will not be eventually mapped. Furthermore, our analysis with the data set revealed that there exists almost 304,963 instances for the relation *haswikipediaurl*. This was interesting since it was already a disambiguated link to the Wikipedia page. Initially, we planned to exploit this and use it as a ready-made gold standard because a NELL term with an explicit Wikipedia url is a matching in itself. However, it had two major problems. First, to use this as a gold standard, we should have a relation instance (other than *haswikipediaurl*) involving the subject or the object. Even if we find such a relation instance, it necessarily does not imply that both the terms are referring to the same real world entity. Hence, we completely removed relation instances of *haswikipediurl* from the data set. Additionally, we also remove triples having literals as values. This was mainly done keeping parity with the core idea of our instance matching algorithm. Since the matching is driven by a probabilistic type information of the instances, it makes sense to consider those OIE relations which are defined between entities and not the ones which contains literal values as subjects or objects.

7.1.3 *Instance Statistics*

In this section we present a snapshot of some numbers related to the two major OIE data sets we used in our experiments (Table 2). This is a pre-requisite for actually starting to work with the data, since we get an initial impression about

Attributes	Nell	Reverb
Version	NELL.08m.920.esv.csv.gz	reverb_clueweb_tuples-1.1
Facts (unpruned)	2,310,307	14,728,268
Facts (pruned)	1,974,872	3,480,752
Pruning factor (facts)	14.519%	76.367%
f^+ facts	54,064	71,237
f^- facts	445,717	1,999,115
un-mapped facts	1,475,091	1,410,400
Relations (unpruned)	15396	664746
Relations (pruned)	294	474325
Pruning factor (relations)	98.090%	28.645%

Table 2: Snapshot of the data set statistics for NELL and REVERB.

the latent patterns of the two data sets and also numerical values to indicate our practical possibilities. We refer to Table 2 for the following discussion.

Version: The release number of the data used. This is definitely not the latest but downloaded in late April, 2015. Since NELL releases the datasets in a frequent interval, the latest one soon gets replaced by a newer version in a short span of time. This, does not affect our setup or analysis. Since our initial experiments were with iteration #725 which had approximately 1.95 million facts. With the above mentioned iteration we have 2.3 million (≈18.57% rise in a span of three years). Hence the growth rate is slow over iterations (assuming linear growth, this is 0.09% in every iteration) and hence we do not sacrifice much in few iterations. However, for REVERB we do not have such problems with iterations. The project homepage hosts two datasets: a larger version of extractions which is more exhaustive and web scale, and a smaller set of extractions performed with with only Wikipedia.
Facts (unpruned): The total number of triples present in the data set. This is before the column/tuple based pruning is performed.
Facts (pruned): The remaining number of OIE facts after the pre-processing step (details presented in Section 7.1.2)
Pruning factor (facts): This represents the fraction of original input triples removed as a result of the preprocessing. Hence, for instance approximately 14.5% of the NELL inputs were truncated. This is a simple ratio of the number of triples removed to the original number of triples. This number is quiet high for REVERB and which actually shows the huge occurrence of facts with literal/numeric values. These are only around 3.5 million facts actually involve in some entity-vs-entity relationships. This simple value provides a deep insight into the two data

sets. We expected REVERB, which is few magnitudes larger than NELL, to yield a much larger pruned fact set. However, REVERB has a huge number of facts with numerical/date-time entries.

f^+ **facts**: These are the number of OIE facts which have an analogous KB assertion.

f^- **facts**: These are the number of OIE facts which do not have an analogous KB assertion. These two values are not relevant with the instance matching module but more evident with the relation matching part. For a consolidated and a comparative view we decided to include them in Table 2.

un-mapped facts: These are the facts for which either of the subject or object terms was un-mapped. This means, there were no DBPEDIA instances which could be linked with the terms in the first place. This is an important figure since it tells about the entities involved in the facts. Such entities are essentially non-mappable to the target KB. We must note its fine line of difference with f^- facts. The later involves entities which could be mapped to the KB but did not have a connecting semantic relation between them. A quick parity check: the sum of the number of f^+, f^- and un-mapped facts should be the number of pruned facts.

Relations (unpruned): The number of distinct OIE relations occurring in the original data set.

Relations (pruned): The number of distinct OIE relations occurring after preprocessing the data set.

Pruning factor (relations): This is similar to the pruning factor as mentioned before but here we report the effect on the relations. It is interesting here to notice that almost 98% relations from the original NELL input data set were removed but still there was only $\approx$ 14% drop in facts count. This indicated that a lot of NELL relations had very few relation instances[3]. Recollecting, a relation instance of p is a fact of the form $p(A, B)$, where A and B are respectively OIE subject and object terms. Using the same numerical calculation, we can conclude that the 28.6% loss in REVERB relations accounted for $\approx$ 59 instances for each relation pruned.

This interesting aspect with the relation instance counts motivated us to make deeper analysis of the underlying patterns in the data sets. We now employed the pruned data set for further experiments. We first created a list of all the OIE relations occurring in the pruned data sets. These were the exact number of rela-

3 This can be re-stated with actual numbers. A pruning of 15,102 (15396 to 294) relations causes a drop in 335,435 number of facts (2,310,307 to 1,974,872). This means, on an average, 15,102 relations had 22 instances each ($=\frac{335,435}{15,102}$)

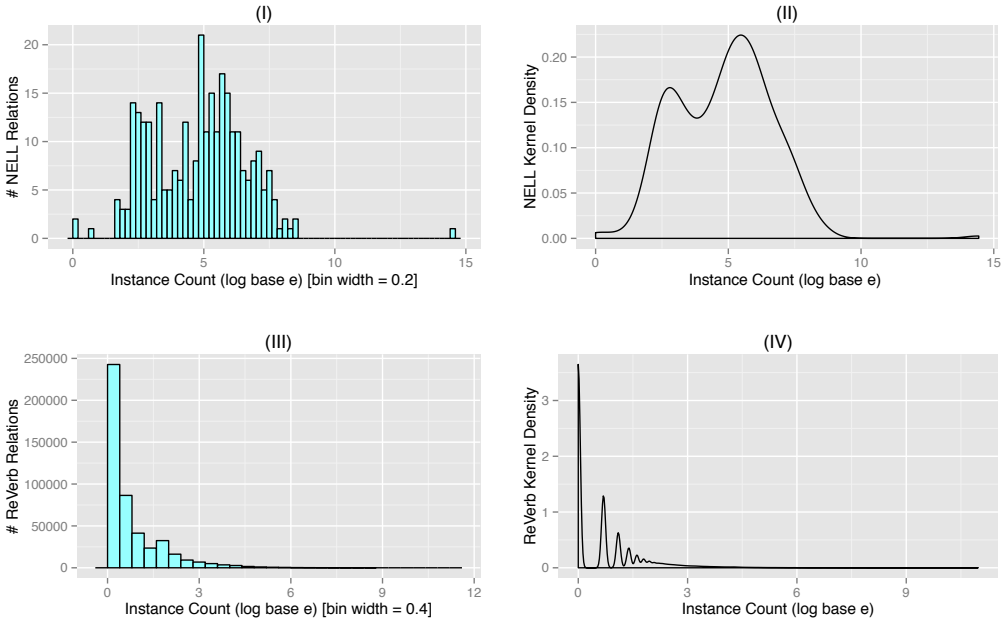

Figure 8: Histogram plot for (I) NELL relation instance counts (III) REVERB . Relation instance distribution approximated with a kernel density curve for (II) NELL and (IV) REVERB.

tions as reported in the row named "Relations (pruned)" in Table 2. For each such relation, we collected the number of instances associated with the given relation in the pruned data set. For instance, (*producesproduct*, 454) is one such counted value from the NELL data set, denoting there were 454 instances with the relation *producesproduct*. Based on these instance counts we partitioned the whole data set into *bins* of equal instance count ranges. For NELL we had the minimum instance count of 1 and maximum was 1858015. We converted them to log scale (natural logarithm with base e) and partitioned into bins (0.2 bin-width for NELL and 0.4 for REVERB). For each bin, we counted the number of OIE relations belonging to that group. That means for some arbitrary bin $[i, i+1]$, we get all the OIE relations having (logarithm of the)instance counts within that bin. This gives us a distribution over the relations with respect to instance counts. Referring to Figure 8(I) the highest peak at the bin range [5, 5.2] denotes there are $\approx$ 21 NELL relations having instance counts within the range of [148, 181] (e^5 to $e^{5.2}$). For the REVERB data set Figure 8(III) we observe that there were 242715 REVERB relations with just 1 instance each (bin e^0 to $e^{0.4}$) and which explains a peak for that bin. However, we observe that for REVERB data set we set the bin-width at 0.4. This

Top predicates	Instances	Random predicates	Instances
generalizations	1867287	**personleads-organization**	718
proxyfor	8004	countrylocatedin-geopoliticallocation	986
agentcreated	4592	**actorstarredinmovie**	1025
subpartof	5331	**athleteledsportsteam**	390
atlocation	4282	personbornincity	203
mutualproxyfor	2785	**bankbankincountry**	596
locationlocatedwithinlocation	2096	**weaponmadeincountry**	564
athleteplayssport	2044	athletebeatathlete	176
citylocatedinstate	2697	**companyalsoknownas**	161
professionistypeofprofession	2677	**lakeinstate**	322
subpartoforganization	1966		
bookwriter	3855		
furniturefoundinroom	2199		
agentcollaborateswithagent	1762		
animalistypeofanimal	1546		
agentactsinlocation	1759		
teamplaysagainstteam	3370		
athleteplaysinleague	3058		
worksfor	1629		
chemicalistypeofchemical	2120		

Table 3: The 30 most frequent predicates found in NELL. The set of predicates we randomly sampled for the gold standard are in bold.

was done for pure aesthetic reasons.

We also present in Figure 8(II, IV) the density curves for the distributions of the data sets. The major peaks are attained at the points where we have more number of relations. Also, the scale on the y-axis for these two sub figures are now normalized from 0 to 1. These figures clearly show a difference in the distribution patterns for the two data sets. REVERB is heavily skewed while NELL maintains a more normalized distribution pattern. The primary target for this exploratory data analysis was to capture the inherent differences within the two data sets. REVERB displayed signs of skewed distribution while NELL was more well distributed. This wide pattern differences in the data sets allows us to better motivate our claim of designing a generalized framework capable of handling any kind of OIE inputs.

7.2 INSTANCE MATCHING GOLD STANDARD

For a principled evaluation, we devise a framework for the instance mapping module. Since, we worked majorly with two OIE data sets, we created two different gold standards for the purpose. The primary target for the gold standard was to contain correct subject and object mappings for a set of sample OIE triples to exact DBpedia instances. In this gold standard, we do not cater to the mapping of the relations, but just the instances. This section presents the gold standard creation technique with reference and examples from the Nell data set. The Reverb gold standard is created in a similar way.

We proceed with the pruned data and we employ it to create a frequency distribution over the relations. In Table 3, we list the 30 most frequent Nell relations. Since the gold standard should not be biased towards predicates with many assertions we randomly sampled 12 predicates from the set of predicates with at least 100 assertions (highlighted in bold in the table). For each one of those relations we randomly sampled 100 triples. We assigned each relation and the corresponding list of triples to an annotator. Hence, each annotator received a complete set of Nell relations with 100 relation instances. We first applied the method described in Chapter 5 to generate possible mapping candidates for each of the Nell subject and object within each triple. In particular, we generated the top-5 mappings, thereby avoiding generation of too many possible candidates, and presented those candidates to the annotator. Note that in some cases our Wikipedia based method could not determine a possible mapping candidate for a Nell instance. In this case, the triple had to be annotated without presenting a matching candidate for subject or object or both. However, in the cases where instance candidates were available, the annotators were allowed to select one of the candidates which to their understanding best reflected the essence of the original Nell triple. In general, the entire annotation task could be broadly classified into the following scenarios,

(i) One of the mapping candidates is chosen as the correct mapping, i.e., the simplest case.

(ii) The correct mapping is not among the presented candidates (or no candidates have been generated). However, the annotator can find the correct mapping after a combined search in DBpedia, Wikipedia or other resources available on the Web.

#	Nell-Subject	NellObject	DBP-Subject	DBP-Object
I	stranger	albert-camus	The_Stranger_(novel)	Albert_Camus
		1st cand.	Stranger (comics)	Albert_Camus
		2nd cand.	Strange (Hilary Duff song)	-
		3rd cand.	Characters of Myst	-
		4th cand.	Stranger (Electric Light Orchestra song)	-
		5th cand.	Stranger (The Rasmus song)	-
II	gospel	henry_james	?	Henry_James
		1st cand.	Gospel_music	Henry_James
		2nd cand.	Gospel	Henry_James_(basketball)
		3rd cand.	Urban_contemporary_gospel	Henry_James,_1st_Baron...
		4th cand.	Good News (Christianity)	Henry James (British Army officer)
		5th cand.	Gospel (liturgy)	Henry James (biographer)
III	riddle_master	patricia_a__mckillip	The_Riddle-Master_of_Hed	Patricia_A._McKillip
		1st cand.	-	Patricia_A._McKillip
IV	king_john	shakespeare	King_John_(play)	William_Shakespeare
		1st cand.	John, King of England	William Shakespeare
		2nd cand.	King John (play)	Shakespeare quadrangle
		3rd cand.	King John (1899 film)	Shakespeare, Ontario
		4th cand.	John, King of Denmark	William Shakespeare's plays
		5th cand.	John Balliol	List of WS screen adaptations

Table 4: Four annotation examples of the `bookwriter` relation (we have removed the URI prefixes). For the sample III, we had just one mapping for the object and none for the subject.

(iii) The annotator cannot determine a DBpedia entity to which the given Nell instance should be mapped. This was the case when the term was too ambiguous, underspecified, or not represented in DBpedia. In this case the annotator marked the instance as unmatchable ("?").

Table 4 shows four possible annotation outcomes for the Nell relation *bookwriter*. Example I explains the cases (i) and (ii). In the object mapping the correct candidate was the one from the candidates, while the subject had no matching candidate from the provided list and hence was provided by the annotator by external help. Example II is a depiction of cases (i) and (iii). The annotators were unable to find the correct mapping for the term *gospel*. Example III presents a scenario when no mapping candidate has been generated for the Nell term `patricia_a__mckillip`. The fourth example shows that the top match generated by our algorithm is not always the correct mapping, but might also be among the other alternatives that have been generated.

7.3 EVALUATION METRICS

For the evaluation purpose, we resort to precision, recall and F-measure. However, in this section, we briefly re-visit the definitions and present their definitions as used in our evaluation scenario. As already introduced before, if $\mathcal{M}$ refer to the mappings generated by our algorithm, and G refer to mappings in the gold standard. The gold standard mappings are also generated in the form of pairs. Precision is defined as

$$\begin{aligned} \mathrm{prec}(\mathcal{M}, G) &= \frac{|\mathcal{M} \cap G|}{|\mathcal{M}|} \\ \mathrm{rec}(\mathcal{M}, G) &= \frac{|\mathcal{M} \cap G|}{|G|} \\ F_1(\mathcal{M}, G) &= \frac{2 * \mathrm{prec}(\mathcal{M}, G) * \mathrm{rec}(\mathcal{M}, G)}{(\mathrm{prec}(\mathcal{M}, G) + \mathrm{rec}(\mathcal{M}, G))} \end{aligned}$$

We use these definitions to also define prec@k, rec@k and F_1@k. k denotes the number of candidates considered for each of the subject/object term. Hence, prec@k is essentially the fraction of top-k retrieved mappings that are correct. Likewise, rec@k defines the fraction of correct mappings that are in the top-k candidates. It must be noted that instance mappings which were marked with a "?" were not considered in the gold standard G. In the following scenario, we

use the examples presented in Table 4 and walk through the precision and recall computations.

Example 2.
Case I: at $\boldsymbol{k = 1}$
$|\mathcal{M}|$ = *size of the retrieved instance mappings. Should have been 8 for the 4 triples with both subject and object terms mapped to* DBpedia *instances, but the mapping riddle master* $\rightarrow -$ *is not generated by* $\mathcal{M}$*. Hence,* $|\mathcal{M}| = 7$*.*
$|G|$ = *size of the gold standard. Also should have been 8, but gospel* $\rightarrow$ *"?" is not considered. Hence also* $|G| = 7$*.*
$|\mathcal{M} \cap G|$ = *number of instance mappings correct both in* G *and* $\mathcal{M}$*. This is 4. Clearly,* $\mathcal{M}$ *maps stranger* $\rightarrow$ *Stranger_(comics) instead of the novel. Hence, wrong.*
Therefore, at k=1, $\texttt{prec@1} = 4/7 \approx \mathbf{57\%}$ *and* $\texttt{rec@1} = 4/7 \approx \mathbf{57\%}$*.*

Case II: at $\boldsymbol{k = 3}$
$|\mathcal{M}|$ = *The total number of candidates in all the four examples for all the subject and object terms. An easy count shows this to be 17.*
$|G|$ = *This does not change compared to the former explanation and stays at 7.*
$|\mathcal{M} \cap G|$ = *On including top 3 candidates instead of just 1, the cardinality of this intersection improve. Observe closely the mapping king john* $\rightarrow$ *King_John_(play) is now generated and matches* G*. Hence, this value grows to 5.*
therefore, at k=3, we have $\texttt{prec@3} = 5/17 \approx \mathbf{29\%}$ *and* $\texttt{rec@3} = 5/7 \approx \mathbf{71\%}$*.*

Note that precision and recall are not the same in general, because $|A| \neq |G|$ in most cases. We have them to be same in the Case I of Example 2, but this is purely coincidental. We observe with this simple example set that precision usually tends to be higher for lower values of k. It can be expected that prec@1 will have the highest score and rec@1 will have the lowest score. In the later parts of this section we present and discuss detailed empirical results which validates our claim.

When generating the gold standard, we realized that finding the correct mappings is often a hard task and sometimes even difficult for a human annotator. We also observed that the problem of determining the gold standard varies strongly across the properties we analyzed. We noted that the fraction of instances actually possible to be mapped across different relations vary quiet much.

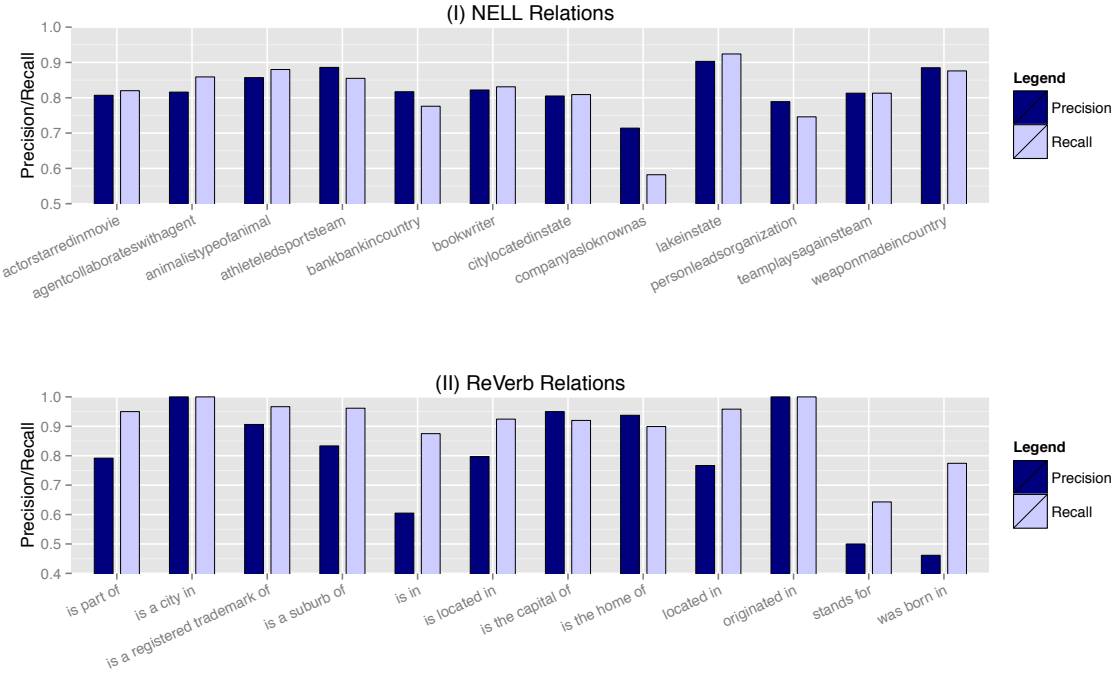

Figure 9: prec@1 and rec@1 of the Wikipedia most frequent sense based method for sample of (I) NELL and (II) REVERB relation instances.

7.4 RESULTS: BASELINE METHOD

Here we report on the naive baseline approach which exploits the Wikipedia frequent sense. As the initial setting, we keep $k = 1$, and run the baseline method against the gold standard[4]. The baseline approach applies the most frequent mapping to the OIE terms and generates a set of mapping outputs in the form of the set $\mathcal{M}$ where each OIE term is mapped to at most one DBPEDIA instance. In Figure 9, we show the precision and recall values obtained. The figure has been generated for both NELL and REVERB relations which were sampled for evaluation purpose (Section 7.2). Precision and recall vary across the predicates with `lakeinstate` having the highest precision for NELL. *is a city in*, *originated in* having the highest precision values for REVERB. Using average, for NELL the baseline method achieved a precision of 82.78% and an average recall of 81.31% across all the predicates, giving us a F_1 score of 81.8%. For REVERB the precision was 77.72% and recall was 90.60% which gives a F_1 score of 84.24%.

4 Please refer to Appendix C for datasets

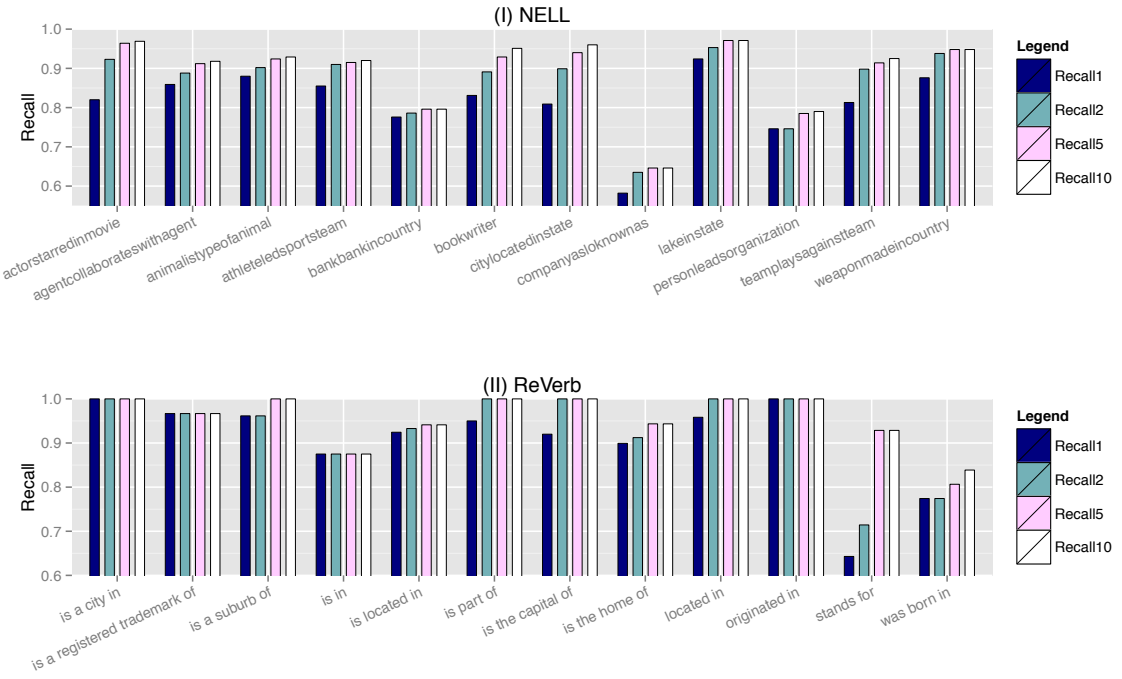

Figure 10: rec@k variation for k = 1, 2, 5, 10 for (I)NELL(II) REVERB

As the next step, we wanted to investigate the effect of k on the top-k candidates for the OIE terms. We ran the baseline with different values of k. Particularly, we chose values of k as 2, 5 and 10. In Figure 10, we show the values for rec@2, rec@5 and rec@10 compared to rec@1 (the recall values reported in Figure 9). For all the sample relations, we observe a similar trend. A rise in the recall scores and then eventual saturation after a certain value. By considering more possible candidates with increasing k, every term gets a better chance of being matched correctly, which explains the rise in recall for lower values k. Particularly, for some relations like that of *bookwriter*, we observe increasing recall even beyond 10. However, it must be noted, that for most of the predicates the values tend to saturate after rec@5. This reflects that after a certain k any further increase in k does not alter the correct mappings, since our algorithm already provided a match within top-1 or top-2 candidates.

As the final analysis, we wanted to capture a generalized trend in the behavior across all the relations. In Figure 11, we analyze our baseline algorithm with increase in values of k. For each setting of k we compute the average precision, recall and f-measures and plot those. The most interesting aspect of this exper-

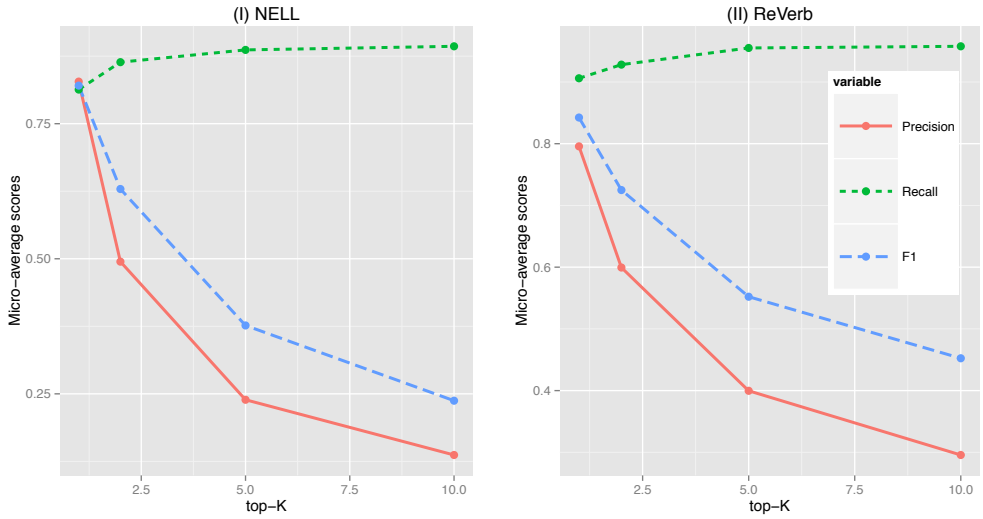

Figure 11: prec@k, rec@k and F_1 for (I) NELL (II) REVERB.

iment was to observe the recall variation. We expected it to increase over the increasing values of k, we observed our expected trend for both the data sets. Furthermore, we are also interested in the value of k for which the number of additionally generated correct mappings in $\mathcal{M}$ is negligibly small compared to the mappings generated in $\mathcal{M}$ for $k+1$. We plot the average values of the precision, recall and F_1 scores over varying k. We captured the values for both the data sets and presented them. For NELL we attained the best F_1 score of 82% for $k = 1$ and the recall values tend to saturate after $k = 5$. Similarly, for REVERB we attained the maximum F_1 of 84.24% at k=1. Furthermore, an important aspect revealed in this result is the gradual saturation of the recall scores after $k \geqslant 5$. This marks a recall saturation point for our experimental setup. Intuitively, we were not able to improve the number of correct mappings by any further beyond $k = 5$. Or in other words, beyond $k = 5$, the chances of finding across the correct mapping falls drastically. If its there, it is mostly within top-5.

However, there are ways to further improve the recall of our method like, for instance, by means of string similarity techniques – e.g., Levenshtein edit distance. A similarity threshold (say, as high as 95%) could then be tuned to consider entities which only partially match a given term. Another alternative would be to look for sub-string matches for the terms with middle and last names of persons. For instance, `hussein_obama` can have a possible match if terms like

	Frequency			Graph			Combined		
	P	R	F_1	P	R	F_1	P	R	F_1
actorstarredinmovie	80.7	82.0	81.3	89.8	91.2	90.5	91.4	92.8	**92.1**
agentcollaborateswithagent	81.6	85.9	**83.7**	69.3	72.9	71.1	81.6	85.9	**83.7**
animalistypeofanimal	84.0	88.0	85.9	62.4	64.1	63.3	85.2	87.5	**86.3**
athleteledsportsteam	88.6	85.5	87.0	87.0	84.0	85.5	91.7	88.5	**90.1**
bankbankincountry	81.7	77.6	**79.6**	68.3	64.8	66.5	81.7	77.6	**79.6**
citylocatedinstate	80.5	80.9	80.7	81.5	81.9	81.7	86.0	86.4	**86.2**
bookwriter	82.2	83.1	82.6	83.8	84.7	84.2	87.6	88.5	**88.0**
personleadsorganization	78.9	74.6	76.7	78.4	74.0	76.1	84.8	80.1	**82.4**
teamplaysagainstteam	81.3	81.3	81.3	61.0	61.0	61.0	85.6	85.6	**85.6**
weaponmadeincountry	88.9	87.0	**87.9**	44.4	43.5	44.0	84.7	82.9	83.8
lakeinstate	90.3	92.4	91.4	84.7	86.6	85.6	91.5	93.6	**92.5**
Average all 11 predicates	83.43	83.39	83.4	73.7	73.5	73.6	86.5	86.3	**86.4**

Table 5: Performance scores of proposed methods and the baseline. Best F_1 values for each predicate is marked in bold [DS14].

`barrack_hussein_obama` has a candidate match. In addition, a similarity threshold can be introduced in order to avoid matching by arbitrary longer terms.

7.5 RESULTS: COMBINED METHOD

The F-measures reported in the previous section is towards the higher side. To get an actual impression of its effectiveness, we decided to pair it up with another graph based disambiguation method. This method has been mentioned in Section 5.3. In this section, we perform detailed evaluation and report the performance for each of the three methods in Table 5: the frequency-based, the graph-based, and the combined approach. As expected, we find that the most frequent entity baseline shows strong results. In contrast, the graph-based method shows an overall F_1-measure of only 73.6 compared to 83.9 for the baseline. Our combining approach, however, improves over the most frequent entity baseline by 2.9% w.r.t. average F_1, which is notably a difficult competitor for unsupervised and knowledge-rich methods. We must point out that in this experimental setup, we considered 11 out of the 12 NELL relations. In particular, we left out the relation *companyalsoknownas*. This was one relation with polysemous entities. The triples for this relation predicate had a wide usage of abbreviated terms (the stock exchange codes for the companies) and that resulted in poor numbers with the graph based method. Since the graph based scheme looks for a connecting path

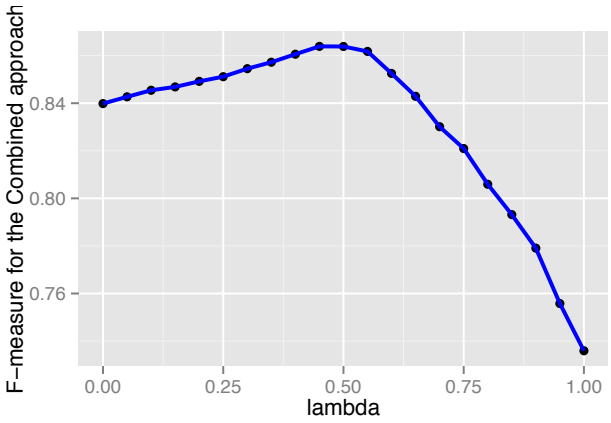

Figure 12: Effect of λ on the average F_1 score.

in the knowledge graph, it is often hard to find one such path between companies and its "nicknames" (IBM vs International Business Machines). Our baseline method was also poor in its performance. This can be easily detected in the precision, recall plots from Figure 9 and Figure 10.

Our observations state that the combined methodology of instance matching is effective in most of the reported cases but for the relations, *agentcollaborateswithagent* and *weaponmadeincountry*. In contrast, the graph-based approach showed a a varied performance. For instance in *actorstarredinmovie*, F_1 increases from 81.3 to 90.5, but for *weaponmadeincountry*, it decreases by 50%, the latter meaning that the graph-based method selects very often highly related, but incorrect subject-object pairs. The cases which had no such notable improvements with the combined method was mainly due to the low graph based score. For *weaponmadeincountry* the F_1 was around 44% and when combined with the frequency based ones, it reduced the scores. This deterioration can be attributed to the lack of sufficient relatedness evidence favoring the likelihood of the correct candidate pairs in DBpedia graph. For example for *actorstarredinmovie*(*morgan freeman*, *seven*), two possible candidate pairs (out of many others) with their probabilities are as follows (example from [DS14]):

$$(\texttt{db:Morgan\_Freeman}, \texttt{db:Seven\_Network}) \quad P_{freq} = 0.227; \quad P_{graph} = 0.074$$
$$(\texttt{db:Morgan\_Freeman}, \texttt{db:Seven\_(film)}) \quad P_{freq} = 0.172; \quad P_{graph} = 0.726$$

With the most frequent sense method, we would have selected the former pair, given its higher prior probability of $P_{freq} = 0.227$. However, the graph-based

method captures the relatedness, as DBpedia contains the directly connecting edge `dbo:starring` and thus rightly selects the later pair. In other cases, as observed often with *personleadsorganization* and *weaponmadeincountry*, a low prior probability was complemented with a semantic relatedness, thus a high P_{graph}, thereby making a highly related, but incorrect subject-object-combination candidate more likely than the correct one. Consequently, the graph-based approach by itself lowers the performances, relative to the baseline.

The fact that our combined method outperforms both the other approaches indicates that the linear combination of the two probabilities effectively yields in selecting the better of the two methods for each Nell triple. However, in addition to this effect, we observe that our combined approach also finds the correct mapping in cases where both, the frequency-based and the graph-based approach fail individually. Giving one example from the data, for the triple *teamplaysagainstteam(hornets, minnesota timberwolves)*[5], the frequency-based approach disambiguates it to the pair `(db:Hornet, db:Minnesota_Timberwolves)`, which is incorrect, as `db:Hornet` is an insect. But the graph-based approach also disambiguates wrongly to the pair (`db:Kalamazoo_College`, `db:Minnesota_Timberwolves`), even though it discovers a very specific path in DBpedia between subject and object in this pair, via the intermediate entity `db:David_Kahn_(sports_executive)`. The gold standard pair, (`db:New_Orleans_Pelicans`, `db:Minnesota_Timberwolves`), however, gets selected by the combined approach, which combines the medium high prior probability and a medium high relatedness originating from the fact that both instances are connected by `yago:YagoLegalActor`. Last, we report on the robustness of our combined approach with respect to the parameter λ, even though giving equal weight to both methods, thus setting λ to 0.5, seems to be a natural choice. Figure 12 shows the F_1-measure for $\lambda \in [0; 1]$. Note that $P_{joint} = P_{graph}$, when $\lambda = 1$ and $P_{joint} = P_{freq}$, when $\lambda = 0$. We observe a clear peak at $\lambda = 0.5$, which confirms our initial choice.

7.6 RESULTS: PROBABILISTIC METHOD

The probabilistic method introduced in Chapter 6, used a parameter α for generating the instance types. The effectiveness of our method relies on the choice of a proper value for the parameter α. In a first set of experiments, we analyze ways

5 "hornets" refers to `db:New_Orleans_Pelicans`, formerly the New Orleans Hornets.

to search for an appropriate value of α. In the later half of this sections, we use the empirically obtained value of α for reporting on the results of our algorithm related to the final outcome. Experiments that focus on the impact of the different iterations are presented in Section 7.6.2.

7.6.1 *Parameter : Search and Effect*

Search

Here we report about performing a 2-fold cross validation on different samples of the whole dataset in an repeated fashion. For that purpose we restrict the possible values of α to be in multiples of $1/8$ in the interval $[0, 1]$, which allows us to repeat the overall process over a large number of sampling steps ($\approx$ 100000). At each sampling step, we first randomly pick half of the properties. This choice defines two datasets consisting of 6 properties (the chosen properties and the residual properties). We call one of them the training set D_{train} and the other the testing set, D_{test}. For every D_{train} we find the α giving the maximum averaged F_1 score over D_{train}. Then we apply our algorithm with that α on D_{test} and compute the resulting F_1. For 35% of the samples we learnt α=0.5, for 30% cases we learnt α=0.375, and for 18% we learnt α=0.625. Approximately 85% of all samples yield an α in the interval $[0.375, 0.625]$, signifying that learning produces a stable outcome. Applying the learned α on D_{test} results in an increased average F_1 score of 85.74% (+3.94%) compared to the baseline with an average F_1 score of 81.8%. Thus, it is possible to learn α based on a small training set (600 triples) that results in a significant improvement of the mapping quality.

Effect

Finally, we compute recall, precision and F_1 values on the entire dataset for all values of α in the $[0, 1]$ range with step sizes of 0.125. This helps us to better understand the impact of α on precision and recall. In Table 6 we report the absolute scores along with the differences (Δ) in scores over the most frequent baseline of [DNMP13] ($\mathcal{M}_0$ in Algorithm 1), and the output of the final iteration of the bootstrapped approach. Our results corroborate the findings from our cross-validation runs in that we achieve the best performance on the full dataset for $\alpha = 0.5$, which yields an improvement of +3.94% in terms of F_1 with respect to the baseline. Low values of α increase the precision by up to +12.3% ($\alpha = 0.0$), thus resulting in an overall precision of 95.1%, with a loss of 5.2% of recall as

α	prec (Δ_{prec})	rec (Δ_{rec})	F_1 (Δ_{F_1})
0.0	95.1 (+12.30)	76.1 (-5.20)	84.1 (+2.26)
0.125	94.8 (+12.04)	77.6 (-3.70)	84.9 (+3.08)
0.25	94.7 (+11.96)	78.5 (-2.80)	85.4 (+3.66)
0.375	94.4 (+11.58)	79.0 (-2.26)	85.6 (+3.84)
0.5	93.1 (**+10.34**)	79.9 (**-1.39**)	85.7 (**+3.94**)
0.625	92.3 (+9.48)	80.2 (-1.08)	85.6 (+3.76)
0.75	91.4 (+8.63)	80.4 (-0.96)	85.3 (+3.46)
0.875	90.3 (+7.53)	80.6 (-0.67)	85.0 (+3.15)
1.0	87.6 (+4.80)	81.0 (-0.35)	84.0 (+2.17)
Baseline	82.78	81.31	81.8

Table 6: Effect of α on the overall performance compared to the baseline. The last row reports the Baseline score, and every cell shows the change (+/-) in points with respect to the baseline. Hence for α = 0.5, the F_1 increased by 3.94 compared to 81.80 [DMP14]

trade-off. While low values of α increase precision by aggressively eliminating many incorrect mappings, increasingly higher values lead to a drop in precision, indicating an ever increasing number of incorrect mappings being produced. The purpose of this experimental setup was not just find an optimal α score for our subsequent experiments, but also to make the general claim that the one can choose different α value as desired. Table 6 illustrates nicely that we can use α to adapt our approach to the needs of a certain application scenario, where precision or recall might be more important.

7.6.2 *Algorithm Performance*

In Table 7 we report the performance figures of our approach for each of the properties in the evaluation dataset. As baseline and initial starting point $\mathcal{M}_0$ we use the most frequent mapping presented in [DNMP13]. The following columns ($\mathcal{M}_i$, $i \neq 0$) report the F_1 scores of our proposed approach. The suffix i denotes the iteration number and $\mathcal{M}_i$ allows us to easily identify the mapping set for each bootstrapped iteration. For all experiments we set α=0.5. This choice is supported by the results of the previously-described cross-validation approach as well as by the theoretical considerations presented above. The results highlight that with an bootstrapped approach we are able to beat the baseline, and improve our results in average across iterations.

Nell Relations	$\mathcal{M}_0$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$
actorstarredinmovie	81.3	87.7	94.5	-
agentcollaborateswithagent	83.7	76.4	82.0	-
animalistypeofanimal	85.9	86.0	86.7	-
athleteledsportsteam	87.0	92.6	93.2	-
bankbankincountry	79.6	86.4	83.4	82.8
citylocatedinstate	80.7	83.2	83.2	83.0
bookwriter	82.6	86.7	86.7	89.0
companyalsoknownas	64.1	64.6	67.1	-
personleadsorganization	76.7	78.1	77.2	77.6
teamplaysagainstteam	81.3	89.6	90.9	-
weaponmadeincountry	87.0	87.0	-	-
lakeinstate	91.4	94.4	94.7	-
Cumulative Gain (%)	-	2.62	3.89	3.94

Table 7: F_1 scores of the baseline ($\mathcal{M}_0$) and our bootstrapping approach, α=0.5

In our experiments, we found no improvements beyond the third iterations $\mathcal{M}_3$, thus indicating that our method quickly converges after few iterations. Performance figures indicate the gradual saturation in the scores after each iteration. As expected, with each iteration the output gets more refined until a plateau in the results is reached, and no further improvement is gained with our method. In some cases our approach does not modify the original baseline (e.g. `weaponmadeincountry`). This is mostly attributed to missing type information in DBpedia. Note that results get slightly worse for some properties in some of the iterations. In the following section we analyze the behavior of our algorithm in details by looking at some concrete examples. These examples help to understand why some cases deviate from the general positive trend.

We wanted to employ the exact strategy of our probabilistic method on the Reverb data set. Our primary goal here is to observe the effectiveness of the model on a completely different data set. This was interesting for us, since, our initial analysis with Reverb data set had revealed some wide differences with the Nell data set. We do not undertake the parameter search steps but directly apply the method by setting α value to 0.5 (as empirically observed in Section 7.6.1). Similar to Nell we sample a set of 12 properties and let our instance matching algorithm determine the correct matches. In Table 8 we report on the F_1 scores of the baseline method and the final refined step of our algorithm. The scores marked in bold are the better scores. As we observe, for some of the relations, we could not beat the baseline. However, on an average, we achieved a higher

Reverb Relations	Baseline ($\mathcal{M}_0$)	Probabilistic IM
is in	71.53	71.42
is located in	85.60	**85.71**
was born in	57.83	**74.19**
is a registered trademark of	93.54	90.00
located in	85.18	**95.83**
is a suburb of	89.28	**96.15**
is part of	86.36	**90.00**
originated in	100.00	100.00
is the capital of	93.47	**94.88**
stands for	56.25	50.00
is a city in	100.00	93.75
is the home of	91.79	**93.33**
Macro-average F_1	84.24	86.27

Table 8: F_1 scores of the REVERB baseline ($\mathcal{M}_0$) and the final bootstrapping approach (at α=0.5)

Candidate	Type	BL	1^{st} Iteration	2^{nd} Iteration
Scarface_(rapper)	MusicalArtist	0.06	$0.06 - 3.04 = -2.98$	$0.06 - 13.81 = -13.75$
Scarface_(1983)	Film	−0.58	$-0.58 + 0.36 = -0.22$	$-0.58 + 3.37 = 2.79$
Scarface_(1932)	Film	−2.22	$-2.22 + 0.36 = -1.86$	$-2.22 + 3.37 = 1.15$
None	[-]	0.0	$0.0 + 0.0 = 0.0$	$0.0 + 0.0 = 0.0$
Salesman	Play	1.59	$1.59 + 0.08 = 1.67$	$1.59 + 0.83 = 2.42$
Salesman_(1951)	Film	−2.50	$-2.50 - 0.05 = -2.55$	$-2.50 + 0.57 = -1.93$

Table 9: Weight refinements across iterations for the object of the triple actorstarredinmovie(*al pacino, scarface*) and for the subject of the triple bookwriter(*death of a salesman, arthur miller*). Grey cells refer to the mappings generated at each iteration. [DMP14]

F_1 score of 86.27% than that of the baseline with 84.24%. This is an rise by 2.03 point in F_1, compared to the baseline. Recollect from the scores reported with NELL, we could have an increase in 3.94% points in F_1. This should not be directly compared, since these are completely different data sets. But, we should note the general applicability of our method. Irrespective of the underlying structure of the input data, our method is robust against variations and is capable of improving the most frequent sense baseline which itself is strong and robust.

7.6.3 *Empirical Analysis*

First, we focus on the object of the NELL triple `actorstarredinmovie`(*al pacino, scarface*). The object term *scarface* has three possible mapping candidates, which are shown in the first column of Table 9 together with the case in which no candidate is chosen (identified by `None` in the table). In the table, we identify the candidate chosen in each iteration with a grey cell. `None` is chosen if the sum of all weights is less than 0, which means that all mapping candidates have a probability of less than 50%. For the column baseline, the only relevant weight corresponds to the most frequently-linked mapping candidate. No type-related weights are available, and accordingly the wrong entity *Scarface_(rapper)* is chosen. In the first iteration, the weights for the types are added to those of the mapping hypothesis. These type weights are obtained by applying the α-tree computation to the baseline. With respect to our example, this results in rejecting all of the candidates, because each has a probability of less than 50%. Specifically, we observe that the weight attached to the range type `Film` is not yet high enough to increase the overall score for the two movies up to a score greater than 0. The second iteration uses the type weights computed on the refined α-tree, which is created on the basis of the outcome from the previous iteration. The weights attached to `Film` have now increased significantly, and consequently one of the two movies is chosen. In this case, the algorithm chooses the right candidate. However, this example also reveals the limits of our approach, namely the fact that our method solely relies on type-level information. For this reason, the final choice between the movie from 1983 and the movie from 1932 is only based on the fact that the movie from 1983 has a higher mapping weight, namely it is more often referred to by the surface form *Scarface*. That is, all things being equal (i.e., given the same type-level information), our approach will still choose the most popular entity, which might be a wrong choice. The second example is the mapping of the subject term in `bookwriter` (*death of a salesman, arthur miller*). In this example, the candidate chosen by the baseline is also that chosen in each subsequent iterations. Contrary to the first example, the type of the chosen candidate is not the highest scoring type according to the α-tree, namely `Book`. While for the second iteration the weight for `Book` is $+1.62$, the weight for `Play` is $+0.83$, based on the fact that `Play` is a sibling of `Book`. Thus, its weight is not only supported by all matched plays, but also indirectly by all matched books. This example illustrates the benefits of the α-tree and the differences of our approach compared to an approach that simply

Nell Relations	Frequency Based	Combined	Probabilistic
actorstarredinmovie	81.3	92.1	**94.5**
agentcollaborateswithagent	83.7	83.7	82.0
animalistypeofanimal	85.9	86.3	**86.7**
athleteledsportsteam	87.0	90.1	**93.2**
bankbankincountry	79.6	79.6	**82.8**
citylocatedinstate	80.7	**86.2**	83.0
bookwriter	82.6	88.0	**89.0**
personleadsorganization	76.7	**82.4**	77.6
teamplaysagainstteam	81.3	85.6	**90.9**
weaponmadeincountry	87.0	83.8	**87.0**
lakeinstate	91.4	92.5	**94.7**
macro-average (%)	83.38	86.39	**87.4**

Table 10: Comparison of F_1 scores of the baseline method, combined approach and probabilistic approach with bootstrapping

uses the majority type as a hard restriction.

7.7 RESULTS: METHODS COMPARISON

In this short section we do performance analysis of our improved probabilistic instance matching scheme. In the process we mentioned four methods: (i) Wikipedia most frequent sense based (ii) A graph based method (iii) a combination method incorporating best of both the worlds from (i) and (ii) and finally (iv) a probabilistic method with bootstrapping. The most advanced method turns out to be the probabilistic one, but we cannot make this absolute claim until we compare it with the combined method. The combined approach showed promising results which has been proven in Section 7.5. Hence, we decide to consolidate all the F_1 scores from these methods and give one comparative snapshot of the instance matching schemes. This numbers are presented in Table 10. We present 11 NELL relations in the table here, since we omitted *companyalsoknownas* from evaluation due to extremely poor numbers with the graph based scheme. This causes the F_1 scores to go up a bit. These numbers have been already presented earlier in different contexts but not in mutual comparison with one another. The table clearly show the robustness of the probabilistic method over the other two.

CONCLUSION

The IM module detailed in this chapter explores and compares two distinct methods: a baseline method of using the most frequent sense from Wikipedia and an advanced probabilistic method formulated as an inference task in markov logic networks. The results for this approach is particularly interesting since, we show the strength of the baseline method and use it as a baseline method. Our probabilistic approach is robust and can beat the baseline achieving around 85% F_1 score for NELL and around 86% F_1 for REVERB. It is a very descent performance given that we do not use any context for performing the matching task. The key to our approach, lies in an iterative bootstrapping, which performs repeated refinement and elimination steps to find the final set of correct mappings. Our implementation and design is general and can work with extracts from any sources.

However, we must report on the drawbacks of this module. First, it cannot differentiate between two similar instances which are typed as the same concept in the target KB. For instance, for the entity reference for an OIE term "*Hillary*" in some OIE context, it might be difficult with this approach to differentiate between two or more contesting candidates which are both typed as `dbo:Persons` in the DBPEDIA ontology. In such scenario, exploiting contexts could be an easy solution. Second, the instance matching is guided by the types of the instances involved in a relation, hence in cases where the type is not available (this is sometimes possible as all the instances are not necessarily typed with some concepts in DBPEDIA), it might be hard to find one. We have used YAGO as an additional knowledge source to augment our type deduction for such instances. This is possible due to the several datasets available inter linking the two IE systems[6].

6 http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/

Part III

RELATION MATCHING

8

INTRODUCTION

This part of the thesis particularly focuses on the task of finding a mapping for the OIE relations to a KB relation. This chapter introduces the problem formally and mentions some of the closely related research areas. This can be considered a complementary research area to that of entity linking since finding the most likely mapping for an open domain relation enhances the possibilities for better instance matching as well. In the context of our work, we perform instance matching as the first task and subsequently followed by relation matching. However, one can adopt the inverse sequence. It would still work, since both the mappings for instances and relations positively influence each other. Furthermore, we must also mention that in this work we are mainly concerned with binary relations, i.e. defined defined two arguments, for instance, *fatherOf(A,B)*, *locatedInC,D* and so on. In this regard, we first present the closely related works on mapping relations in Section 8.1.1. In Section 8.1.3 we discuss about relation extraction. Since we employ cluster-based methods for relations for performing the mapping, we also mention some works in this area in Section 8.1.2.

8.1 RELATED WORK

8.1.1 *Relation Mapping*

The relation mapping task concerns with finding the semantic representations of relational phrases connecting two entities in a given unstructured source. Such phrases are usually in natural language texts and the mapping task attempts to find a KB representation of them. The relation can be a word like *married*, *birthPlace* or can be complex phrases like *has been the doctoral advisor of*. We found a close resemblance with the work of Liu et al., [LLZ+13]. Their proposed solution was based majorly on Wikipedia text snippets and mapping the relations extracted from them to Wikipedia infobox properties. This is radically not a different setting than ours, since we are mapping NELL and REVERB relations to the DBPEDIA relation set. We chose this particular work for evaluation of our pro-

posed relation mapping approach on REVERB. In Section 11.1.4 we detail on the experimental setup and present the detailed numbers. In particular, they considered two classes of triples: one is termed as open domain triples, which is a collection of relation instances extracted from Wikipedia articles (can be considered to be of the same form as that of REVERB extracts). And the other termed as attribute triples, which is a collection of triples from the Wikipedia infobox properties. They concluded that the open relation and the infobox attribute are equivalent if the corresponding subjects and objects in the two classes of triples are same. In comparison, we do not make such equivalence assumption on the instance levels, but use the probable domain and range restrictions for aligning the relations. However, there are few drawbacks with the proposed method of Liu et al. First, it will not work for a open relation triple which is not from Wikipedia articles. Second, it does not consider scenarios where subsumption relationships may exist between the triple instances. Our method is more general on these aspects and overcomes these issues. We do not match on instances but on the domain and range which allows us to incorporate hierarchical relationships into our model.

The work of Soderland et al., [SRQ+10], which aims at mapping the relations from REVERB to "domain relations". The authors adopted a two step approach: first, by finding a suitable class recognizer for the OIE terms, second, learning a mapping of the relation under the constraints of the classes recognized. However, they used NFL (National Football League) as the target KB, unlike broader DBPEDIA in our case. Soderland et al., [SM07] also worked on "ontologising" REVERB relations. This work has some resemblance with our work. The authors have tried to define entailment relationships within REVERB relations using Wordnet [Mil95] sync-sets. This approach allows deductive reasoning on REVERB by using the hierarchical structure of the relations not encountered in the corpus. Note that the work was not about mapping to some target knowledge base but defining a hierarchy within REVERB relations.

8.1.2 *Relation Clustering*

Clustering in general refers to the task of dividing a set of items into multiple disjoint partitions such that each partition consists of closely related items and items across different clusters are sparsely related. The task of relation clustering is about forming logical partitions within a group of relations such that relations

within a cluster are semantically synonymous. In our case, we use the OIE relations and also the KB relations to find such logical groups. The idea of relational clustering has been in use across various research problems: unsupervised approach to extracting semantic networks from large volumes of text [KD08] and in semantic parsing as mode for mapping texts to formal representations [PD09]. More recently, Zhang et al., [] employed the idea of clustering surface forms of relations across news streams to paraphrase them using temporal features. Some of the other related works in this area was done by Yates et al., [YE09]. They introduced the system called RESOLVER which is a synonym discovery tool for both the instances and relations. The clustering algorithm they used was that of greedy agglomerative clustering with spatial feature representations. They used generative probabilistic model as their key ingredient which yielded a 90% precision and 35% recall. These values represent the efficiency of their clustering algorithm in keeping synonymous relations together. However, we required an approach which would work across cross domains. The experimental setup as explained in the paper uses web crawls and relations from such extracts, however, it is bit ambiguous how effective their modeling would be for finding synonyms between OIE relations and KB relations. Furthermore, there is a broad range of work which performs clustering of synonymous relation to reason about the KB [HSG04, HSG04, AVH+12, dLL13]. We have similarity with this strain of research where we perform clustering to leverage better statistics for the instance matching task.

8.1.3 *Relation Extraction*

With the recent advances in open extraction methods, relation extraction has achieved a prominent status. However, relation extraction task aims at finding the binary relations in an unstructured information source. This can be considered as a precursor to the mapping task. The relations in the triples from REVERB and NELL data sets are the output of such relation extractors. Our work does not involve any extraction task, but we present some of the major works in this area since REVERB and NELL employ this in order to discover relational phrases across the web. This strain of research is wide and consists of a huge body of work. They range from supervised approaches (feature based [Kam04], kernel based methods [ZG05], [BM05]) to unsupervised and bootstrapping based approaches (DIPRE [Bri98], Snowball [AG00], definitely KNOWITALL [ECD+05],

TextRunner [BCS+07]). There have been other evolved techniques employing tensor factorization models, some of the prominent ones are TripleRank [FSSS09], RESCAL [NTK11], PITF [DRST12]. These techniques in general are good in predicting new relations between entity pairs. This sets tensor based methods apart from the matrix factorizations techniques [JTHN12, HTN+14] which cannot predict new relations. Usually relation extraction research involves exploiting contextual information [ZHW12, dLL13, TSN11]. Content in the form of parts-of-speech, concept associations of entity, also bag of words have been used for the purpose.

Broadly, there is TargetIE or target information extraction and OpenIE pattern for relation extraction. The former starts with a set of entity pairs for a given relation (preexisting in a KB) or simply seed relation instances and learn relation extractors based on these from text. These learnt extractors are used to discover more relation instances and eventually these are bootstrapped to learn better extractors [STNM12, MBSJ09]. In contrast, the later class extracts all relations from a text irrespective of their existence in a KB [BCS+07]. According to Riedel et al., [RYMM13] open extractors outperform the targeted ones.

8.2 PROBLEM STATEMENT

In this section, we formally introduce the problem of relation matching in the context of our whole workflow. As stated by Rong et al., [RNX+12], the task of relation matching has been simply described as the following: "*Property matching links the properties from different data sources which have similar semantics.*". In the rest of this work, we often use relations and properties interchangeably and they necessarily do not mean different in our context. As also correctly pointed out by Rong et al., the task is not simple since, the different data sources often have different ontologies. In our case, the source input (OIE end) usually lacks the ontologies. This makes the problem setting for us even more difficult. In particular, the OIE data sets are open domain extractions with no guarantee of a clean ontology. Moreover, this is the setup for our problem, a schema independent way of mapping.

Let, $\mathcal{T}^R$ be the set of relations from the target data source and $\mathcal{S}^R$ be the set of relations from the source data. In our case, the source is essentially the OIE inputs

and target is DBpedia. If r_s be an element of $\mathcal{S}^R$ and r_t be an element of $\mathcal{T}^R$, then our aim is to create a set of relation mappings, defined as $\mathcal{M}^R$, and given by,

$$\mathcal{M}^R = \{ (r_s, r_t) \mid (r_s, r_t) \in \mathcal{S}^R \times \mathcal{T}^R \} \tag{10}$$

Observe that, the representation of the relation matching set has strong correspondences with the instance matching set $\mathcal{M}$ (as defined in Expression 1). But, we do not have a cardinality constraints in $\mathcal{M}^R$. This means, this set is a many-to-many mapping pattern from source to target data sets. This is a realistic formalism of the actual world, since a multitude of OIE relations can map to the same KB property and also a number of KB relations can be a mapping for a single OIE relation. For instance, Reverb relations *hold three degree from* and *study at* map to the DBpedia relation `dbo:almaMater`. And likewise, a single Nell relation *haswife* can be mapped to both `dbo:spouse` and `dbo:partner`. Furthermore, the definition ensures that the mapping is always between an OIE and a KB relation and not between relations within the same data source.

In the subsequent chapters, we present different ways of generating the mapping set. A major difference with the IM task lies in the range of correct mappings a given OIE relation can have. In contrast, IM task has at most one correct mapping. In designing our solution, we incorporated these aspects and proposed two different approaches for relation mapping: rule based approach and a cluster based approach. The Expression 10 above presents a generalized form of the relation mapping set. Before presenting the details of either of the methods, we briefly introduce the notion of clusters. A cluster is a set of synonymous items and in this case set of relations in particular. We do not explicitly mention here if it is a set of OIE relations or KB relations, since it can be any. Given a collection of relations $\mathcal{R}$, a cluster set $\mathcal{C}$ is defined as a collection of pairwise disjoint clusters c_i such that,

$$\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\} \tag{11}$$

$$\textit{where,} \quad \forall c_i, c_j \in \mathcal{C} \text{ and } i \neq j, \quad c_i \cap c_j = \emptyset$$

The cluster definition above ensures that the individual clusters c_i, c_j are strict subsets of the collection $\mathcal{R}$ and there are no common elements across different clusters. Hence, the complete set $\mathcal{C}$ is an union of all the individual clusters. We must note that, often a particular relation may not be synonymous with the other relations and hence constitutes an one element cluster. The cardinality of the set

$\mathcal{C}$ denotes the size of the cluster set.

The Expressions 10 and 11 above provide a general formalism of the relation mapping task along with the cluster definition. We present in the following section how the formulation holds under the different workflow modes. We use the same set of notations across the different scenarios and show the general nature of the formalism.

Rule Based (single relations):
This is a scenario where the source relations from OIE are not clustered. The individual relations from the input data source thus can be considered an one element cluster. Hence, we have the following,

$$\begin{aligned} \mathcal{R} &= \mathcal{S}^R \\ |c_i| &= 1, \quad \forall c_i \in \mathcal{C} \\ \mathcal{M}^R &= \{ (r_s, r_t) \mid (r_s, r_t) \in \mathcal{S}^R \times \mathcal{T}^R, \quad r_s = c_i, \quad \forall c_i \in \mathcal{C} \} \end{aligned}$$

This is the default case of clustering where every single relation forms a one element cluster. Note that every element r_s from the input set of relations is essentially the cluster c_i which has exactly one element. And this cluster is mapped to KB relations.

Cluster Based:
This is another alternative of clustering, where we cluster the OIE relations and the KB relations together. This has no rule based components. The eventual clusters formed are a natural grouping where OIE relations are synonymous with KB relations and hence can be well considered as a mapping. In particular, we have the following setting in this case,

$$\begin{aligned} \mathcal{R} &= \mathcal{S}^R \cup \mathcal{T}^R \\ |c_i| &\geqslant 1, \quad \forall c_i \in \mathcal{C} \\ \mathcal{M}^R &= \{ (r_s, r_t) \mid (r_s, r_t) \in \mathcal{S}^R \times \mathcal{T}^R, \quad r_s \in c_i, \quad r_t \in c_i, \quad \forall c_i \in \mathcal{C} \} \end{aligned}$$

It is important to note the values assumed by the set elements r_s and r_t, they essentially belong to the same cluster c_i. This exploits the general idea that if multiple relations are in the same cluster then they are synonymous. Hence, we

do not need any rule based mapping here since the individual clusters themselves group OIE relations and KB relations. It must be noted that it cannot be assured that every cluster should contain a KB relation. Hence, we can have scenarios where a cluster has no KB relation in it or vice versa, i.e. a cluster with only KB relations. In either of these cases a mapping is not possible and hence a pair (r_s, r_t) is also not generated. This follows directly from the $\mathcal{M}^R$ definition for this case. If there are no KB relation, the condition $(r_s, r_t) \in \mathcal{S}^R \times \mathcal{T}^R$ fails and hence a mapping is not generated.

Rule Based (clustered relations):
This is an aggregate method which employs the best of both the worlds. We first cluster the OIE relations only (without any KB relations) and use the clusters to be mapped to KB relations using rules. Thus this is a hybrid approach which has the following settings,

$$\begin{aligned} \mathcal{R} &= \mathcal{S}^R \\ |c_i| \geqslant 1, \quad \forall c_i &\in \mathcal{C} \\ \mathcal{M}^R = \left\{ (r_s, r_t) \mid (r_s, r_t) \in \mathcal{S}^R \times \mathcal{T}^R, \quad r_s \in c_i, \quad \forall c_i \in \mathcal{C} \right\} & \end{aligned}$$

The difference in this approach is subtle, since the range of values r_t takes is from the set $\mathcal{T}^R$ instead of the cluster as in the former case, while r_s essentially is a cluster element.

All these above mentioned formalisms were defined with minor variations depending on the workflow but still maintaining a strict parity with the general definitions of relation mapping set $\mathcal{M}^R$ (Expression 10) and the cluster definition $\mathcal{C}$ (Expression 11). In the rest of this part, we provide details of the methodologies for each of the rule based (Chapter 9) and cluster based (Chapter 10) methods. In Chapter 11 we provide empirical results, analysis and comparative studies with other related works for relation matching task.

9

RULE BASED APPROACH

In this chapter we detail our methodology for using association rule mining techniques as a method for performing the relation mapping task. In this particular chapter, we present an approach which exploits the domain and range restrictions learnt as evidences and applies them to map OIE relations to a KB. The underlying idea we employ is that of distant supervision. In terms of Augenstein et al., [AMC14], "*If two entities participate in a relation, any sentence that contains those two entities might express that relation*". Hence, each OIE relation instance, if encountered in some relation instance in a KB, then it is a positive indicator for the equivalence of the two relations in the two knowledge source. In the following sections, we present in details our methodology based of this idea. Some of the concepts and experimental results presented in this chapter have been used from our former publication [DMS14].

9.1 METHODOLOGY

This section presents our approach for mapping an OIE property to an analogous DBpedia property. In this work, we start from mapping ambiguous entities to the KB instances first and then try to use the refined mappings to find the correct relation mapping. This is indeed not a strict order but one can also perform matching the other way round. Hence, while presenting the details of this section, we already have precise and disambiguated OIE terms expressed as KB instances. For every OIE triple of the form p(s, o), we map the subject (s) and object (o) individually to DBpedia instances d_s and d_o respectively using the instance mapping technique as presented in Chapter 6. Using a DBpedia SPARQL endpoint, we query[1] for an assertion in the KB. In particular, a triple involving some property d_p and having the form of the form $d_p(d_s, d_o)$. The key idea is, if such a triple exists, then d_p can be considered as a likely DBpedia mapping of p. The SPARQL query can return multiple results, a single result or no result at all. In the later sections, we categorically deal with each of these cases and present ways to

1 SPARQL: select ?d_p where {?d_s ?d_p ?d_o. ?d_p a owl:ObjectProperty}

tackle them. This general technique is applied over all OIE relation instances occurring in the dataset. Furthermore, we also consider inverse property mappings. We denote d_p^{inv} to be an inverse mapping for p, if the triple d_p^{inv} (d_o, d_s) exists in DBpedia. The methodology proposed in this work is applicable for both the two cases and we use d_p as a general notation for DBpedia property. We must also note that we restrict our SPRQL endpoint query results only to the set of DBpedia object relations (`owl:ObjectProperty`)[2], i.e. the binary relations in DBpedia having entities as both the arguments (e.g. `dbo:hometown(db:Eminem, db:Detroit)`) and not functional properties which have one of the arguments as a literal values (e.g. `dbo:birthDate(db:Eminem, "1972-10-17")`). We do not consider the later class of `owl:DatatypeProperty`[3] since we are interested in entities in general.

9.1.1 *Likelihood Estimate*

The idea proposed in the previous section is promising but has a minor complication. The SPARQL query may return multiple values for d_p or just a single value. Hence, to decide on the mapping of $p \rightarrow d_p$ we require to decide on the correctness of a particular mapping. A simple scheme to quantify the likelihood of a KB relation to be a matching OIE relation candidate could be a frequency count. In particular, we want to estimate the likelihood of every possible mapping of p to one or more d_p. As an initial step, a naive frequency count of the mappings can give us a likelihood estimate. For instance, if the Nell property *bookwriter* is mapped to `dbo:author` in k out of n cases and to `dbo:writer` in (n-k) out of n cases, then the likelihood scores for these two cases can been given as follows:

$$\text{likelihood}(\text{bookwriter} \rightarrow \text{dbo}:\text{author}) = \frac{k}{n}$$
$$\text{likelihood}(\text{bookwriter} \rightarrow \text{dbo}:\text{writer}) = \frac{n-k}{n}$$

These likelihood scores can have a threshold which denotes the acceptance limit of the scores. Finally, selecting candidates above the threshold, could give us a simple solution. However, this approach suffers from two major drawbacks: first, any conceptually similar property (as in this case) might be eliminated out due to lack of sufficient evidence (low likelihood score). For instance, (n-k) can be very small in the above expression just due to lack of evidences, but semantically the

2 http://www.w3.org/TR/owl-ref/#ObjectProperty-def
3 http://www.w3.org/TR/2004/REC-owl-semantics-20040210/#owl_DatatypeProperty

mapping is not wrong in . Second, finding a correct threshold.

We propose an alternative to finding the mapping. This is an improved approach and incorporates the type information of the mapped DBpedia instances. For each OIE triple in the input, let the subject s be mapped to d_s and object o mapped to d_o. Using the publicly available DBpedia endpoint, we collect the type of these mapped DBpedia instances. For easy reference, we denoted them as $dom(p)$ and $ran(p)$ respectively. This notation should not be read as "domain of p", and likewise "range of p". The concept types of the mapped subject (and object) are being referred to as the likely domain (and range) of the OIE relation p. It must be observed that querying for an analogous KB triple (of the form d_p (d_s, d_o)) might not often result in some triples, and also can have multiple possibilities as shown in the Example 3. Hence, we identify and differentiate these three cases as:

I a single possible value is returned for d_p (Case I in Example 3)

II multiple values for d_p are returned (Case II in Example 3)

III an empty set is returned, indicating absence of any d_p. This can happen if there is no such triple in DBpedia or the mapped instances are wrong at the first place.

We observe that, depending on the returned value of the result set, we can have multiple interpretations. All these variations can be represented under a single unified form using tuples. Case (2) and Case (3) are given an unified representation by framing discrete association tuples as,

$$p, d_p, dom(p), ran(p) \tag{12}$$

In Example 3 we present the different possible cases and also their representations as association tuples. Observe that now, we do not consider the occurrence of d_p alone but also consider the domain and range associations instead. We intend to achieve two major goals with this variation: (1) finer granular vision of the relation matches (2) exploit the KB ontology to our advantage to ease the relation mapping task.

Example 3.

*Case I: **Single** value for airportincity(helsinki_ vantaa_airport, helsinki)*

dbo:city(db:Helsinki_Airport, db:Helsinki)
After transformation into an association : airportincity, city, Airport, Place

Case II: ***Multiple*** *values for airportincity(vnukovo, moscow)*
dbo:city(db:Vnukovo_International_Airport, db:Moscow) and
dbo:location(db:Vnukovo_International_Airport, db:Moscow)
After transformation into associations :
airportincity, dbo:city, dbo:Airport, dbo:Place
airportincity, dbo:location, dbo:Airport, dbo:Place

Case III: ***No*** *values. There is nothing to transform in this case.*

The positive impact of such a translation is that, both the cases (I) and (II) mentioned above can have one representation. Intuitively, for Case (II) in particular, if multiple properties are possible then each one of them is equally likely to hold true. All the associations for p thus formed is denoted as A_p. As a subsequent step, we apply an association rule [LC01] of the form

$$p \Rightarrow dom(p) \wedge ran(p) \tag{13}$$

on A_p. This means, if the OIE relation is p then the type of the mapped DBpedia subject instance is $dom(p)$ and type of the object instance is $ran(p)$. Intuitively, the Expression 13 makes it evident that a likelihood of a particular OIE relation is manipulated by the types of the mapped subject and object instances. Hence, this reinstates our claim once more that, better the quality of the instance matches, better would be the relation matching.

Once expressed as an association rule, we can compute the confidence for each of these rules. We denote the confidence as $conf$ for each such rule. It denotes the frequency of co-occurrence of $dom(p)$ and $ran(p)$, whenever p occurred. For some rule i and relation p, the confidence is denoted as $conf_p^i$, and defined as,

$$\begin{aligned} conf_p^i &= conf(p \Rightarrow (dom(p) \wedge ran(p))) \\ \implies conf_p^i &= \frac{count(p \wedge dom(p) \wedge ran(p))}{count(p)} \end{aligned}$$

$$\implies \mathrm{conf}_p^i = \frac{\mathrm{count}(p \wedge \mathrm{dom}(p) \wedge \mathrm{ran}(p))}{|A_p|} \tag{14}$$

The definition of confidence is used from [LC01]. Referring to Table 11,

$$\mathrm{conf}^3_{\mathrm{agentcreated}} = \frac{\mathrm{count}(\mathrm{agentcreated}, \mathrm{Person}, \mathrm{Book})}{|A_{\mathrm{agentcreated}}|}$$

The table does not explicitly show the $\mathtt{conf}$ column, but the associations presented are used for the confidence score. The suffix 3 denotes the i^{th} association for the particular relation i.e. *agentcreated* in this case. We must note that the computation is performed on the association set A_p, hence, the $\mathrm{count}(p)$ will be the size of A_p. Intuitively, the task of finding the confidence of a particular rule reduces to finding the count of joint occurrence of the relation and its associated domain and range over the whole set of associations. Note the $\mathtt{count}$ function is not just the frequency count of the joint occurrence of a particular p and its associated DBpedia domain and range values, but also the sub-classes of each of the domain and range. The rational is, if we observe an association like *agentcreated* $\Rightarrow$ (`Person`, `Book`) then any other association like *agentcreated* $\Rightarrow$ (`Scientist`, `Book`) should also be considered as an evidence for the former association. `Scientist` being a sub-class of `Person` in the DBpedia ontology, is not a different association but a more particular case. Needless to say, count of the later is not affected by the count of the former. This technique is an improvement over the previously mentioned naive count based which did not exploit this inherent hierarchical structure of the structured KB. Finally, each association, is awarded with a confidence of $\mathtt{conf}_p^i$.

Our initial analysis with the data sets revealed that often OIE relation instances had varying degrees of KB counterparts. Some had quiet a lot of KB assertions while some extremely few. This motivated us to quantify the notion of *mappability* of a particular OIE relation p. It determines the degree to which a particular OIE property can be mapped. It is denoted by K_p and defined as:

$$K_p = \frac{\sum_{j=1}^{|T_p|} C(j)}{|T_p|} \tag{15}$$

$$\text{where } C(j) = \begin{cases} 1; & \textit{atleast one property mapping for } p \textit{ in } T_p^j \\ 0; & \textit{otherwise} \end{cases}$$

We introduced T_p, which is simply the set of all OIE relation instances for the given relation p. Note that, $C(j)$ in the formula above is counted as one even if p has single possible mapping (Case (I)) or multiple possibilities (Case (II)). Example 4 presents a simple scenario to show the computation of K_p. However, the actual value is 0.55 as shown in Table 11. Also note the use of T_p in the expression of K_p above. We do not require A_p here, since this factor is for determining the mapping degree. We are not looking into domain/range associations here. It can be easily and accurately determined with the set T_p. A_p would also work, but it would provide false signals. This can be illustrated with Example 5. The intuition is, if an OIE relation is mapped in multiple ways, it necessarily does not enhance its mapping degree. Hence, we chose T_p for this step, while use A_p for the evidence collection and confidence calculation.

Example 4. *Let's assume we have only ten relation instances for the* NELL *property airportincity. Hence each is of the form airportincity(*,*). This makes* $|T_p| = 10$. *Assuming that, 8 out of them have been mapped (mixture of Case (I) and (II) above), meaning there was a counterpart assertion in* DBPEDIA *after the* NELL *subject and objects were mapped to* DBPEDIA *instances. And 2 have no such counterpart assertion, i.e could not be mapped (Case (III) above). Hence, in this toy scenario,* $K_{\mathtt{airportincity}} = \frac{8}{10}$.

Example 5.
Let us consider just 2 instances in the set $T_{\mathtt{islocatedin}}$ *as*
{is located in(Kendall, Miami-Dade County), is located in(Kendall College, Chicago)}.
Using, the instance mapping techniques and subsequently finding the analogous DBPEDIA *assertion, we have the set* $A_{\mathtt{islocatedin}}$ *as*
{(is located in, n.a, n.a, n.a), (is located in, `dbo:campus`, `dbo:University`, `dbo:City`), (is located in, `dbo:city`, `dbo:University`, `dbo:City`)}.
Using $T_{\mathtt{islocatedin}}$ *for computing* $K_{\mathtt{islocatedin}}$ *would give* $\frac{1}{2}$, *since only the second instance actually can be mapped. While using* $A_{\mathtt{islocatedin}}$, *we would have* $K_{\mathtt{islocatedin}} = \frac{2}{3}$, *which is higher than the former value.*

In an attempt to define a likelihood score, we identified two closely related factors. First, the $\mathtt{conf}_p$ value which considers the observed evidences in the input data set. Second, K_p which defines the degree to which a relation can be actually mapped. These two can be combined as one factor to give an unified score. We do

not want to map relations with low K_p, even if there is high confidence for them. We combine K_p and $conf_p^i$ to define the factor called τ (*tau*) for each association rule i and is defined as,

$$\tau_p^i = \frac{(1 - K_p)}{conf_p^i};\ \forall i \in A_p \tag{16}$$

This equation combines the values from the confidence scores (Equation 14) and mapping degree (Equation 15). This is an unified way to quantify the goodness of a particular rule for a particular p having mapping factor K_p. In Table 11 we present 4 example relations, 2 from REVERB (*are in*, *grew up in*) and 2 from NELL (*airportincity*, *agentcreated*), which presents the actual values for each one of the major notations introduced so far. These values are not the complete set of tuples, but only a snippet of them. A low confident association with low K_p will give a high τ_p^i as seen with $\tau^3_{airportincity}$ in the table. While, a more confident association with high K_p minimizes the ratio, hence making the τ value less ($\tau^5_{airportincity}$ in Table 11). The last column in the table is a learnt threshold value for the maximum allowance limit of the τ. We discuss the learning scheme in Section 9.1.2. The whole idea of combining confidence and mapping factor was to incorporate the inherent differences across OIE data sets and also across relations within a data set. This is a generalized scoring scheme, which considers the two important aspects of the input data set.

9.1.2 *Threshold Learning*

Our final goal is to find a mapping of the OIE relations to a correct DBPEDIA property. In the previous section we have formulated a score for each association tuple as τ_p^i. It is clear from the examples in Table 11 that there can be a wide range of values for τ_p^i, for varying i and p. In this section we propose a technique to learn a correct threshold value for τ_p^i for a given p. We cannot have a single threshold value across all the relations since each relation is different with varying mapping degrees and confidence scores.

Intuitively, for each instance set of p (i.e. T_p), there can be three broad observable association patterns:

- a single high $conf_p^i$ association, among many others.
- multiple closely spaced possible KB properties with almost same $conf_p^i$

A_p	p	K_p	i	d_p	dom(p)	ran(p)	τ_p	$\hat{\tau}$
$A_{\text{are in}}$	are in	15%	1	isPartOf	Village	AdministrativeRegion	29.0	3.07
			2	country	City	Country	19.67	3.07
			3	isPartOf	Settlement	Settlement	20.3	3.07
			4	family	Plant	Plant	29.0	3.07
			5	...	...	...	...	...
$A_{\text{grew up in}}$	grew up in	79%	1	hometown	Person	Settlement	**0.42**	0.52
			2	residence	Person	PopulatedPlace	**0.33**	0.52
			3	deathPlace	Boxer	Settlement	19.17	0.52
			4	hometown	Band	Town	115.0	0.52
			5	citizenship	Person	Country	3.02	0.52
			6	...	...	...	...	...
$A_{\text{airportincity}}$	airportincity	55%	1	location	MilitaryStructure	-	150	1.21
			2	location	Airport	-	**0.86**	1.21
			3	isPartOf	Settlement	Settlement	150	1.21
			4	isPartOf	Settlement	-	37.5	1.21
			5	city	Airport	-	**0.86**	1.21
			6	...	...	...	...	...
$A_{\text{agentcreated}}$	agentcreated	9%	1	notableWork	Writer	Play	496.6	3.12
			2	notableWork	Writer	TelevisionShow	3973	3.12
			3	occupation	Person	Book	12.7	3.12
			4	occupation	Settlement	-	37.5	3.12
			5	knownFor	Scientist	Book	3973	3.12
			...	...	...	...	...	...

Table 11: A snippet of the actual associations presenting a positive example with 4 example OIE relations: *airportincity* and a negative example with *agentcreated* (from NELL); *grew up in* and a negative example with *are in* (from REVERB). A blank value ('-') is attributed to a missing instance type in DBPEDIA or often scenarios where there are no DBPEDIA relation to capture the semantic relationship expressed by p.

- multiple candidates with low $conf_p^i$

We aim at modeling these different scenarios which would select the first two cases but not the third one. The rational is, any association rule with a low confidence is not an appropriate predictor for p. Now, when we include K_p into the equation, we get the confidence values translated as τ values. From the expression of τ in Section 9.1.1, it is clear that the best score is attained when a particular rule confidence for a relation p attains maximum. We denote this minimum tau value simply as τ_p^{min}. This is of particular interest since, this is the best possible score a particular relation can have. For different p this value changes. We capture this variation by presenting a distribution pattern for τ_p over K_p with detailed figures in the experiments section (Section 11.1.1).

It must be observed that, with relations with very low mapping factor K_p, there is a drawback in using them for the purpose of relation mapping task. A low K_p indicates low mapping degree which is caused due to dearth of adequate evidences supporting a mapping $p \rightarrow d_p$. Hence, the observed evidences are too weak to make a strong decision on the mapping. If we chose such relations, we are then trying to deduce a relation mapping based on low evidence, which in our opinion is not justified. However, relations with higher K_p are devoid of this problem. Hence, we select a threshold value of K_p which is appropriate for a given data set. The choice of this threshold is different across NELL and REVERB. In Section 11.1.2, we report this behavior with some experiments and provide rationale for our choices. The set of experiments allow us to make an empirical decision in choosing a K_p which is *high* enough to make conclusive deductions. We define a set D consisting of pairs $\{\ldots, (K_p, \tau_p^{min}), \ldots\}$, for all $p \in T_p$ such that K_p is higher than the empirically determined threshold value. We fit a linear regression model on set D, and compute the squared loss for each. The threshold which gives the minimum loss is finally chosen. In Section 9.2 we briefly discuss linear regression as relevant for the error calculation in our approach.

With such a linear predictive analysis method, we can have an estimate of τ, defined as $\hat{\tau}$ for every K_p. Note that, we trained our model using the data points attained using the maximum confidence (analogously τ_p^{min}), hence, the linear model is an automatically learnt threshold on τ_p. We use $\hat{\tau}$ to compare with every τ_p^i, $\forall\, i \in A_p$. Some scores fall below the prediction (the likely associations) and some are way above it (less likely ones) (refer to Table 11, correct association values are marked in bold). The likely associations allow us to select the rule with acceptable τ values. These are the rules which are considered to have higher credibility than the rest. We would re-visit the three goals we set in the beginning of this section and show that the computation scheme actually addresses them well.

- Multiple associations but a single one with a high $conf_p^i$. This makes $\tau_p^i \simeq \hat{\tau}$, since the high confidence association will lead to the best τ_p score.
- Multiple closely placed associations with almost same $conf_p^i$, making $\tau_p^i \simeq \tau_p^j \simeq \hat{\tau}$; $i \neq j$. (refer Table 11, $\tau^2_{\mathtt{airportincity}}$ and $\tau^5_{\mathtt{airportincity}}$)
- No clear winner, but multiple candidates with low $conf_p^i$ making $\tau_p^i \ggg \hat{\tau}$ (refer Table 11, $\tau^1_{\mathtt{agentcreated}}$, $\tau^5_{\mathtt{agentcreated}}$)

Recollect from the association tuple (Expression 11.1.2) and the association rule (Equation 13), that we can have a direct mapping for the corresponding DBpedia relation. In our entire analysis, we never used the DBpedia relation for any direct computation but let its domain/range restrictions allow us to find the appropriate mapping.

9.2 LINEAR REGRESSION

Model Fit: In this part we work out the error calculation for the linear model we employ for our relation matching step. Given a set of data points, $\mathcal{D}$ in a 2-dimensional feature space we can estimate the linear dependency between the two variable using a linear regression model. In our context, $\mathcal{D}$ contains the point $\{\ldots, (K_p, \tau_p^{min}), \ldots\}$, for every particular OIE relations p. Hence, the data set can be represented as, $\forall p \in T_p$,

$$\mathcal{D} = \left\{ (K_p, \tau_p^{min}) \mid K_p \in \mathbb{R}_{\geqslant 0}; \quad 0 \leqslant K_p \leqslant 100 \right\} \tag{17}$$

We select a subset, $\mathcal{D}'$ ($\subset \mathcal{D}$), for which we try to model with a linear regression line. The only difference the subset has is that every data point has K_p value greater than a particular threshold. The choice is made based on the linear trend as observed in Figure 17 and 16. For this subset of data points, fitting a regression line necessarily means to find a straight line passing through the set of points in a way such that it is the *best* fitting line. Best usually is defined as minimizing the sum of the squared errors. Hence, the dependent variable τ_p^{min} can be predicted by the linear formulation with K_p as,

$$\hat{\tau}_p = \theta_0 * K_p + \theta_1 \tag{18}$$

where $\hat{\tau}_p$ is an estimate of τ_p^{min} and θ_0, θ_1 are the parameters. The sum of errors is given by,

$$\text{error} = \sum_{p}^{|\mathcal{D}'|} (\hat{\tau}_p - \tau_p^{min})^2 \tag{19}$$

which is simply the sum of the squares of the difference in the actual value of τ_p^{min} and its estimated values for all the points in the data set. Intuitively, if the actual and estimated values are identical, that signifies zero error at that point and if that is true for all the data points, then it is the perfect fit. Or, geometri-

cally, all the points lie on the regression line. It is evident from Equation 18 that the parameters play an important role, θ_0 determines the slope of the regression line and θ_1 gives the intercept. The choice of these two parameters can be easily solved by gradient descent algorithm, which involves altering the values of the two parameters iteratively, such that the cost function, the error (in Equation 19) is minimized in every step. However, to avoid falling into local minima, stochastic gradient decent may help. We used the standard java math libraries from Apache foundation to estimate these parameters[4].

Mean Squared Error (MSE): Once the linear model is generated, we can use it to compute the error for any arbitrary data points. The sum of errors would give us a sense of the goodness of fit for the linear model. While calculating the error, we now also consider the data points which were not considered previously during model fitting. Thus the MSE is given by,

$$\text{MSE} = \frac{1}{|\mathcal{D}|} * \sum_{i}^{|\mathcal{D}|} [(\theta_0 * K_p + \theta_1)_i - \tau_i]^2 \tag{20}$$

For any arbitrary point i, the estimated value is given by $(\theta_0 * K_p + \theta_1)_i$ while the actual value is τ_i which readily allows us to compute the error. Thus a lower MSE indicates better model fitting. We chose MSE as an error measure, however, one can also choose Mean absolute Error (MAE) as an alternate performance indicator. Is is to be noted that the index i runs over $\mathcal{D}'$ while model fitting but for error calculation, it uses the whole data set. In Figure 17, we vary the threshold value gradually starting from as low as 1% and moving till 50%. For each threshold, a new linear model is learnt, and fitted on the data set. This gives varying values MSE for varying threshold. We observed that, extreme low and high threshold led to worse MSE. Finally, we select the threshold values which gives us the lowest MSE.

4 http://commons.apache.org/proper/commons-math/

10

CLUSTER BASED APPROACH

In this chapter we outline an alternative relation mapping approach. The idea in this chapter is to have a natural grouping of the relations which are semantically synonymous. Once a natural grouping is achieved, it can be safely assumed that they all bear similar semantics. Such a group is referred to as a cluster of relational phrases. In this regard, we perform two clustering methodologies. First, cluster only the OIE relations without any additional seed inputs from KB relations. The clusters thus created can be mapped to one or more KB relations. This is very analogous to the rule based method proposed in the previous chapter, the only difference being, now we treat clusters of relations as a unit instead of single relations. This is a rational choice to group synonymous phrases into a cluster which can then be treated as one collective unit. The importance of this particular scheme is more applicable to Reverb than to Nell. The former data set often contains relation phrases which are natural language textual phrases and it is a natural choice to bundle them together. While the relations in Nell are necessarily not textual phrases but normalized into a form unique to Nell's own internal hierarchical structure[1]. Second, cluster together both the OIE and KB relations with seeds from the later. This forms a group with a collection of OIE relations with one or more semantically similar KB relations. We explore these two methodologies and design our system workflows around these.

For computing these clusters, we explore and analyze two different schemes: graph-based and vector-based. For the graph-based method, we use Markov clustering (Section 10.2) in which nodes represent relational phrases and edges weigh the degree of similarity of the connected phrases. As a vector-based scheme we opt for k-mediod (Section 10.1) where every relational phrase is represented as a vector in feature space. Although the clustering technique is more applicable to the Reverb data set, our algorithm makes no basic assumption of the underlying data sets and is oblivious to any specific structure within the relations.

1 http://rtw.ml.cmu.edu/rtw/kbbrowser/

We briefly outline the major differences between the graph-based and vector-based schemes, before getting into the methodology specifications. Consider a set of items which needs to be grouped in sets of closely related items. Vector-based clustering necessarily tries to represent each data points (or items) in a high dimensional feature space, typically dimension greater than 2. The item (data) representation is usually made of from a vector of features. Hence the name vector-based clustering. We will discuss these details in Section 10.1. In contrast, a graph-based scheme purely trusts on the pairwise affinity scores between two data points and the assumption that the elements are distinct nodes in a graph. Hence, the former has the onus of representing data points as a vector while the later has the onus of defining a similarity function between items.

10.1 VECTOR BASED CLUSTERING

Given a set of n data points, the k-$medoid$ clustering [JH10] algorithm initializes k medoids which act as central points. Every other data point is assigned closest to one of these medoids. This scheme iteratively toggles between two steps, initialization of a medoid and assignment of the data points nearest to the chosen medoid. After the first iteration of assignment phase, the new mediod is calculated by minimizing the cost[2] for the configuration. The new mediod giving the best configuration (least cost) is chosen and the re-assignment is repeated. The algorithm runs till convergence, i.e. no more newer medoids can be found leading to a lower cost. This is a stable configuration with a set of k disjoint clusters. This method is similar to the k-means algorithm [HW79] but with the only difference that for each iteration, the actual data points form the new medoids, unlike the means of data points as done in k-$means$ clustering. We choose k-$medoid$ scheme over its sibling, due to its better robustness to noise and outliers [Vel12].

In our setting, every OIE property is represented as a vector of n-elements. We refer to the elements as features and likewise the representative vector as a feature vector. In particular, we use two feature spaces, one for domains and the other for ranges for the OIE relations. They are exactly identical just that the features are different. The need for two spaces will be clarified shortly. For simplicity, consider

2 Cost is usually computed as the sum of all the pairwise Manhattan distances between the points. One can use other measures as well.

three REVERB relations: *is situated in*, *is a region in* and *is a part of*. Let each has exactly two instances. We present the instances for one of the relations here:

is a region in (Asturias, Spain)
is a region in (Amsterdam, Nederland)

We describe the details with one relation and its instances, the description is the same for the other relations as well. With a refined instance mapping already computed, the mapping for the subject terms in the two relation instances above are: {. . . , (*Asturias*, `db:Asturias`), (*Amsterdam*, `db:Amsterdam`), . . . }. We present the mappings in the form of mapping pairs as defined by $\mathcal{M}$ (Expression 1). The dots denote the existence of the object mappings as well as the other mappings from rest of the relation instances. Once correctly mapped, it is not hard to find the domain of the relation, *is a region in*. In this case it is `dbo:Settlement` and `dbo:City` (respective instance types of `db:Asturias` and `db:Amsterdam`). Analogously, the other two of the remaining OIE relations would have similar vector representation which are presented below:

$$\begin{aligned} \textit{is situated in} &\equiv < \texttt{dbo:Settlement}, \texttt{dbo:City}, 0 > \\ \textit{is a region in} &\equiv < \texttt{dbo:Settlement}, \texttt{dbo:City}, 0 > \\ \textit{is a part of} &\equiv < 0, \texttt{dbo:City}, \texttt{dbo:Album} > \end{aligned}$$

In this example, the domain space spans over three axes: `dbo:Settlement`, `dbo:City` and `dbo:Album`. Hence, the relation *is a region in* spans over 2 out of the 3 axes in the domain feature space. We see that the feature vector is of length 3 and each feature is either present or absent. For instance, "*is a part of*" has `dbo:Settlement` feature absent, hence marked 0. This is an over simplified example with 3 dimensions only to make the intuition transparent. In reality we have extremely high dimensions for a given property. In our implementation, we use integer valued ids to identify the DBPEDIA classes and also maintain an inverse lookup table to map the ids back to its respective textual representations. Ideally, we had extremely long vectors, whose length was typically in the order of the input data size. Generally, a feature vector for some relation p would look like:

$$p \equiv < (1 = 0), (2 = 0), (3 = 0), (4 = 1), \cdots >$$

The feature element (1=0) means the DBPEDIA class indexed "1" (e.g. `dbo:City`) is not a feature for this relation, similarly (4=1) means the class indexed with "4" is a

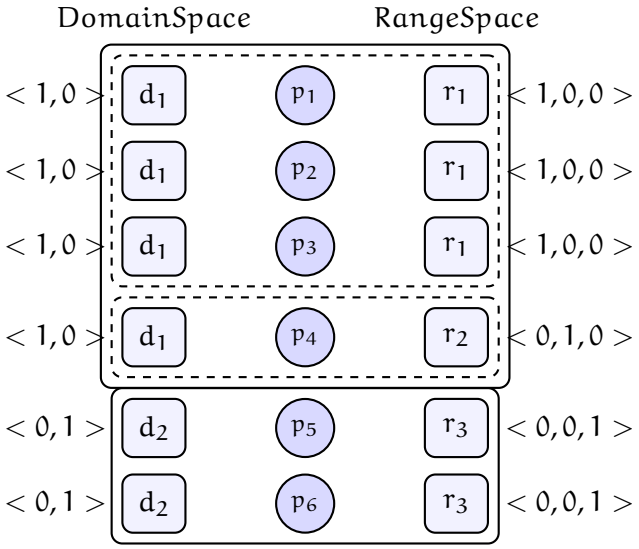

Figure 13: A schematic representation of the k-medoid based clustering technique using the OIE relation vector representation in two different feature spaces. Adjacent to every domain and range values, the feature vector is written, for instance, $p_4 \equiv < 1, 0 >$ in the domain space.

feature. It is obvious that such feature vectors can be extremely sparse with lots of zeros. We used the sparse vector data structure[3] for this purpose which maintains a extremely compact form of the above vector with just the existing features.

We maintain a similar feature space for the range as well. The broad idea is to have a projection of a given relation in two feature spaces; cluster based on one projection (domain features) and then refine further with the other projection (range features). This is not a strict order, one can also start from the range space with identical outcomes. This is schematically represented in Figure 13, where we present six properties with their respective associated domain and ranges. We observe that, domain space is two-dimensional with d_1 and d_2 as axes, while range space is three-dimensional with axes r_1, r_2 and r_3. Thus, in the range space, p_2 has a representation of <1, 0, 0>. We first cluster relations on the domain space (represented by thick rectangles) giving us two clusters {p_1, p_2, p_3, p_4} and {p_5, p_6}. Now, considering the projections in range space as well we find further sub-clusters (represented by dotted rectangles), finally leading to three clusters {p_1, p_2, p_3}, {p_4} and {p_5, p_6}.

3 http://lenskit.org/. This project specializes in developing tools for recommender systems, however the data structures and collections are available via Maven/Github

We must point that this is not a strict vector representation. One can also have a combination of features to represent one unified boolean feature. For instance, $f_i f_j$ can be one feature denoting if DBpedia class i is a domain and class j is the range. But the problem is it would make the feature space extremely huge. If there are d ($\gg 1$) classes as domains and r ($\gg 1$) classes as range, the combined feature space would be d X n. Whereas with the split feature space, we have to deal with either d or n values at a time. Hence we chose this scheme of split feature space representation. Even then, it is very natural that the individual vectors can be very large and extremely sparse with lots of zeros. We further optimized it by our choice of data structures which are designed to deal with such sparseness. They maintain an extremely compact form of the vectors with just the true features.

The principle idea we exploited in this approach is, if two properties have their domains and ranges similar they are likely to be similar. We used the standard machine learning library in java, javaml, for computing the vector-based clusters[4]. The methodology can be viewed as an incremental clustering scheme where the properties are first clustered on the domain space, and then the ones already clustered are further clustered on a finer level using the range space. The final output is a bunch of OIE relations clustered based on both the spaces. In the experiments in Section 11.2.3, we compare these different variants of clustering and present the results.

10.2 GRAPH BASED CLUSTERING

In contrast to the vector-based method, graph-based technique considers the cluster elements as nodes in an undirected graph. The clustering is based solely on the affinity score between two adjacent nodes. The pre requisite for this scheme is to define a likelihood score or similarity measure between a pair of relation phrases, which is presented next in Section 10.2.1. The following Section 10.2.2 presents the idea of markov clustering, which is based on random walks in a graph.

4 http://java-ml.sourceforge.net/api/0.1.5/net/sf/javaml/

10.2.1 *Similarity Metrics*

We introduce a similarity measure $sim(r_i, r_j)$ which captures the degree of similarity between any two property phrases $r_i, r_j (i \neq j)$. In defining the similarity, we looked into two major aspects:

- exploit the input data set to extract evidences which allow to quantify the similarity between r_i, r_j.
- exploit the rich semantics via external sources to define a similarity function.

For the former requirement, we have the set of relation instances for the two relations in question. These can be exploited to define a similarity co-efficient, in particular, we use the overlap similarity $ov(r_i, r_j)$ for our task. Let f_{r_i} and f_{r_j} be the set of relation instances for the relational phrases r_i and r_j, respectively. If n denotes the number of instance pairs where both the subject and object terms are in common across both sets, then the overlap similarity is, in our context, defined by,

$$ov(r_i, r_j) = n/min(|f_{r_i}|, |f_{r_j}|) \tag{21}$$

One can also consider using other measures like Jaccard for this similarity. We opted for the overlap co-efficient for attaining higher similarity scores in general. This can be illustrated better with the Example 6.

Example 6. *Let us consider two relations r_1 and r_2. For each one of them, we define few relation instances as follows:*

$r_1(a_1, b_1)$ $\quad$ $r_2(a_1, b_1)$
$r_1(a_2, b_2)$ $\quad$ $r_2(a_2, b_2)$
$r_1(a_3, b_3)$ $\quad$ $r_2(a, b)$
$r_1(a_4, b_4)$

Clearly, $n = 2$, since there are just two pairs ((a_1, b_1) and (a_2, b_2)) across the two relation instance sets. . And $|f_{r_1}| = 4$; $|f_{r_2}| = 3$
Using, Jaccard co-efficient, the score would have been $\frac{n}{|f_{r_1} \cup f_{r_2}|} = \frac{2}{7} = 0.285$
But, with overlap similarity, the score will be $\frac{n}{min(|f_{r_1}|, |f_{r_2}|)} = \frac{2}{3} = 0.66$

The example shows that overlap similarity tends to give higher scores. This is important for us since our goal is to find a similarity score between two relational

phrases. Now, for typical open information extraction, the size of the relation instance sets are not of prime importance. For instance, if the extraction produced just one instance of *is the writer of* and 1 million instances for *is the author of*, then jaccard similarity will make these two phrases very less similar, but that is not the case as they are semantically similar and should be weighed higher. This defeats our purpose. But, overlap similarity exactly avoids this scenario and better suits our use case.

Now, measuring the likelihood with overlap coefficients can capture the relations having similar instances as arguments, but often it might not be the case. We need something more sophisticated to determine if relations like *are essential in* and *is vital in* are similar even if they might not have any common instances between them. We use Wordnet [Mil95] as our lexical reference for computing similarities in these complicated cases. In particular, we used a similarity API[5] which internally uses a hybrid approach involving statistics from a large corpus [KHY+14, HKF+13]. The RESTful [RR07] API allows to retrieve the score for a given pair of relation phrases r_i and r_j. We denote this score as $wn(r_i, r_j)$.

Eventually, we make a linearly weighted combination of these two weights to define our intra-node affinity score $sim(r_i, r_j)$, which is given as,

$$sim(r_i, r_j) = \beta * ov(r_i, r_j) + (1 - \beta) * wn(r_i, r_j) \tag{22}$$

where, β, is a weighing factor ($0 \leqslant \beta \leqslant 1$). In Section 11.2.2, we present an empirical analysis for the choice made for β and discuss its effect on the overall clustering task. Applying the measure to the phrases r_i = *is the capital of* and r_j = *is the capital city of*, for example, we obtain the score 0.719 with $wn(r_i, r_j) = 0.829$ and $ov(r_i, r_j) = 0.585$ if we set $\beta = 0.45$.

Additionally, one can also use an n-gram[6] overlap similarity as an additional measure. The given phrases can be splitted into 2/3-grams and a cosine similarity score can be computed. This works well for phrases which are similar token wise, for instance *is located in*[7] and *is city located in*. But, it gives a low score for a pair of relations with similar semantic sense but low token overlap, like *spouse of* and

5 http://swoogle.umbc.edu/SimService/
6 a sequence of n items where items can be letters, words, syllables etc.
7 Using characters, a 2-gram would look like [is, _l], [_l, oc], [oc, at], ...

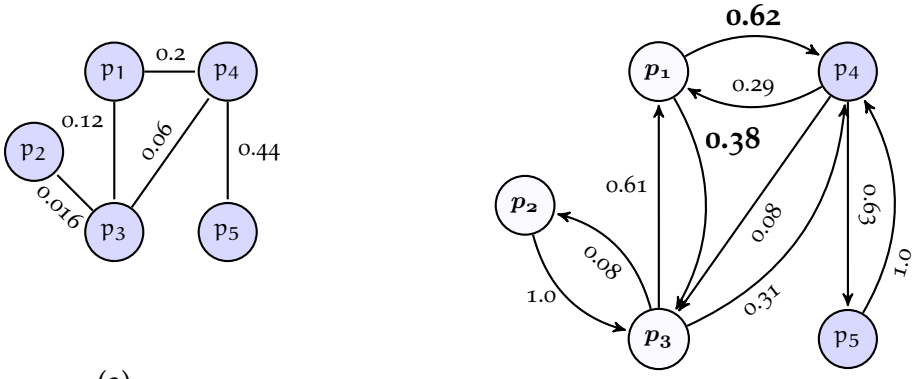

Figure 14: (a): A weighted undirected graph representing similarities between nodes. (b): same graph showing the transition probabilities. The directed edges represent the probability of moving to the connecting node. Note that the bold values add up to 1. p_1: *is a village in*; p_2: *is a town in*; p_3: *is a suburb of*; p_4: *currently resides in*; p_5: *currently lives in*; nodes of same color are eventually in the same cluster.

married to. Moreover, there is a wide range of other similarity measures one can use for the purpose, but we chose the wordnet API and the overlap. The former is a one-step solution which considers distributional semantics, statistical coverage and token similarities but not the instance overlap. This is nicely complemented with the overlap score, which on the other hand misses the semantics achieved by wordnet. Hence, in our opinion, the above similarity measure is a simple yet comprehensive choice for the task. We consider two broad aspects which determine pattern similarities and linearly combine them as one unified score.

10.2.2 *Markov Clustering*

We apply markov clustering (MCL) to generate the clusters that we finally map to the properties in the target ontology. MCL works well with large inputs and it has been applied successfully for finding semantically similar words [DWL+04]. Also this graph clustering methodology is known to perform better than other clustering algorithms [BvH06]. The primary work on MCL was done by van Dongen [vDoo]. In Figure 14(a), we depict an example for a set of five Reverb properties p_1 to p_5 (the exact relation phrases are stated in the figure caption). The nodes represent individual property phrases. A weighted edge connecting two nodes denotes their affinity score (introduced as sim in Section 10.2.1). A

missing edge denotes absolute dissimilarity, for instance, there exists no edge between p_5 and p_3. Now in Figure 14(b) we transform the original graph into a directed structure. The weights are converted to probabilities. For instance, node p_1, is connected to p_3 and p_4, hence the transition probability to p_3 is given by $0.12/(0.12+0.2) = 0.38$ as shown in Figure 14(b) with a directed edge. The sum of all the transition probabilities from a particular node has to be 1.0. There are couple of observations that need to be made while creating the transition graph from Figure 14. First, the original graph was an undirected one, while the transformed graph is a directed one. The directed edges are important to capture the flow direction towards or from a node. These edges now denote probabilities instead of edges as it was with the original graph. Formally, a graph with n nodes has a probability matrix, $\mathcal{P} \in \mathbb{R}^{nXn}$. An element $p_{ij} \in \mathcal{P}$ represents the probability of node j transiting to node i, where i, j are the row and column indices ($1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant n$). And so essentially

$$\sum_{i=1}^{n} p_{ij} = 1.0 \tag{23}$$

thereby making $\mathcal{P}$ a column stochastic matrix. Observe that the sum is over a column which intuitively says the respective probabilities of jumping to the nodes presented in each row. (The finer details will be clarified with a working example presented in Figure 15).

The markov clustering algorithm is driven by the idea of a random walk [Pea05]. The algorithm is performed by simulating an agent moving randomly across the nodes and along the directed edges in the transition graph (Figure 14(b)). The probabilities of each edge from each node define how likely it is for the agent to follow that edge. Referring to the figure, if the agent is at node p_1, then in its next random step, it is more likely to jump to p_4 having a higher probability than jumping to p_3. These jumps are randomly repeated over and over again. After a large number of such jumps, the agent tends to remain or hop around those nodes which have higher probabilities within themselves. This can be observed with nodes p_3 and p_4 which tend to form a close pair and hence a cluster.

Donjen et al., [vDoo] have simulated the concept of random walk using markov chains, where the steady state probability distribution is obtained by an iterative product of the probability matrix. Hence the n^{th} state probability is given by $\mathcal{P}^n$.

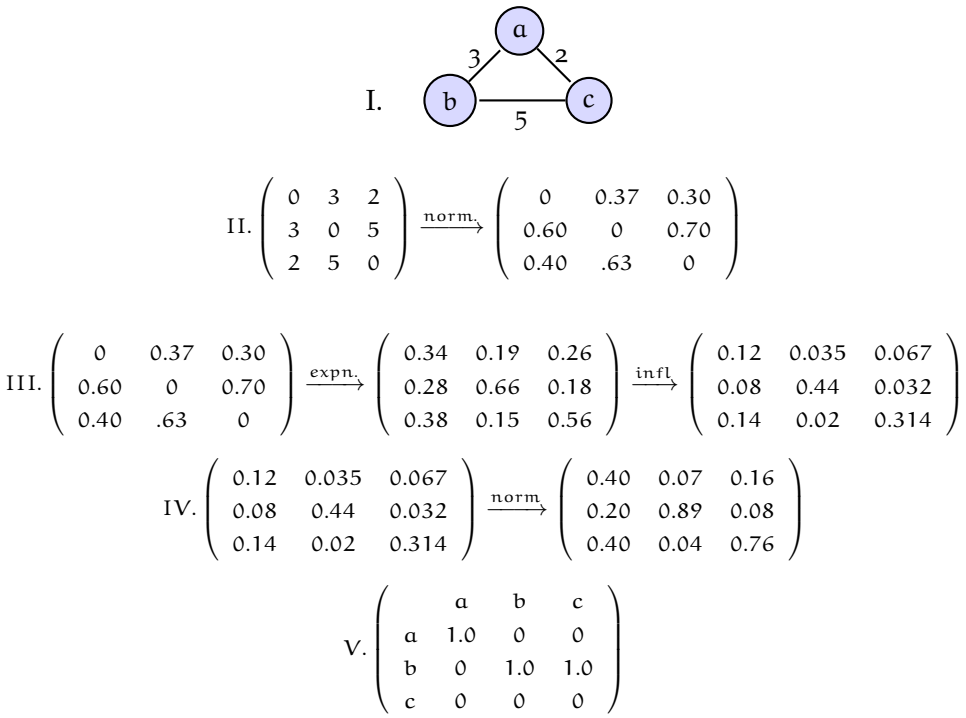

Figure 15: (I) A sample graph with 3 nodes and associated edge weights denoting the affinity scores between the nodes. (II) the equivalent matrix representation of the graph with simultaneous normalization (denoted as *norm.*) step performed. (III) Performing expansion and inflation step on the resulting column stochastic matrix from the previous step. (IV.) Normalization step to repeat with the step (III) again. (V) after infinitely many steps, the final steady state matrix.

The idea is to make a stronger link even stronger and weaker ones even weaker, with these repetitive steps. The notions mentioned here are from the original work of Donjen [vDoo]. We try to clarify those notions with an entire working mechanism with the sample graph of Figure 15(I). This is a simplified graph and has been chosen on purpose to present the matrix operations that follow. Given such an undirected graph the clustering algorithm performs the following steps repeatedly and in succession,

1. Normalize: Converting the matrix representation of the graph into normalized form. This is the matrix $\mathcal{P}$ as referred to earlier. This is achieved by dividing the individual matrix elements by its column sum. This is shown in Figure 15(II). It must be noted that the original graph is symmetric but the normalized one is no longer symmetric. Observe that $\mathcal{P}$ is now a column stochastic matrix and follows the property given by Equation 23.

2. Expansion: The transition matrix ($\mathcal{P}$) is raised to the power of 2 i.e simply matrix squaring. The purpose of this step is to connect the regions of the graph. This is shown in Figure 15(III) with the arrow marked *expn.*.

3. Inflation: This consists of performing a hadamard power of the resulting matrix. This involves raising each element of the matrix to the power of ϕ, the *inflation factor*. Hence, referring to the Figure 15, the element of the first row and first column is obtained by doing $0.34^2 = 0.12$, since in this example we set ϕ as 2.

The process does not terminate here, but is repeated again over the above mentioned steps till a saturation is reached. We show this re-normalizing step only in the next line of Figure 15(IV). This makes the final matrix column stochastic again. The following steps are repeating the cycle of expansion, inflation and normalization. These repeated steps keep making the partitions within the graph more and more evident and thus leading to logical segments knows as the clusters. The final matrix gives the stable state distribution as shown in Figure 15(III). Every row is defined as *attractors*, and the column nodes are being attracted [vDoo]. Hence, from the figure, node b is attracting b and c as seen in the positive values in their respective cells. Thus (b, c) and (a) form two clusters. We used the available implementation[8] for our experiments.

There are two major challenges in applying the method successfully. First, the choice of the inflation factor ϕ is crucial. Setting it too small, makes the cluster too coarse and vice versa. And second, our initial experiments showed that with a reasonable ϕ, some of the final clusters had further local sub-clusters. We implemented an extension to the algorithm which allows to find these local sub-clusters. For a given ϕ, we apply markov clustering on each of the cluster, ignoring the influences from all other clusters. In Section 11.2.2, we report about experiments to automatically determine the optimal choice of ϕ, which avoids a random selection of this parameter.

10.3 NAIVE CLUSTERING

Apart from the two clustering methods mentioned in the last two sections, we implemented another version of the graph-based clustering. We refer to this as a

8 http://www.micans.org/mcl/index.html?sec_software

naive technique. The idea of this scheme is influenced by the vector-based method, especially with respect to the choice of medoids. For a given set of n OIE relations, a random subset of k relations are chosen ($k \leqslant n$). These k relations can be considered seed points and necessarily define the size of the final cluster set generated. As a following step, each one of the remaining n - k relations are assigned together with the one of the k seed points having the highest similarity measure. It is intuitive that the quality of the output clusters heavily depends on the input seeds. Hence, we draw k random seeds repeatedly for a large number of times. For each seed input set, we determine the cluster quality. Eventually we choose the seed configuration which leads to the best[9] cluster output. The repeated sampling step is performed to ensure that we improve as much as possible with this approach. This scheme is naive for obvious reasons. First, it has heavy dependency on the seed quality and variations. Second, the cluster elements are assigned to the closed data point and thereby it makes the entire clustering very coarse grained. It completely ignores any sub grouping which might be possible. In the experiments section, we present results which compare these three clustering schemes. This naive clustering is based on very simple heuristics and can be considered as a baseline scheme.

The CL module generates as output clusters of relational phrases having similar meaning, i.e., that can be correctly mapped to the same property. We implemented three different clustering methods including the trivial case for which we treat each relational phrase as a one-element cluster. There are two non trivial clustering methods. One works with DBpedia input seeds and the other one works without DBpedia object relations as input seeds. These two methods require a similarity score computed for each pair of relational phrases in the input OIE data set. The similarity computation module is hence applied only on these two non-trivial clustering methods. As illustrated in Figure 2, we execute three different workflows wf_1, wf_2 and wf_3 that differ with respect to the applied clustering technique. The output of this module are clusters of relational phrases, which includes the simplest case where the phrases constitute a one element cluster, i.e for wf_1. The clusters generated by wf_2 and wf_3 are forwarded to the IM module (dashed arrow in Figure 2), which is executed again to improve the instance mapping due to better statistical coverage of clusters compared to the coverage of individual relational phrases.

9 In Section 11.2.1 we provide the quantitative notion of best.

Algorithm 2 Algorithm for Rule based Relation mapping

Require: f^+ facts from OIE system; clusters, IM

1: $r_{maps} \leftarrow null$ ▷ relation mappings collection
2: $assoList \leftarrow null$ ▷ rule associations
3: **function** RULEBASEDRM(f^+)
4: $assoList \leftarrow getAssociations(IM, f^+)$
5: $model \leftarrow learnModel(assoList, clusters)$
6: $r_{maps} \leftarrow predict(model, assoList, clusters)$
7: **return** r_{maps}

8: **function** GETASSOCIATIONS(i_{maps}, f^+)
9: $associationList \leftarrow empty$
10: **for** $fact$ in f^+ **do**
11: sub, p, obj from $fact$
12: **if** $rel \in clusters$ **then**
13: $sub_{KB} \leftarrow i_{maps}.get(sub)$
14: $obj_{KB} \leftarrow i_{maps}.get(obj)$
15: $rel_{KB} \leftarrow$ query DBPedia endpoint (sub_{KB}, obj_{KB})
16: $domain \leftarrow getType(sub_{KB})$
17: $range \leftarrow getRange(obj_{KB})$
18: $associationList.add(p, rel_{KB}, domain, range)$
return associationList

19: **function** LEARNMODEL(assoList, clusters)
20: $\mathcal{D} \leftarrow empty$
21: **for** $p \in clusters$ **do**
22: compute K_p from assoList
23: compute maximum $conf_p$ from assoList
24: compute τ^p_{min} from assoList
25: $\mathcal{D}.add(K_p, \tau^p_{min})$
26: $model \leftarrow \mathcal{D}.regress$
27: **return** $model$

28: **function** PREDICT(model, assoList, clusters)
29: $mappings \leftarrow empty$
30: **for** $p \in clusters$ **do**
31: compute K_p from assoList
32: compute $conf_p$ from assoList
33: compute τ_p from assoList
34: $\hat{\tau} \leftarrow model.predict(K_p)$
35: **if** $\tau_p \leqslant \hat{\tau}$ **then**
36: $mappings.add(p, rel_{KB})$
37: **return** $mappings$

Algorithm 3 Algorithm for Generating Clusters

Require: p: set of relations; mode

```
1: function CLUSTER(p, mode)
2:     C ← null
3:     p ← S^R
4:     if mode = wf1 then
5:         C ← every element from p
6:     else                                          ▷ wf2 and wf3
7:         if mode = wf3 then
8:             p ← S^R ∪ T^R
9:         simCompute(p)                             ▷ similarity scores
10:        C ← markov cluster on p
11:    return C
```

10.4 ALGORITHM

We use this section to present a collective summary in the form of all the relation matching algorithms we have discussed so far. We present in particular, the rule mining based algorithm, the clustering algorithm and finally the complete relation matching algorithm. In designing the algorithms, we considered the variations in workflows to tackle the different input settings.

Algorithm 2 presents the rule based relation mapping algorithm. where the main function is RULEBASEDRM. It is built with the three major components which have been also presented as sub-routines in the algorithm. The input to the RM module are clusters, instance mappings and f^+ facts. This is seen in the workflow illustrations in Figure 3. For the basic clustering mode (wf_1), clusters contain exactly one single OIE relation. While in advanced clustered mode (wf_2), they contain groups of relations with the t^{th} cluster denoted as c_i as seen in Expression 11. This algorithm clearly indicates the execution flow. It is to be noted that p in Line 12 denotes a cluster c_i. Hence, in wf_1, the association is for each OIE relation, while in wf_2 it is for each cluster. The same idea follows later in model learning and prediction routines. We present this algorithm here, since the rule based relation mapping can work with both clustered and non-clustered relations.

As the next step we also present the simple clustering Algorithm 3. We do not present the detailed markov clustering since, that is not the core contribution of this thesis. As observed, the algorithm requires only the set of relations and the

Algorithm 4 Algorithm for Relation mapping

Require: f^+ facts from OIE system; IM; $\mathcal{C}$: clusters; mode

1: $r_{maps} \leftarrow null$ ▷ relation mappings collection
2: **function** RM(f^+, IM, $\mathcal{C}$, mode)
3: **if** $mode = wf_1$ or wf_2 **then**
4: $r_{maps} \leftarrow ruleBasedRM(f^+, \mathcal{C}, IM)$
5: **else**
6: $r_{maps} \leftarrow from\ \mathcal{C}$
7: **return** r_{maps}

workflow mode. The final result is a set of clusters. We maintain the same cluster notation as we introduced in Section 8.2 and explicitly defined in Expression 11. We also maintain the same notations for referring to the set of OIE relations ($\mathcal{S}^R$) and set of KB relations ($\mathcal{T}^R$). The wf_3 makes explicit use of the KB relations by mixing with the OIE ones. We initialize the default set of relations to be that of the OIE relations, but for wf_3, Line 7 updates the set to the union of OIE and KB relations.

As the final block, we present the complete relation matching Algorithm 4, which combines the rule based and clustering-based mapping techniques. It requires the f^+ from OIE input, but it is only for the rule based module. The cluster-based module is easier in the sense that it requires very few inputs. The rule based step makes the call to Algorithm 2 on line 4. The rule base method also requires clusters as one of its inputs. It is very important to note the flow of executions for this algorithm under different workflows and its correspondence with the illustrations in Figure 3. The input to this algorithm are clusters which have been generated by Algorithm 3. In the later sections, when we talk of the integration module, we will use the RM algorithm and we will observe how the clusters are pre-generated and fed into this module. The sequence of execution flows discussed in these algorithms strongly adhere to the workflow illustrations we have discussed in Section 3.2.

11

EXPERIMENTS

We discussed, an association rule mining based approach (Chapter 9) and also a clustering based one (Chapter 10) as two means for performing the mapping of the OIE relations to the target KB. Here, we make detailed analysis of these broad methods and present some of our empirical results for these broad individual methodologies. It must be noted that the rule mining approach is applicable to both NELL and REVERB, but that is not the case with cluster based methods. The clustering scheme is only applicable for relation phrases in the natural language form and not normalized forms as observed in NELL. Hence, for the later we have the results only for REVERB. In particular, we broadly split our empirical analysis over two sections: in Section 11.1 we present the details for the rule mining scheme and in Section 11.2 we have the cluster based evaluation results. For the rule mining methodology, we perform detailed data analysis task on each of the two data sets which is a pre-requisite step and supports our rational for some of the choices made. The rule mining scheme was employed on both NELL and REVERB and we present the values in Section 11.1.3 and Section 11.1.4 respectively. For NELL we perform manual evaluation technique while for REVERB we perform a comparative study with a very closely related work of Liu. et al., [LLZ+13]. While in the next subsequent sub-sections on cluster methodology, we present some of the detailed parameter selection steps and finally in Section 11.3 we compare the cluster based method with the rule based method.

11.1 RESULTS: RULE BASED METHOD

11.1.1 *Relations Statistics*

For the NELL data set, having approximately 2.3 million triples (Table 2), we did not consider the relation *generalizationof* since it is analogous to `rdf:type` which expresses class instantiation. In this paper, we are focusing on the more complex task of finding the correct mapping for a potentially ambiguous relations from NELL instead of generating type assertions like *isA(Albert Einstein, Person)*. How-

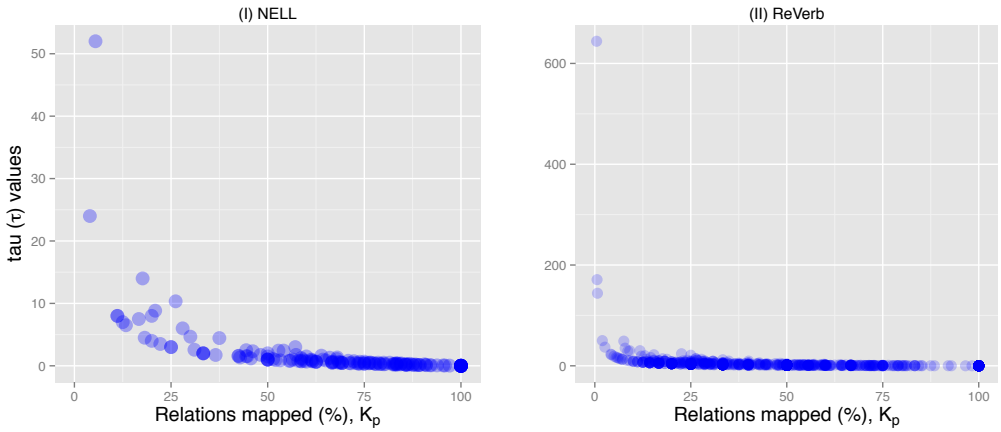

Figure 16: Variation of τ^{p}_{min} with the mapping degree, K_p.

ever, with the REVERB data set there was not necessity to prune off any particular relation having a *isA* semantics.

After the preprocessing steps (discussed in Section 7.1.2) we use the pruned data sets for the experiments. We must reiterate that the quality of the relations is heavily dependent on the instance mappings. Hence, for each of the data sets, we first performed the instance mapping step and used the set of refined instance mappings to perform the relation mapping. As a first step, we wanted to analyze the degree of mappability of each of the relations and their respective value for τ_{min}. The concept of τ_{min} has been introduced in Section 9.1.1 and it defines the likelihood/confidence score for the best association rule for a particular relation. This has been presented in Figure 16 for both NELL and REVERB. This figure captures the best score for τ (lower the better) that a particular relation can have. Every relation p has a mapping degree of K_p (Equation 15), and this degree uniquely quantifies the relation. For instance, the REVERB relation *is a city of* has $K_{\texttt{is a city of}} = 74.19\%$ and the plot directly translates the value of the best (minimum) τ it is allowed to have. The figure above present the values for both NELL and REVERB but these values are for the single relation scenario (wf_1). The figure, especially for REVERB, would change for a clustered scenario. Those figures are presented in the following sections.

We make two major observations in this figure. First, the distribution patterns in the two data sets. Both seem to maintain a linear nature towards higher values of K_p but is non-linear towards the lower ends. The figures presents a power law [FFF99] distribution between K_p and τ. But it is an important to remember

that we are not interested in the entire range of K_p values, but only in the region which is towards the tail. We explain the rational for this in further details in the subsequent section (11.1.2). As we move towards the lower ranges, we see some too high values. This trend is observable in both the two data sets. Higher values of K_p indicates better evidence for building the domain/range restrictions from the relation arguments. For lower values of K_p, the expression for τ (Expression 16) immediately evaluates to a high score. For those low mappable relations (or clusters) to have a lower τ value requires extremely high confident rules. But this is a rare scenario as it seems from the data. Both the sub figures, show some overshot points in the lower K_p regions. It is exciting to observe that the two inherently varied data sources with a lot of structural differences still exhibits a general pattern with our formulation of τ. This regular trend motivates us to adopt an unique scheme and likewise supports our rational for the choice of linear regression for learning a threshold (Section 9.1.2). Second, the figure has been intentionally plotted with shaded points which makes it easy to observe the denser regions in the plots. Here we notice a bit of a difference, since with NELL the density is higher towards the tail end. This indicates that NELL has better set of relations in terms of mappability. However, REVERB displays some sparsity towards the end but a well distributed pattern in the earlier regions. Both the data sets had relations which could be 100% mappable (observe the dark region at $K_p = 100$).

11.1.2 *Regression Analysis*

Figure 16 reveals a general pattern across different data sets. Now, our goal is to find a threshold value which determines if a particular association rule should be accepted or rejected. We use the data patterns observed to our advantage in coming up with a dynamic threshold value. For instance, for some relation or cluster which is extremely well mapped, the rule confidence need not be very high for qualifying the rule. While, a relation with a low mapping degree means it has very few evidence (in terms of domain and range) to learn something concrete. Hence, the association should be strong enough to qualify. This has been made explicit in Section 9.1.2, while explaining the learning mechanism. Hence, there is no fixed value which can be set.

The next question we face is the reason for a linear regression. The patterns in the figure are clearly not linear, hence a non-linear regression curve should have sufficed. But, we would like to recollect that the rule based RM method is based on a evidence based learning approach. The better the evidences we have, the stronger we can deduce about the validity of a rule and eventually a possible relation mapping. If the evidences are weak enough, it is not appropriate to actually deduce anything from them. Hence, it is always better to work with relations with a good K_p. This intuitively means to find relations fulfilling a certain cutoff limit on K_p. Then we can consider only those relations (or clusters) which have K_p over this cutoff value. Now the Figure 16 clearly shows, the more you raise the cutoff value (moving right on the x-axis in the figure), the pattern gets linear. Hence, the reason for choosing a linear modeling scheme. In particular, we are especially interested in determining a cut-off value of K_p which gives us the best (least error) model. In the following, we present a principled way of finding the cut-off. We must make a clear distinction at this point about the two kinds of thresholds we are referring to in this context. First, the threshold for fitting a linear model on the range of values for K_p (essentially 0 to 100%). We refer to this as the cutoff. Second, once the former is known, we can fit a linear regression model to find an actual threshold to decide on the acceptance of a particular association rule (the actual rule based RM step). As the first step, we present the details for the former cutoff determination.

We, altered the cutoff value starting from 0% and ranging till 50%. For each setting, we ran the relation mapping algorithm and obtained the set of data points which comprises the set D (as defined in 9.1.2). To run RM under various cutoffs necessarily means, to ignore all the relations which have K_p less than the set cutoff. This has been depicted in the Figure 17. We present the values for wf_1 for NELL (subfigure (I)), wf_2 and wf_3 for REVERB (subfigures (II) and (III) respectively). The respective cutoffs are stated as percentage values on the top of each figure columns and it is easy to observe the absence of data points below those cutoffs. At a glance, the plots for NELL contain fewer data points which can be seen with the sparseness of the figures. This is natural since, the number of REVERB relations is few magnitudes larger than that of NELL but again in figure 17(III) we observe sparseness due to clusters of relations. The dots in this subfigure are for the cluster mappability, not for the individual relations.

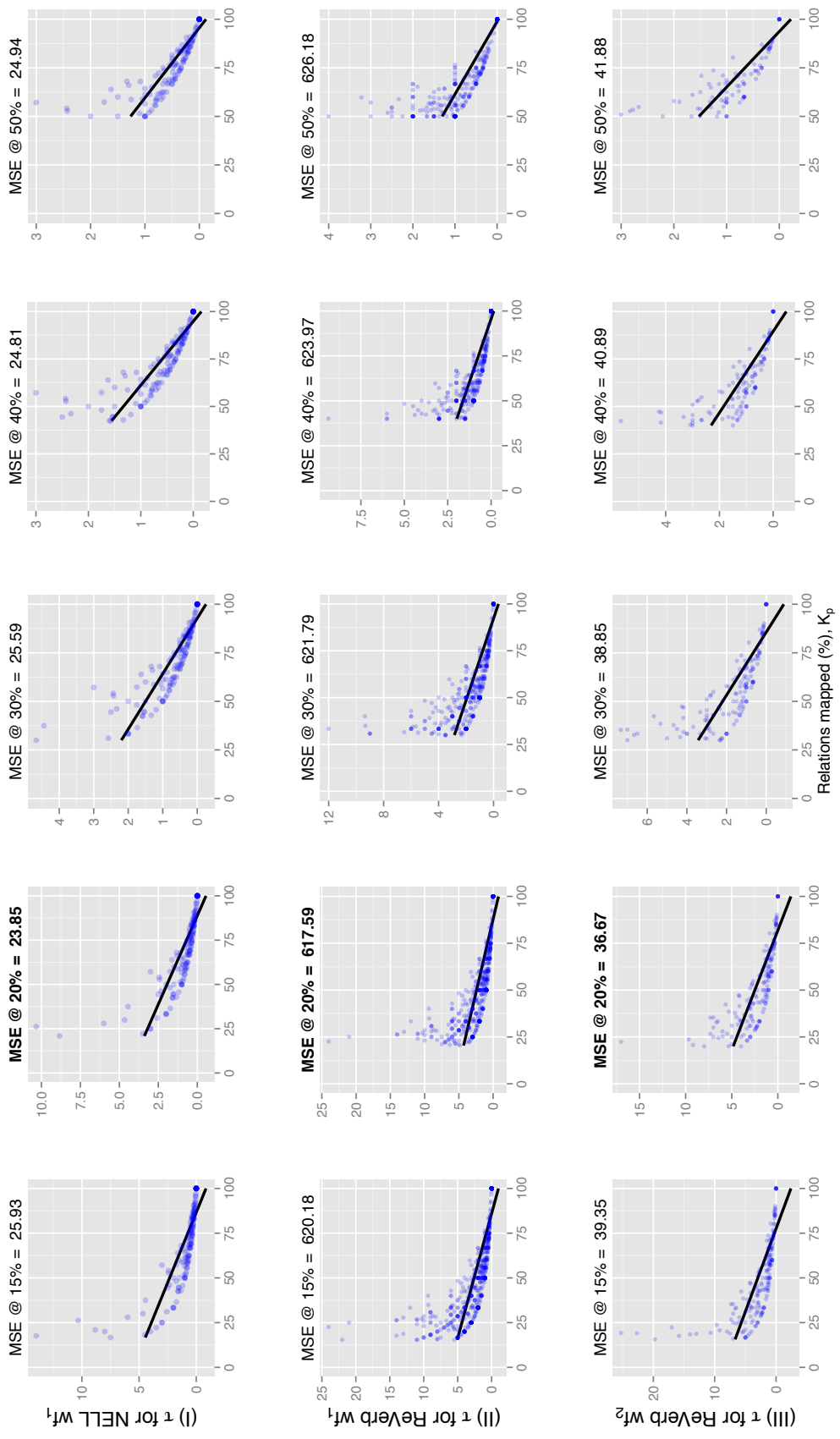

Figure 17: Variation of threshold and regression fit. We measured the error also for higher threshold values but we present only the interesting regions in the given space.

We fit a linear regression line on the set of values we obtain for each cutoff value. This learnt model is then applied on the complete set of K_p vs τ_{min} values (obtained at 0% threshold i.e at $K_p \geqslant 0$). The effectiveness of the learnt model is determined by the Mean Squared Error measure (MSE). The detailed error definition has been presented in Section 9.2. Lower the error, the better model fit it is. Each of the figure thus obtained is titled with its corresponding cutoff value and the mean-squared error for the regression fit. The one which gives the least squared error is considered the optimal threshold value for that particular data set and the workflow. We also notice a variation of errors over the cutoff K_p. A low cutoff induced higher error and so did a higher cutoff. A valley was reached somewhere in the middle. Also with the sub-Figure 17(II), the MSE is higher than the ones in sub-figure (I) or (III). This is due to the higher data points in general for wf_1 wit REVERB which makes the squared error sum larger than the other two setups. For all the 3 setups we observed an optimality at 20% cutoff value of K_p.

Furthermore, the regression line has a negative slope, hence it minimizes the acceptable value of τ even more towards the relations with higher K_p. This is dynamicity in threshold we mentioned earlier. Intuitively, it would require really a very low confident rule to be rejected as appropriate evidence for relations with high mapping degree (follows directly from the formal definition of τ (Equation 16)).

Thus, we explored our input data to empirically decide on the allowable τ value for any association. Referring back once more to Table 11, let us consider the association rule,

$$grew\ up\ in, residence, Person, PopulatedPlace$$

The following steps are followed for deciding on a mapping:

- $K_{grew\ up\ in}$ is computed using the Expression 15
- $conf^2_{grew\ up\ in}$ for the above rule is calculated using the Expression 14
- $\tau^2_{grew\ up\ in}$ is calculated using the Expression 16
- $\hat{\tau}_{grew\ up\ in}$ is calculated by the linear model given by Equation 18
- $\tau^2_{grew\ up\ in}$ and $\hat{\tau}_{grew\ up\ in}$ are compared to make the final decision.

NELL relation	DBPEDIA property	Precision
headquarteredin	`headquarter`	1.0
visualartistartform	`movement`	0.06
	`field`	0.95
personhasresidenceincountry	`nationality`	0.33
airportincity	`city`	1.0
	`location`	1.0
stadiumlocatedincity	`location`	1.0
televisionstationincity	`locationCity`	1.0
	`location`	1.0
televisionstationaffiliatedwith	`broadcastNetwork`	1.0
	`formerBroadcastNetwork`	1.0
radiostationincity	`broadcastArea`	1.0
	`city`	1.0
personhasethnicity	`deathPlace`	0.0
	`birthPlace`	0.60
haswife	`partner`	1.0
	`spouse`	1.0
*musicianinmusicartist**	`bandMember`	1.0
	`associatedMusicalArtist`	1.0
*agentcreated**	`author`	0.80
*citycapitalofcountry**	`largestCity` 0.91	
	`capital`	1.0
*automakerproducesmodel**	`manufacturer`	1.0
Macro-average		0.96

Table 12: Precision of relation mapping task on NELL. N.B. * denotes inverse property mappings [DMS14].

Eventually, if the association is accepted, the mapping from *grew up in* to *residence* is considered correct and thus a relation mapping is achieved. Thus, the final learnt model for a particular setup and data set, is a trend line to determine the best (minimum) possible allowed τ value. Hence, any other unlikely associations would lead to a very high τ value, and would be rejected. This is where the strength of our τ formulation lies.

11.1.3 *Performance with* NELL

In this section, we apply our regression techniques on NELL data set and evaluate the quality of the relation mappings generated by manually annotating them. Three annotators were provided samples of 300 NELL triples each. The annotation scheme here was slightly different from the one adopted with the instance matching evaluation. Apart from marking the property mapping as correct or incorrect, annotators also marked the original NELL triple to be "Correct", "Incorrect" or "Ambiguous". The later annotation was important since, even if the mapping of instances and properties might be correct in the given context, evaluating property mappings for an incorrect triple or an ambiguous NELL triple is not valid. For instance, the following triple: *statelocatedincountry(stateorprovince:nh, country:united_states_of_america)* was marked "Ambiguous", since is was not clear about the reference of "nh" in the triple. And hence, it is hard to accurately judge the correctness of the relation mapping. And likewise the triple: *arteryarisesfromartery(artery:pulmonary_artery, artery:right_pulmonary_artery)* was marked "Incorrect" for obvious reasons. Based on this agreement, only the triples with correct instance and property matches were considered as *true positives*.

In Table 12 we present the precision scores for the property mappings. In this table, we also present some of the inverse property mappings as well. The NELL properties with an asterisk (*) denote the inverse properties. Observe that for some properties, we have multiple choices for DBPEDIA properties, *visualartistartform* for instance. When mapped to `field` the precision of the mappings was better than when mapped to `movement`, even though the later fitted the domain/range restrictions. This is a major difference with the instance matching and its evaluation. Since with IM, there cannot be multiple correct mappings but just one, while with RM multiple correct candidates are possible. Also note the presence of some completely wrong mappings which made the precision values to be 0, for instance with *personhasethnicity*. This table has a different format of representing the mappings, we select few NELL relations and map them to DBPEDIA relations with respective precision scores. This is more inline with the task of mapping relations in general: mapping OIE relation to KB. Overall, we had a precision of **96.0%** for property mappings.

	Instance based (Liu)	**Rule based**
Target KB	Wikipedia (Oct. 2008)	DBpedia (release 2014)
KB facts	≈ . 14Mi	≈ . 883Mi
KB relations	26,458	1079
Reverb facts	407,247	331,131
Reverb relations	65,536 (82,582*)	72,925
Mapped KB relations	509	200
Mapped Reverb relations	2527	1749
Mapped Reverb facts	8969	16,300

Table 13: Comparative values for the relation matching approach as proposed in the works of Liu et al., [LLZ+13] against our proposed method.

11.1.4 *Performance with* Reverb *(system compare)*

As the next valuation setup, we were curious to evaluate our method with the Reverb data set as well. Hence, in this section we present the performance figures for our rule based approach on Reverb. In particular, we make a direct comparison with the work of Liu et al., [LLZ+13] using the Reverb data set. We found a similarity with our relation matching work which makes this comparison justified and our experimental setting sound.

In the related work section, we have provided an overview of the work of Liu et al., hence we skip the details. Here and discuss more about their experimental setup. We replicate their scheme for our evaluation purpose. Liu aimed at mapping Reverb relations to Wikipedia infobox attributes so, they used the smaller Reverb corpus containing only the extractions from Wikipedia. We did the same and used the Wikipedia-Reverb corpus[1] as input for our complete workflow. Since Liu uses an instance based similarity check between the OIE terms and Wikipedia article, we refer to it as "Instance Based" and is shown as such in Table 13. In contrast, we refer to our proposal as the "Rule Based" relation mapping (RM). We try to stick closely to the presentation format by reporting the numbers against the same set of values that were reported in the original work of Liu et al. They had referred to as attributes which we call as relations. The table presents the numbers for the KB, OIE and the final mappings. Although we are using the most recent DBpedia data set compared to the old dump of Wikipedia used by

1 This is a smaller corpus released by Reverb and consists only extracts from Wikipedia.

Liu, it does not make any radical differences with the methodology.

The values we present for REVERB (fact count and relation counts) are after the pre-processing step as we have done with the complete REVERB data set. The pre-processing steps have been already outlined in the experiments section of the instance matching step (Section 7.1.2). We prune OIE facts based on a confidence threshold of 90% as done by Liu et al. in their setup. It must be differentiated here that, the work of Liu had considered both functional and object properties for mapping. In contrast, we focus only on object properties (entity-vs-entity relationships in the KB). Hence, our preprocessing step required us to remove literals, date and time values from the set of input OIE facts. Comparing the respective values in the relevant rows reveals that, we loose 18.69% (407,247 to 331,131) of the REVERB facts due compared to preprocessing. However, we detected an anomaly while counting the relations. Our analysis reported 82,582 REVERB relations instead of 65,536 as presented in the paper. Hence, after pruning, we were left with 72,925 relations. Finally, the mapping related numbers show that, we could match to 200 DBPEDIA relations compared to 509 by Liu. But, we must also consider that 509 includes both functional and object relations. unfortunately, we do not know the exact number of object relations out of these 509 relations to make a neutral decision. But, the number of REVERB relations were comparable in both the approaches. It is an interesting to observe that the number of mapped REVERB facts were nearly double in our case. This occurred due to the a better instance matching scheme we employ instead of page specific entity matches.

These numbers give an impression of the quantitative aspect of the two approaches. We can safely conclude that, even though we have a selective set of target KB relations, they are not varying in large orders of magnitude. Next, we present the qualitative aspect and present the precision scores for the actual mapped relations. For evaluation, we selected a set of 400 random mappings from the set of final relation mappings. We followed the exact evaluation scheme as done by Liu et al. : marking a mapping as correct if the evidence in the REVERB fact supports the mapping to the DBPEDIA relation and incorrect otherwise. For instance, we marked the relation *have plan for* $\rightarrow$ `dbo:birthPlace` as incorrect in support of the REVERB fact *have plan for(evo morales, bolivia)*. Similarly, we found some positive examples in REVERB like *be a film by (manderlay, lars von trier)* which supported us to mark *be a film by* $\rightarrow$ `dbo:director` as correct. In Table 14, we

KB relation	#Mapped Reverb relations	(Rule / Inst.) Precision	Sample Reverb relation phrases
dbo:isPartOf	135	100.0	*be consider part of, be now part of, be a municipality in, be a neighborhood on*
dbo:city	100 (149)	100.0 (99.9)	*be a university locate in, be also headquarter in, be also in, be the fifth largest employer in*
dbo:headquarter	49	100.0	*be a newspaper publish in, be a regional airline base in, be a weekly newspaper in*
dbo:parentCompany	27	100.0	*be currently a subsidiary of, be now a part of, be now a wholly own subsidiary of, be the venture capital arm of*
dbo:birthPlace	151 (69)	94.11 (93.5)	*be an american author from, be an american politician in, be an american singer from*
dbo:hometown	78	92.30	*be a heavy metal band base in, be a musical act from, be a pop band from, be a punk band from*
dbo:location	173 (234)	91.66 (99.7)	*be a school in, be a state park in, be a software company base in*
dbo:team	40	90.90	*have play for, have play with, start his career at, start his career in*
dbo:country	207	80.00	*be a rural district in, be a settlement in, be a village in north, be also a town in*
dbo:associatedBand	38	76.47	*be the guitarist for, be the lead guitarist for, be the lead singer for, be the lead singer of*

Table 14: Precision scores for the top-10 DBpedia properties to which the Reverb relations were mapped to.

present the precision scores obtained for the top-10 most frequently mapped to DBpedia properties. The second column mentions the number of Reverb relations that were mapped to the particular DBpedia relation mentioned in the first column. The respective precision values are given along with few sample Reverb phrases. The adjacent values in some of the cells, denote the corresponding values as obtained by Liu. In the work by Liu, the authors had reported only on a set of top-10 most precise relation mappings which includes both functional and object relations, but we focus on only object relations and it was not possible to get a corresponding value for every relations. Overall, on the sample of 400 tuples, we had an approximate precision of **92.25%**. This is higher than the value of **88.0%** as obtained by Liu et al.

In this broader section 11.1 we analyze the method on two state of the art OIE systems, with particular focus on Reverb where we compare with an approach proposed by Liu et al., [LLZ+13]. Our manual evaluation scheme achieved over

90% precision for both the data sets, thereby creating the strong impression on the effectiveness of the usage of domain/range restrictions as a robust methodology for relation mapping.

11.2 RESULTS: CLUSTER BASED METHOD

In Chapter 10 we introduced the different clustering schemes. Here, we evaluate those and discuss the effects of some of the parameters we had introduced. First, we define the notion of a "good" cluster by presenting an intrinsic clustering quality measurement (Section 11.2.1). Second, this measure is used for our choice of optimal parameters (Section 11.2.2), especially the graph clustering parameter (inflation factor ϕ, introduced in Section 10.2.2) and linear weight aggregation factor (β, introduced in Section 10.2.1).

11.2.1 *Metric*

To define a cluster quality score, which considers two factors, intra-cluster and inter-cluster sparseness. For a set of clusters, $C = \{c_1, \ldots, c_{|C|}\}$, we measure the cluster outputs in terms of a quality measure [RL99, DJ79], denoted by S and defined as,

$$S = \left(\sum_{c_i \in \mathcal{C}} \frac{comp(c_i)}{iso(C)} \right)^{-1}$$

where, $comp(c_i)$ denotes compactness and is defined as

$$comp(c_i) = min(sim(r_i, r_j)); \forall r_i, r_j \in c_i$$

The authors [RL99] referred to this measure as "*Separated Clusters*" since this measure tries to evaluate by maximizing the separation of clusters relative to their compactness. Intuitively, it measures how tightly any two arbitrary phrases r_i and r_j are connected in cluster c_i by looking at the minimum pairwise score between all elements in c_i. Note that $comp(c_i)$ is defined only if a cluster has at least two elements. Otherwise, we set it to zero. The metric $iso(\mathcal{C})$ measures isolation. It is defined as

$$iso(\mathcal{C}) = max(sim(r_i, r_j)); \forall r_i \in c_i; \forall r_j \in c_j; c_i \neq c_j$$

It denotes how sparsely the elements are distributed across clusters in C. Ideally, for a good cluster scheme, every cluster c_i should contain very closely similarly elements i.e high compactness and there should very low similarity between elements across different clusters, i.e. low isolation. This tends to make S low for good clustering schemes.

11.2.2 *Parameter Search*

In our cluster based workflows, we used two parameters: β which is the weighing factor for the pairwise relation similarity scores; ϕ which is the inflation factor for performing the markov cluster. In this section we present a principled way of choosing the optimal values for these parameters. We alter β in steps of 0.1 starting from 0 to 1.0. For each of these settings, we obtain different pairwise scores for our set of relational phrases. For every β we create different similarity files, which serve as inputs to the markov clustering routine. Here, we let the inflation ϕ vary in steps of 1 ranging from 2 to 40. We had tried with ϕ=1, but it did not converge after a finite amount of time.

Furthermore, we chose 40, after observing that cluster size did not vary much beyond $\phi \geqslant 30$. However, choosing a maximum of ϕ=40, was enough to capture the saturation trend. Essentially, we executed the markov cluster routine 11 * 39 times (11 values of β times 39 values of ϕ). Each resulted in a configuration which allocated the elements accordingly. Needless to say, these configurations were different from one another, and for some particular combination of β and ϕ, the configuration would be the best. In order to make that qualitative judgement we employed the metric S as discussed in Section 11.2.1, on each individual cluster configuration. This has been depicted in Figure 18(a). For a given ϕ, we present all the β values along the y-axis. We must remember that lower the value of S for a configuration, better is that cluster formation. The figure shows a valley around $13 \leqslant \phi \leqslant 16$. We fitted a smoothed curve over these data points and it represents the general variation. Detailed analysis in this particular range of ϕ values revealed that the lowest score of S was obtained for ϕ=14, and this is expanded over the β values in Figure 18(b). Once, ϕ was chosen, it was simple to pinpoint the β giving the best cluster. We attained it at β=0.4.

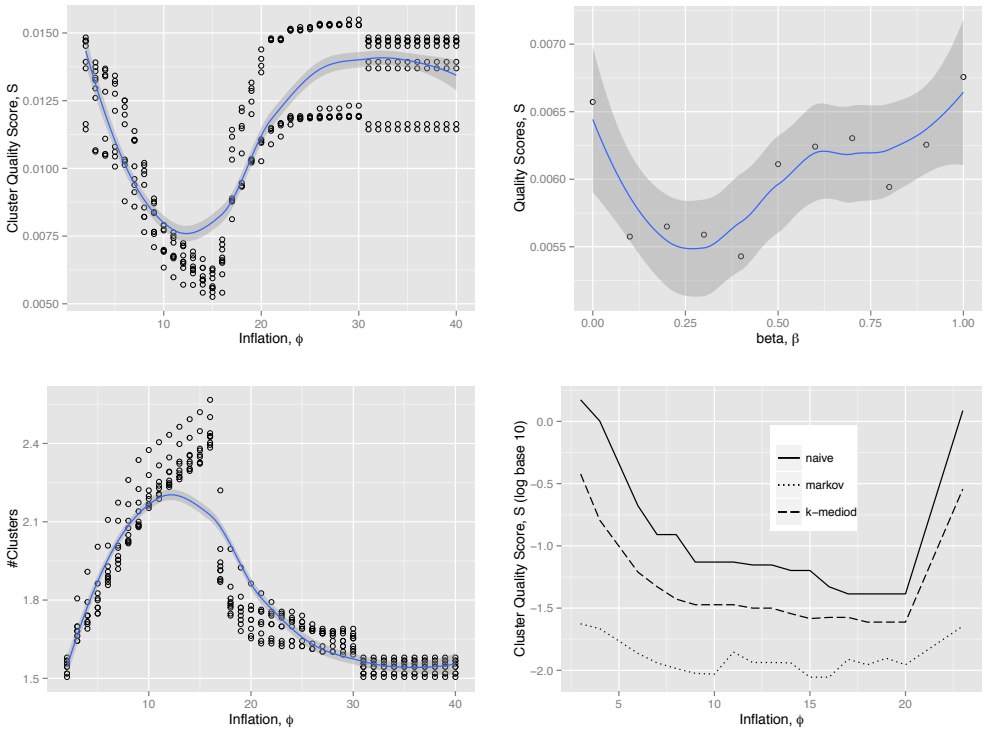

Figure 18: (a) Variation of cluster quality, S with Inflation, ϕ. For a given inflation value, all the corresponding values for β are plotted and a trend line is fitted to capture the overall behavior. Comparison of the Markov clustering based scheme with a naive mediod based scheme. (b) Variation of cluster scores for the minimum beta values for ϕ = 14. (c) Number of clusters depending on ϕ. (d) Comparison of the Markov, k-mediod and a naive clustering scheme with respect to the cluster qualiy scores. [DMS15]

We were also interested to find the change in cluster sizes for the same set of variations. This has been captured in Figure 18(c). Instead of score S, we plot the cluster size (marked as #Clusters) here with the same set of parameter values. It clearly reveals a trend as represented by the smoothed curve and maintains parity with Figure 18(a). A steady improvement phase ($2 \leqslant \phi \leqslant 13$), optimality, then deterioration ($16 \leqslant \phi \leqslant 22$) and eventually saturation ($\phi \geqslant 22$).

11.2.3 *Clustering Techniques*

We compared the performance of the Markov cluster approach with two different variants of clustering. The first one is a naive clustering. It selects k random relational phrases from an input set (in our case, $k < 500$) and tries to assign the rest (i.e. 500-k) of the phrases closest to one of these k feed relational phrases.

However, we have chosen the number of clusters generated by the Markov technique, and feed it as k for the naive clustering. Thus we ensure that the number of clusters generated by the two schemes is comparable. It is a graphical clustering scheme where the pairwise similarity scores were taken into consideration.

The next clustering scheme was the k-medoid which is vector based clustering scheme. The underlying mechanism has been described in Section 10.1. Even with this scheme, we ensured that the number of clusters generated remains the same. Once again this was strictly set to compare each of the schemes with one another. We measure the score S, as introduced above, for capturing the individual qualities of the schemes. The comparative values are presented in Figure 18(d). Since, the naive approach is based on random feeds, we perform a repeated random sampling ($\approx$ .1000 times) for each step of ϕ, and get the best score from them. As discussed in Section 10.2.2, we try to reduce the granularity of clusters by iteratively re-clustering. For this step, we try to find sub-clusters for clusters with more than 10 elements. While doing this, ϕ was kept the same. It often happened that a sub-division was not possible and the iteration was terminated. The cluster sizes and quality scores reported here are the numbers measured after performing this refinement step. We had an average cluster size of 5.204 at the optimal inflation.

Even though we plot "Inflation (ϕ)" in the independent axis, there is no direct influence of the inflation factor on either of the naive scheme or the k-medoid scheme. Inflation affects the cluster sizes, and that affects the cluster scores. The purpose of these preliminary experiments was to measure the effectiveness of the Markov clustering technique compared to other approaches. In the following experiments we adopt markov clustering as our preferred clustering method and run it with different sets of inputs; only on REVERB in wf_2 and on a mixture of REVERB and DBPEDIA in wf_3.

11.3 RESULTS: RULE VS CLUSTER BASED APPROACHES

We observed that the vector based scheme was not so efficient in finding accurate clusters within the OIE relations phrase. We wanted to evaluate the cluster based scheme against the rule based one with the work of Liu et al., [LLZ+13]. In particular, this comparative study specifically analyses three methods of relation

mapping. We are analyzing only the REVERB data set here, since it makes the comparison with the related work easier and direct and also the advanced clustering modes are not applicable to NELL.

- **RL - CL**: Read as *rule minus cluster*, this employs the rule based mapping technique only as described in Chapter 9. This considers only the REVERB relation phrases as units and has no clustering component involved. The clustered relations are then mapped to DBPEDIA relations b exploiting association rules. This is the part of wf_1.
- **RL + CL**: analogous to the former shorthand, this is read as *rule plus cluster*, and this method first clusters the REVERB relations as described in Chapter 10. This results in individual groups of relations. The idea is to map the whole cluster to a KB relation. Hence, this method applies rule based on the clusters. This is a part of wf_2.
- **+Cl**: Simply read as *cluster*, this is without the rule based component and constitutes only clustering. This is a part of the wf_3 mode.

One must observe the markings of +CL and -CL to differentiate the two setups. '+' denotes inclusion of clustering and '-' denotes exclusion. In Section 11.1.4 we presented the details of the comparative evaluation setup. In this scenario, we repeated the exact same steps. The evaluation scheme is in a way simple, since, it considers only the final relation mappings. Hence, it is convenient to directly generate such with different methodologies and make a direct comparison. The values for RL - CL are exactly the same as already shown in Table 13. Furthermore, we do not re-state the values which remain the same across the rule based and the cluster based setups, especially the input data set size.

In particular, we generate additional values for extending Table 13 for the cluster based approaches **+Cl**. We replicate the setup and report the respective numbers in Table 15. For a better snapshot we have also reported the similar values achieved by Liu and as reported in their original work. We observe that, that both quantitatively and qualitatively, the cluster based method combined with the rule based method is unable to achieve the precision values as achieved formerly. We must remember that, the REVERB relations are first clustered into synonymous groups and then the whole cluster was mapped to a KB relation with the rule based method. This approach is essentially not a completely cluster based approach, but a combination with rule based. While the improvement was attained

Attributes	Liu et al.	RL - CL	RL + CL	+CL
Mapped KB relations	509	200	103	355
Mapped REVERB relations	2527	1749	1185	354
Mapped REVERB facts	8969	16,300	14,963	-
RM Precision	88%	92.25%	89.10%	**93.85%**

Table 15: Comparative values for the relation matching approach as proposed in the works of Liu et al., against our proposed method.

with the complete cluster based mode. This is the last column in Table 15. However, this particular experimental setup revealed some interesting aspect. With the RL + CL method the precision values dropped compared to its non clustered version. The cluster based method treats every group as an individual unit, unlike individual relations as with the only rule based method. Hence, any unit failing to meet the threshold requirements of the rule based approach, leads to truncation of the whole cluster unit and thereby few REVERB relations simultaneously. This has its effect both on the REVERB and mapped to DBPEDIA relations. However, the precision score was not severely affected. We could still achieve a better score than the method of Liu, but comparing with our previous non-clustered rule based method, we were poor in performance.

We could push the bar further when we opted for a complete cluster based method. We see that a lot more number of KB relations that are actually mapped to know, although the number of source relations dropped. On detailed analysis, we figured that, often there were some clusters without any KB relation in them. This made all the relations in those clusters practically unmappable. It is actually an interesting observation, that with the RL + CL method the target KB relations gets confined to a smaller set of possibilities. While with the pure clustering scheme (+CL) the candidate KB relations enhance, since, we allow all the object properties to be clustered with the OIE relations. Also note that with +CL method we have no value for "Mapped Reverb facts". Referring to the workflows as illustrated in Figure 3 it is clear that with the wf_3 there is no evidence based learning involved, hence the support OIE facts f^+ are not required. With the former two mapping methods, these facts were actually mapped and contributed for finding the relation mappings. Hence, the value is not applicable in this case.

In Table 16, we present some sample examples from the two methods. It is seen that for the DBPEDIA relations `dbo:occupation` and `dbo:country`, the methods perform at par. While for the third relation `dbo:hometown` we observe some phrases

DBpedia Relation	Rule (-Cl)	Rule (+Cl)
dbo:occupation	*be the founder of be the principal of, found, join on retire from, serve as, work as*	*is chairman of, is Chairman of, is currently looking for, is director of, is Director of, is head of*
dbo:country	*be a rural district in, be a settlement in, be a village in north, be also a town in*	*is a region of, is a town in, is an island of, is grounded in, is located north of, is the president of, located in*
dbo:hometown	*be a heavy metal band base in, be a musical act from, be a pop band from, be a punk band from*	*founded in, included for, is based in, is currently based in, is founded on, is included for, is published in*

Table 16: Example mappings for the two RM schemes.

which should not have been clustered together. For instance, *is included for*. Also, it is to be observed that for **Rule (+Cl)** the sample relations are clustered first and then mapped. While with the **Rule (-Cl)** the sample values are individually mapped first and thus they form a natural grouping.

CONCLUSION

We present two different relation matching techniques: a rule based and a cluster based approach. The former exploits the domain/range restriction of a given relation and uses them as guidelines to find a matching KB relation. We evaluate this on the two state of the art OIE systems and were able to achieve above 90% of precision in both the cases. While the later focusses on distributional semantics of the relational phrases to find logical groups within themselves. These two techniques should not be compared as rivals, since we do not make an explicit claim that one is better than the other. The reason being, the input data set are different. With NELL, relations are normalized and not of the natural language forms. For such cases, we observe rule based method to produce good results, achieving almost 96% relation matching precision. While with REVERB, the relations are more of the natural language form and non-normalized. Clustering seems to be a better choice in that case. However, we focussed only on the precision scores but not on recall values. In the next part we devise a gold standard to evaluate recall and likewise F_1 scores as well.

Part IV

KNOWLEDGE GENERATION

12

INTRODUCTION

This is the final part of the thesis which concerns with knowledge generation. We can also refer to it as KB extension since, with our approach it is possible to extend a KB with additional facts. In this context, we are referring to extending only structured knowledge bases, i.e. the ones which have a well-defined schema and necessarily maintain a concept/relation hierarchy. In particular, we intend to extend DBpedia in our case but the proposed strategy would be appropriate for any other KB as well. In the rest of this chapter, we discuss some of the related works in the area of KB extensions. In Section 12.2 we formally introduce the problem of knowledge base extension in our context. Furthermore, in this part we intend to combine the outputs of each of the individual modules into one coherent workflow. The framework was presented in an abstract fashion in Section 3, but here, we present the final block which makes the framework complete. In this chapter, we primarily focus some of the related works in this direction and formally introduce the problem. In the later chapters, especially in Algorithm 13.1 we present an algorithm for solving this KB extension task. And in in Chapter 13 we introduce an distant supervision based gold standard creation technique, and we evaluate our approaches against the gold standard.

12.1 RELATED WORK

12.1.1 *Knowledge Base Constructions and Debugging*

There has been some development towards scalable knowledge base creation with a minimal amount of human intervention. Chen et al., [YC13] introduced a system called ProbKB, performing deductive reasoning on web extracted facts by a set of first order logic rules and solving as an inference task in MLN. As a follow up work, in [WCG+12], ProbKB was used for automated KB construction. Our work does not target creation but rather using the open information for extending an already existing structured KB. Also, considerable work has explored unsupervised methods for tasks like acquisition of binary relations [Bri98], facts [ECD+04], and

instances [PVD08]. Pujara et al., [PMGC13] have used probabilistic soft logic to detect inconsistencies in knowledge graphs by exploiting dependencies within the graph. These works aim at reasoning within a given structure, provided, for example, by the concept and property hierarchy of NELL. In our work, the instance mapping module exploits OIE reasoning to a considerable degree in the context of a target KB like DBpedia and Yago to disambiguate OIE terms. Our assumption is that the source facts might be unstructured and do not maintain a concept/role hierarchy. This makes our approach independent of the existence of a source ontology. Moreover, we do not aim at refining the OIE itself, but use the OIE data in its given form to generate semantified facts.

12.1.2 *Distant Supervision Based Approaches*

On a different note, there has been a lot of work on distant supervision based approaches since the early 90s. Work like DIPRE [Bri98] was first of its kind to use a set of seed KB facts to discover patterns all across the web. Those patterns were used to discover more facts, furthermore, bootstrap these two to learn more facts and more patterns. The idea was also seen in systems like SOFIE [SSW09], where natural language texts were excavated for entities and relationship patterns and most likely entity references were solved as a joined satisfiability problem. In particular, systems like PATTY [NWS12] provide a taxonomy of relations. This is yet another example of a system which exploits relational patterns between entities and uses them to create a hierarchy of relational phrases. The authors of PATTY tried to paraphrase DBpedia and Yago entity relations with multiple paraphrases. This is different, since our approach tries to find a mapping given a set of paraphrases. Even Nell [CBK+10] and Reverb [FSE11, EFC+11] bear a similar architecture in identifying and finding relationships across the web. More recently, there was clustering based work by Moro et al., [MN13], Sun et al., [SG10]. This genre of work primarily focuses on the relations from open domain extractions; it extracts them, clusters them and finally disambiguates them. They exploit distributional semantics of relational phrases to aid their clustering (which is based on shortest path kernel method). Eventually, their goal is to disambiguate OIE facts based on the context, for instance "*is part of*" may be used in the sense of a location part of a larger place, or a person part of a band or organization. However, we have a different objective. We want to fully semantify an OIE triple in terms of a target KB, i.e., we want to select the correct property from the KB given an

ambiguous relational phrase (and we have to solve a similar mapping task for subject and object terms).

There are also some interesting works from the industry research groups, of which the works by Gattani et al., [GLG+13] is especially interesting. The developed a system to identify and disamabiguate entity mentions from micro blog streams (tweets) and eventually tried to use Wikipedia as the background knowledge base to classify the tweets into topics. They exploit two major sources explicit information, the context of the micro blogs. They show this is helpful but necessarily not always, since often some mentions are too short to actually provide some context, for instance "*Go! Giants!*", which actually refers to the New York Giants Football team. They proposed to exploit social signals in the form of related tweets in the past time frame to perform efficient disambiguation. They have empirically showed that such aggregation helps to improve entity linking and majorly topic classification as a whole. The problem setting is different than ours, but still we mentioned it as we also perform linking under limited context. The idea of incorporating social signals is unique but unfortunately we lack such possibilities.

12.2 PROBLEM STATEMENT

We primarily distinguish between two types of knowledge bases: the unstructured one, $\mathcal{U}$ and the structured one, $\mathcal{O}$. The former consists of triples or facts from open information extractions and are in the form of $p(s, o)$, where p is an OIE predicate or relations, and s, o are the subject, object terms to the relation respectively. Formally, the entire KB can be considered as a set of 3-tuples or triples as,

$$\mathcal{U} = \{ (s, p, o) \mid s, o \in \mathcal{S}, \quad p \in \mathcal{S}^R \} \tag{24}$$

We must recollect from Section 4.2 that $\mathcal{S}$ defines the set of OIE argument terms, and from Section 8.2 that $\mathcal{S}^R$ denotes the set of OIE relations. In a similar fashion, we can define the structured KB as,

$$\mathcal{O} = \{ (s_t, r_t, o_t) \mid s_t, o_t \in \mathcal{T}, \quad r_t \in \mathcal{T}^R \} \tag{25}$$

where, s_t, o_t are the DBpedia subject and object terms involved in a relation r_t. We have maintained the similar notations for the target KB instances and relations

as has been used in our earlier description of instance and relation matching.

The idea of knowledge generation is broadly formulated as an integration task between two separate KBs. We use the triples from $\mathcal{U}$ to augment $\mathcal{O}$. This makes sense only when we do not add some preexisting facts to the target KB. In order to ensure this, we had a simple look up module incorporated in our workflow which splits the input structure into two fact sets: f^+ and f^- (details in Section 3.1). Hence,

$$\mathcal{U} = f^+ \cup f^- \tag{26}$$

The instance matching set $\mathcal{M}$ (Expression 1) and a relation mapping set $\mathcal{M}^R$ Expression 10 gives us the term and relation mappings respectively. Hence, for a fact in f^-, we can individually apply the instance mappings and relation mappings by looking for the associated pair values from these sets. Hence, the knowledge generation step from f^- can be formulated as a generation of a new set of tuples with 6 elements, the original OIE triple and its transformed triple expressed in terms of the KB vocabulary. Formally, we represent the set as $\mathcal{N}$, and defined as,

$$\begin{aligned} \mathcal{N} = \{(x, r, y, x_t, r_t, y_t) \mid (x, r, y) \in f^-, \\ (x, x_t), (y, y_t) \in \mathcal{M}, \\ (r, r_t) \in \mathcal{M}^R, \\ (x_t, r_t, y_t) \notin \mathcal{O}\} \end{aligned} \tag{27}$$

Thus, we have a 6-tuple with the OIE terms mapped to the KB instances and relation mapped to the KB relation. It is important that the mappings (x, x_t), (y, y_t) necessarily belong to the instance mapping set and so does the relation mapping belong to the set $\mathcal{M}^R$. These restrictions ensure that the KB instances/relations for the newly generated triples are existing in the KB but not as this assertion. It is also guaranteed that if either of the mappings of the instances or relations are missing, it does not generate a new triple. This follows directly from Expression 27. In theory, we can have exactly $|f^-|$ number of new triples generated. This is based on the assumption that, each and every triple from f^- is essentially transformed and thereby leading to a new fact in the KB.

13

EXPERIMENTS

This chapter particularly presents an algorithm for achieving the task of knowledge generation from OIE inputs. In Section 13.2 we describe the used datasets and the hardware configurations we employed. In the subsequent Section 13.3 we present a detailed description of our gold standard creation with focus on the distant supervision based approach for creating one. In the final Section 13.4 we compare each of the workflows against the gold standard and provide in depth analysis of the results. It is to be noted that we do not present any dedicated methodology section for the knowledge generation step. This final piece of the pipeline is focussed on integrating the modules we have presented so far. In the introductory sections, (Section 3.1) we have presented the working principle of the knowledge generation task with a running example. Hence, in this section, we focus on the algorithm and on empirical results.

13.1 ALGORITHM

We combine all our components, and present an abstract algorithm for the complete framework in Algorithm 5. This can be considered as a generalized architecture for extending a structured KB with triples from unstructured source. The algorithm presented is very general and can easily work with triples from different extraction sources. It accepts as input the different workflow modes and performs the mapping task accordingly. The algorithm we present is modular and consists of the contributions from each of the prior modules, especially the instance matching and relation matching modules. Especially, we observe the calls made to the IM, RM and LM modules. Furthermore we have briefly presented the cluster algorithm and the final integration routine which takes as input the instance mappings, the relation mappings and the f^- triples. These three inputs are also seen as the three incoming edges in the workflow illustration of Figure 3.

The algorithm makes a distinct difference between the different workflows, and they have been marked at each logical points in the algorithm. In workflow wf_1,

we treat each OIE relational phrase individually and map it to a DBpedia property. This involves a direct application of the rule based approach described in Chapter 9. In the experimental sections, we present our empirical results for wf_1 on Reverb. The second workflow wf_2, is an extension of the former, but involves clustering (Algorithm 5, line 14) the Reverb relational phrases. We treat clusters of relational phrases as an unit and apply the rule based technique on them. The intention for this approach is to leverage the collective evidence from all the OIE relation instances belonging to a cluster. In workflow wf_3, we opt for a purely cluster based approach in which the mapping task is automatically solved by computing the clusters. Here, we *add* DBpedia properties as seed and allow them to be clustered with the Reverb phrases using the markov clustering technique. Pairwise similarity scores (Algorithm 5, line 21) between the newly fed DBpedia properties and the Reverb relational phrases need to be computed in that setting. Applying clustering on this heterogeneous mixture of Reverb phrases and DBpedia properties results in clusters where most of the clusters had a DBpedia property along with a set of other Reverb properties. For instance, {`dbo:origin`, "*originated in*", "*is the source of*", "*is a source of*"} is one of the clusters. However, the major drawback of this approach is that, we cannot ensure that every final cluster contains a DBpedia property. There can be clusters without any DBpedia property, for instance *{"is ideal for", "is perfect for"}*. This might be the effect of an unsuccessful clustering or, DBpedia might not have an analogous property capturing the similar sense of a cluster of Reverb phrases.

13.2 EXPERIMENTAL SETTINGS

We briefly restate the settings and the data set we used in this experimental setup. We used the clueweb Reverb data set having approximately 15 million facts annotated with a confidence score. In Section 7.1.2 we have detailed the preprocessing step we perform on the data set. And also some of the data statistics have been presented for Reverb in Table 2. The filtered dataset contains 3.5 million triples with 474325 different relational phrases. As target knowledge base we used DBpedia (Version 3.9). With respect to wf_3, we used all of the object properties of DBpedia as input to the clustering. All experiments were conducted on a 64-bit Linux machine with 4GB physical memory and 2 cores. however, for the similarity computation module, we implemented it as an asynchronous and multi-threaded application, allowing us to exploit the multiple cores of a 8-core

Algorithm 5 Algorithm for Structured KB Extension with open extractions

Require: F: facts from OIE system; mode

```
1: function genFacts
2:     i_maps ← null                                  ▷ instance mappings collection
3:     r_maps ← null                                  ▷ relation mappings collection
4:     phrases ← relations from F
5:     cl ← cluster(phrases)                          ▷ call to line 14
6:     i_maps ← IM(cl)                                ▷ call to IM module
7:     f⁻, f⁺ using LU(i_maps)                        ▷ call to LU module
8:     if mode = wf1 or wf2 then
9:         r_maps ← mapRelations(f⁺)                  ▷ call to RM module
10:    else
11:        r_maps ← from cl
12:    newFacts ← integrate(i_maps, r_maps, f⁻)       ▷ call to line 24
13:    return newFacts

14: function integrate(i_maps, r_maps, f⁻)
15:    newFact ← null
16:    for sub, rel, obj in f⁻ do
17:        sub_KB ← i_maps.get(sub)
18:        obj_KB ← i_maps.get(obj)
19:        rel_KB ← r_maps.get(rel)
20:        if sub_KB or obj_KB or rel_KB ≠ null then
21:            newFact ← rel_KB(sub_KB, obj_KB)
           return newFact
```

machine.

13.3 GOLD STANDARD

In this section, we discuss in details the gold standard creation. In our previous sections, we had evaluated the instance and relation mappings manually, but we must make it clear the necessity of a gold standard. It can be attributed to broad reasons:

1. **Limitation with RM**: The broad idea is to find a correct mapping of the full OIE triple to a target KB assertion, DBpedia in our case. A simple approach would have been to present a sample set of OIE triples to the annotators, and ask them to map each one of the subject, object and relation to a KB vocabulary. This would readily form our gold standard consisting of complete OIE triple mappings. A list of top-k candidates for the subject and

object could also be provided to the annotators to ease them decide the correct ones. But the problem we faced was with the relation mapping. It was nearly impossible for annotators to fetch from the KB an appropriate relation mapping (essentially from the whole list of DBpedia owl properties). Reason being, unlike the terms, a direct surface form search of OIE relations would not link to a set of likely KB properties. Also, we must note that, the final gold standard should always contain unique, unambiguous references for the terms, but for relations there can be multiple ones. Often a set of generalized KB properties which can be considered equally true.

2. **Recall Value**: In all our previous evaluation steps, we could not evaluate the recall values that is the fraction of correct mappings retrieved. This was specially the case with relation matching for both Nell and Reverb data sets. However, with instance matching evaluation we had a gold standard created (as discussed in Section 7.2). But in the integration step as a whole, it was important to measure the recall for the relation mapping tasks.

Hence, we designed a semi-automated approach based on distant supervision, to automatically find the possible set of relation mappings. We present this in details in Section 13.3.1. And as a second step, we annotate the instances. We provide details in Section 13.3.2, explaining how the overall evaluation was performed.

13.3.1 *Seeder-Leecher Architecture*

As the name indicates, this consists of two interconnected components: a seeder and a leecher. As the first step, we *seed* for KB assertions. This module randomly selects an OWL object property instance from the KB. A KB assertion involving an object property is called a relation instance. For instance, `dbo:locationCountry` (`db:Canterbury_of_New_Zealand`, `db:New_Zealand`) is a relation instance. Henceforth, we refer to such instances as seed facts or simply *seeds*. Given such a seed, we intend to find out an analogous OIE fact, which best captures the essence of the seed. The core idea is that, if there is KB assertion of the form $p_{kb}(sub_{kb}, obj_{kb})$ and there is an analogous OIE fact of the form $p_{oie}(sub_{oie}, obj_{oie})$, where sub_{oie} is a surface form of sub_{kb} and obj_{oie} is a surface form of obj_{kb}, then p_{oie} is very likely to express the same semantic relationship as conveyed by p_{kb}. This methodology has been in use for sometime and observed in the recent works of Augenstein et al., [AMC14]. It might be exactly

same, general or even specific representation. In this design, we are actually starting from a KB seed and trying to find the analogous relation in the OIE side. This is in contrast to the rule based method where we traverse the opposite direction, from OIE to structured KB.

For each seed, we look up for the top-k surface form representations for the both subject and object terms in the seed. For these k x k pairs, we try to find any matching OIE triple that may exist. For fast lookup, we created a Lucene index on the REVERB data set with the subject and object terms as the search-able fields. This made it possible to scan the entire OIE data set at most k x k times for each seed, in a very time efficient manner. Any successful hit, was stored. For the current example, we recorded the OIE triple "*is located in (Canterbury, new zealand)*", where (*Canterbury, new zealand*) was one of the k x k possible surface form pairs. This mechanism is similar to the one observed in RM module but in the former section, we relied on the OIE facts as basis for searching a KB assertion which is likely to be more uncertain. Here we do the inverse and search for an OIE triple similar to a seed fact. The later suits well for a gold standard creation, since it is less error prone on grounds of precise and unambiguous KB seeds.

The choice of k was critical. Initially, we set it to 5, however, we had extremely low hits. The reason being, the surface form pairs were two strict to find a match on the OIE data set. We increased it to the other extreme by setting it to 30. The problem we had was the matches were two general, and that made the relations often too drifted away. For instance the triple `dbo:spokenIn` (`db:Hawaiian_language`, `db:Hawaiian_Islands`), gave us a match for the REVERB triple *"was born in(Kai, Hawaii)*", since "*Kai*" is one of the surface forms of `db:Hawaiian_language`. Hence we chose, k to be 10. This choice is based on heuristics and not experimentally verified. Moreover, we were not too meticulous about the exact k, because a value bit higher than 10 would still generate matches but maybe more false negatives. So there was no strict definition of optimal k.

As a subsequent step, every seed was written out to a local file with all the possible pairs. For k = 10, the number of pairs should have been 100, but in practice it was often fewer than that since each of the candidate list was not always 10 but lesser than that. We refer to this as one *snapshot*. The leecher is designed in the form of a file listener for this snapshot, so that each time a new snapshot was

written out, a trigger was fired for performing a lookup of Lucene indexes. Every successful match is stored in a database for further analysis. For the next seed, the snapshot was again re-written and not appended to the file.

This module was designed in a very lightweight fashion. Multiple asynchronous worker threads look for seeds from the DBpedia endpoint. Hence, every snapshot always created a small local file with less than 100 lines. Hence, it was less CPU intensive, fast and had extremely low memory footprint (only few kilobytes of the snapshot file). We ran this module for weeks and collected over a 1000 OIE triples along with its analogous KB seed. These were presented to an annotator, and were marked as "Correct", "General", "Subsumption" and "Incorrect".

13.3.2 *Annotation and Metrics*

In Chapter 3 we introduced f^{-}, the portion of input OIE triples which had no analogous KB assertions. These are the OIE triples which can generate new KB assertions and thus new knowledge. We perform a random sample of 1000 triples from this set of f^{-} triples. Each of the subject and object terms are mapped to top-5 DBpedia entities and provided to the annotators. The task was to select the best candidate subject and object from the individual candidate lists. If none of the given options were appropriate, the mapping was marked as "?". Annotators were not required to browse the web for finding the correct mapping since our system considers only the top-5 candidates. If the correct candidate is not within the top-5 list, it is incorrect to penalize the system for producing an incorrect match.

This gives us a set of 1000 Reverb triples, with each subject and object term mapped to a DBpedia entity or a "?". And the Reverb relation is matched to the set of DBpedia properties as annotated in the previous section. The rationale is, if it was learnt that p_{oie} is property representation of p_{kb} then the we can fill in every other occurrences of p_{oie} with p_{kb} for the f^{-} triples. Thus, we generate full mappings of the OIE triples. For the evaluation, we resort to precision, recall and F_1 scores for both instance and relation mappings.

The task is to evaluate every mapping generated by our method (wf_1, wf_2 or wf_3), both instance and relation mappings. Let A refer to the mappings generated by the method, and G refer to mappings in the gold standard, then precision, recall and F-measure are defined as

$$\begin{aligned} prec &= \frac{|A \cap G|}{|A|} \\ rec &= \frac{|A \cap G|}{|G|} \\ F_1 &= \frac{2 * prec * rec}{prec + rec} \end{aligned}$$

In case of instance mappings, each time the gold standard and the algorithm produce mapping pairs, we scrutinize them based on the cases defined in Table 17. Remember that a mapping is a pair consisting of an ReVerb term mapped to a DBpedia instance in the context of the original ReVerb triple. Hence, in one context *Apollo* → `db:Apollo` may be a correct mapping while in some other not. In the referred table, Case I, II and III are the most obvious and self-explanatory cases allowing easy judgments on the mapped values. While in Case IV, if the algorithm produces an output but in the gold standard, it was impossible for a human to annotate, we consider it a false positive and hence incorrect. And the final Case V is where both the gold standard and algorithm fails to produce any mapping, and we completely disregarded this case. The rational being, we do not consider a non-mapping to be a valid mapping since we cannot make a decision if it is correct or incorrect and hence irrelevant for evaluation. In a way we adopt a strict evaluation, a more relaxed approach would have been to ignore Case IV as well.

For the property mappings, the only difference we have is that every OIE relation is mapped to a collection of possible KB relations and not just one. Hence, we can measure the metrics for each individual property mappings and then use them to build a global measure over the complete set of mapped OIE relations. We use micro-average precision, recall which measures the number of correct matches for each relation mapping.

Example 7. *Suppose, we have two OIE relations* p_1 *and* p_2 *in our gold standard, with the following mappings;* $p_1 \rightarrow (a, b, c)$*;* $p_2 \rightarrow (d, e)$*. The algorithm produces the property*

Cases	G annotation	A annotation	Decision
I	$x \rightarrow X$	$x \rightarrow X$	Correct
II	$x \rightarrow X$	$x \rightarrow X'$	Incorrect
III	$x \rightarrow X$	$x \rightarrow ?$	Incorrect
IV	$x \rightarrow ?$	$x \rightarrow X'$	Incorrect
V	$x \rightarrow ?$	$x \rightarrow ?$	Ignore

Table 17: Cases for evaluating the IM module against the gold standard. [x = OIE term; X, X′ = DBpedia instance; ? = reference unknown]

mappings as $p_1 \rightarrow (a)$; $p_2 \rightarrow (d, f)$. *We work out the precision and recall for this simple example*[1].

$$\text{prec for } p_1 = \frac{|\{a\} \cap \{a, b, c\}|}{|\{a\}|} = 100\%$$
$$\text{rec for } p_1 = \frac{|\{a\} \cap \{a, b, c\}|}{|\{a, b, c\}|} = 33.33\%$$
$$\text{prec for } p_2 = \frac{|\{d, f\} \cap \{d, e\}|}{|\{d, f\}|} = 50\%$$
$$\text{rec for } p_2 = \frac{|\{d, f\} \cap \{d, e\}|}{|\{d, e\}|} = 50\%$$
$$\text{prec} = \frac{|\{a\} \cap \{a, b, c\}| + |\{d, f\} \cap \{d, e\}|}{|\{a\}| + |\{d, f\}|} = 66.66\%$$
$$\text{rec} = \frac{|\{a\} \cap \{a, b, c\}| + |\{d, f\} \cap \{d, e\}|}{|\{a, b, c\}| + |\{d, e\}|} = 40\%$$

Also note the use of micro average precision/recall here, using macro average, we would have a precision of 75% ($\frac{100+50}{2}$*). The later is more suited for scenarios with more or less equal data sizes, unfortunately we do not have so.*

13.4 RESULTS

In this section, we provide detailed numbers of each of our experimental settings. Table 18 presents the comprehensive results for all the workflows we have mentioned.

1 {•} represents a set. And |{•}| denotes the cardinality of the set

	REVERB		
	wf_1	wf_2	wf_3
IM Precision	83.85%	85.79%	**90.31**%
IM Recall	22.48%	22.77%	**32.54**%
IM F_1	35.45%	35.98	**47.84**%
PM Precision	35.37%	35.43%	**43.02**%
PM Recall	23.96%	24.59%	**42.01**%
PM F_1	28.57%	29.03%	**42.51**%
matched OIE relations	497	**894**	354
target KB properties	131	84	**355**
new facts generated	96,049	**99,363**	97,267

Table 18: Comparison of workflows. We categorically report on the instance and property matchings. The values marked in bold denote the best value for that category.

1. wf_1: considering every relational phrase uniquely, i.e. without any clustering, and applying association rule mining techniques for property mapping
2. wf_2: clustering relational phrases and applying rule mining for mapping the resulting clusters
3. wf_3: clustering relational phrases with DBPEDIA object properties as seeds without applying any rule mining.

Furthermore, we tried to compute the precision, recall and F-measure for each run. We compare these different workflows based on the following aspects:

1. *IM Precision/Recall/*F_1: for each completely translated OIE fact, this determines the correctness of subject and object mappings.
2. *PM Precision/Recall/*F_1: for each completely translated OIE fact, this determines the correctness of relation mappings.
3. *matched OIE phrases*: the number of unique OIE relations that have been mapped
4. *target KB properties*: unique target KB properties to which relational phrases have been mapped, in this case DBPEDIA object properties.
5. *new facts generated*: number of new KB assertions generated

As the first step, we ran the full workflow under the simplistic scenario, i.e. wf_1. This is the trivial setting where the rule based mapping technique was applied on the REVERB data. Note that, in this setting, every relational phrase was reasoned and dealt with individually, completely disregarding the effect of its synonymous sibling phrases. Furthermore, we could match around 497 distinct REVERB relations. These relational phrases have been mapped to 131 different DBPEDIA object properties. And during the annotation process, it was found that one or more options were wrong. For instance, the phrase "*is a city in*" was mapped to the following DBPEDIA properties: `dbo:councilArea`, `dbo:district`, `dbo:isPartOf`, `dbo:leaderName`, `dbo:timeZone` and more. It is clear that `dbo:leaderName` and `dbo:timeZone` are wrong, and it made the whole translated KB assertion incorrect.

The next improvement, was to jointly deal the relational phrases. We clustered the REVERB relational phrases and performed the instance mapping and property mapping using the rule mining technique. With this workflow (wf_2), we expected that the clustering would push the instance matching precision higher due to the feedback loop we incorporated. Our expectation was also to observe a positive impact on the property mapping phase. Results were interesting in this setting, since, we achieved a better instance mapping and marginal improvement with the property mapping. This speaks in favor of the clustering approach. We must note that, in both these workflows wf_1 and wf_2, we computed the precision, recall and F_1 values against the same gold standard and respective outputs. Hence, the marginal improvements can be attributed to the positive effect of clustering instead of chance factor. We further analyzed the complete set of output mappings for the same relation phrase as we did in wf_1, i.e. "*is a city in*". It had no incorrect mappings to `dbo:leaderName` or `dbo:timeZone`, instead had more valid mappings generated like: `dbo:ceremonialCounty`, `dbo:principalArea` and so on. This is also reflected in the numbers: the number of matched KB properties, reduced from 131 to 84 which explains the fact that multiple relations in the clusters had a more stronger effect in deciding a likely KB property match than the individual relations as in wf_1. Clustering looked promising, but still we waned to compare it with the wf_3 and see the effect of jointly clustering the REVERB relations along with DBPEDIA properties.

Finally, we choose the Markov clustering approach but with DBPEDIA properties as cluster seeds (workflow wf_3). We also re-run the instance mapping mod-

ule using a feedback loop. We observe an increase in instance mapping precision compared to wf_2. We also observe that the property mapping is much better compared to all previous workflows. The improved instance mapping precision combined with a better property mapping precision leads to higher number of new facts which are of the same quality as that achieved with NELL. Adding DBPEDIA properties as seed also helped to achieve better instance mappings. We report the recall and F_1 scores in each of the workflows and clearly, wf_3 excels each of the other two. We achieved considerable gain in precision, and recall values values leading to an overall better F_1.

Analyzing further, the phrase "*is a city in*" was matched to `dbo:city`, `dbo:county`, `dbo:region`, `dbo:state`. Even more interesting was to observe that `dbo:city` was actually a mapping for *is a city in*, *is a city of*, *is a suburb of*, *is a region in*, *is a town in*, *is the capital city of* and also *is the county seat of*. A large cluster of relational phrases was mapped correctly to properties in the target KB. Mathematically, every DBPEDIA property is mapped to approximately almost 1 (=354/355) relational phrases. The mappings for the property `dbo:city` is thus a typical example. Similarly, for `dbo:location`, there were 11 REVERB phases mapping to it, including *is located in*, *is situated in* and *lies in*.

Even though with wf_3 we achieved better recall, we observed that there is often a limitation on the number of phrases which could be mapped. We had clusters of phrases, which could not be mapped or actually had no analogous DBPEDIA property to be mapped to. Examples are phrases like {*is an essential part of*, *is essential for*, *is essential to*, *is the essence of*, *is vital to*, *are essential for*, *are essential to*}, which were in the same cluster, but had no KB property to be mapped to. In this case our approach is right in not generating any mappings for this cluster, because there exists just no counterpart for this cluster in DBPEDIA. However, the clustering itself worked fine, since these relational phrases have obviously a very similar meaning.

Our results indicate that the clustering approach with DBPEDIA properties as input seed is better suited for mapping REVERB facts and produces large number of precise facts. Whereas, the rule mining approach seemed to perform equally good for REVERB but at the cost of a lower recall. This highlights the importance of applying an appropriate clustering technique within an overall framework that

deals with OIE systems like REVERB or similar ones having raw non-normalized textual relational phrases.

As a final note, it is important to mention that the results we obtained with REVERB in the relation matching step were had higher precision scores, this was because, the annotation was carried out on a sample of mappings and every mapping was decided as correct or incorrect. But with this gold standard evaluation, we do not have the myopic case of individually judging if a relation mapping is correct. Rather, we use the seeder-leecher module generated set of *gold* mappings and compare with what our method produces. This is yet another way of evaluation and is specially helpful in getting an impression on the recall scores.

Part V

CONCLUSION

14

SUMMARY

By choosing to work with open information triples instead of on full text as input, our proposed approach separates itself from the usual text-based challenges including that of named entity recognition, pattern extractions or document selection. This work thus locates itself at the confluence of Ontology Mapping, Word Sense Disambiguation, Information Extraction, and Reasoning. We propose a framework for extending a given KB with additional facts. The source of those facts is open extraction systems which employ unsupervised learning models to extract domain independent binary relations and relation instances. The fine grained goal of knowledge generation has been defended in this work as a mode for achieving the broader objective of integrating heterogenous sources. We can envision our solution as a two-layered approach which provides very low level solutions and subsequently abstracts those to achieve the bigger target. In particular, the task of KB extension can be imagined as an intersection of multiple related research areas including data integration and semantification of ambiguous triples.

Our proposed framework is an architectural pattern to achieve the above mentioned objective. The most interesting aspect of it is its modular nature. This has a two fold advantage: first, it makes the framework very generic. One can employ any other instance matching technique of choice and still the framework continues to be a valid. Similar reasoning holds for any other modules. Second, it makes the design very adaptive to different format variations of the input data sources. We observed this closely with our experiments with the two state of the art systems: Nell and Reverb. Both are open domain extraction systems, yet both exhibit a lot of inherent structural differences. Our framework was gracefully able to handle such variations with minor alterations in the configuration settings.

The idea of KB extension crops from the domain independent coverage of the OIE systems. It is a very natural extension to the idea of triple semantification. If we can correctly identify the latent semantics of an OIE triple, then we can, in

theory, generate an exact semantified copy of the entire OIE input data set. Unfortunately, this is never the case, since there are usually a set of triples already pre-existing in the KB. Hence, only those triples which have no analogous assertion in the target KB can potentially towards an extension for the KB. This task is an abstraction and we show in pure empirical results that OIE provides sufficient rational for believing that it can extend a structured KB.

The finer grained approach includes solving a bunch of sub-problems. Ambiguity within the source triples is one of the major in the list. We devise a probabilistic approach to find references of the OIE facts to a KB vocabulary. We make no prior assumptions and use no context information in our instance matching task. Using domain and range restrictions we solve the problem after encoding it as a MAP inference problem in a graphical model. Evaluation shows over 85% of F-measure for both NELL and REVERB. The complementary problem to the instance matching is the relation matching. And we have reasoned that these two solutions are not independent of each other but rather closely coupled. We maintain this coupling in our implementation and use one to improve the other. For relation mapping we propose a type guided approach and formulate a measure for likelihood. We compare the approach against a very similar related work. Our type based approach beats their approach and achieves better precision scores under identical evaluation conditions. In this context, we propose a cluster based relation mapping task as well, which is seen to perform better than the rule based one. But, we deny to claim that cluster based method is always better. It depends on the input data. This is yet another feature of the framework: the ability to choose a method option.

We use this concluding section to discuss some of the future prospects for this work. We mention few major areas which we consider interesting to explore

- We completely disregarded the idea of using contexts in our overall approach, but it will be rational to explore the context driven approaches. We currently suffer from a low number of new facts generated, but it is still not known if better contexts would make only the instance matching better or can generate more number of facts as well.
- We proposed a pipeline architecture to solve a set of sub problems. It will be an interesting effort to design a joint model for the complete task. Our

estimation is that such a modeling will handle the instance and relation mapping tasks together and not as two separate components. This can be a good extension of the the prior task we mentioned, since to design such a joint model, we require contextual information. We modeled the relation-instance dependency with the signal from domain and range concepts, but it can be further enhanced with better signals from the source/sentence in which the extractions occurred. This idea is similar to that of SOFIE [SSW09], which solves a set of hypotheses as a weighted satisfiability problem. SOFIE uses the source documents and patterns from those as an inputs but both might not be always available for OIE. Hence, to jointly model, the real challenge is to hypothesize on the relations. Often, we cannot know a-priori which might be the KB relations to hypothesize with. It can be even more complicated for simple OIE patterns like *Obama likes Blues*, one can *like* an inanimate object or another person. There can be alternate possibilities to model the entire disambiguation (relation and instances) task without seeds with MLN or Probabilistic Soft Logic [KBB+12, PMGC13] or even with satisfiability solvers.

- Another important area to explore would be to find missing instance types. We have encountered quiet often that some DBpedia instances lacked the type information. Now, given a set of evidence facts f^+ from OIE, can we use the instance matchings to find probable instance types?

This thesis work presents a very basic and simplistic framework with the assumption that there isn't any context available. We performed implicit exploitation of contexts by probabilistic α-tree scheme and by rule based semantics. The results show that, our methods and approaches were powerful enough to built up an evidence base on the basis of the input data sets only. We did not aim at designing another entity disambiguation system but a framework for context free translation of OIE triples to a target knowledge base. This framework serves as an end-to-end pipeline for complete transformation from an ambiguous input triple to a semantified new fact expressed in terms of the KB vocabulary.

Part VI

APPENDIX

A

NOTATIONS

$\mathcal{U}$: set of OIE facts.

$\mathcal{O}$: set of KB facts.

f^+ : set of OIE facts which have an assertion in the KB. $f^+ \subseteq \mathcal{U}$

f^+ : set of OIE facts which do not have an assertion in the KB. $f^+ \subseteq \mathcal{U}$

$\mathcal{H}$: Matching hypothesis consisting of top-k candidate mappings. For easy reference, '$\mathcal{H}$' stands for hypothesis.

$\mathcal{M}$: Matching hypothesis consisting of top-1 candidate mapping. For easy reference, '$\mathcal{M}$' stands for mapping.

$\mathcal{M}^R$: Matching set for the relations. R stands for relations.

$\mathcal{T}$: set of DBpedia instances associated in some owl object properties.

$\mathcal{T}^R$: set of DBpedia relations, essentially these are owl object properties.

$\mathcal{S}$: set of OIE terms associated in any semantic relationship.

$\mathcal{S}^R$: set of OIE relations or relation phrases defining some semantic relationship between OIE terms.

$\mathcal{N}$: set of OIE triples mapped to new KB triples.

B

BACKGROUND

Markov Logic Network

A Markov network $\mathcal{MN}$ is an undirected graph $\mathcal{G}$ consisting of a set of nodes and connecting edges. The nodes represent a random variable and the connecting edges model dependencies with the connected nodes. For a set of random variables $\mathcal{X} = \{x_1, x_2, \ldots x_n\}$, a markov network models the joint probability distribution over $\mathcal{X}$ expressed as $p(x_1, x_2, \ldots x_n)$. Since the random variables(nodes) in general tend to form stronger dependencies with its adjacent nodes, the graph can be factorized into potential functions $\phi(x)$, defined as a non-negative function over the variable x, such that $\phi(x) \geqslant 0$ [Bar11]. A joint potential $\phi(x_1, x_2, \ldots x_n)$ is a non-negative function over $\mathcal{X}$. Hence, the model can be represented as a product of potential functions and its joint probability distribution is a normalized value of the product.

For instance, let us consider four random variables a, b, c, d modeled with two different dependencies as shown in Figure.

(a) (b)

Figure 19: A Markov Network with four random variables having two different dependencies amongst them.

For Figure 19(a) and 19(b) we respectively have,

$$p(x_1, x_2, \ldots x_n) = \phi(a, b)\phi(b, c)\phi(c, d)\phi(d, a)/Z_1 \tag{28}$$

$$p(x_1, x_2, \ldots x_n) = \phi(a, b, c)\phi(c, d)/Z_2 \tag{29}$$

where, Z is the normalizing constant, often called as *partition function* and defined as,

$$Z_1 = \sum_{a,b,c,d} \phi(a,b)\phi(b,c)\phi(c,d)\phi(d,a) \tag{30}$$

$$Z_2 = \sum_{a,b,c,d} \phi(a,b,c)\phi(c,d) \tag{31}$$

Generally, $\mathcal{MN}$ is defined as a product of potentials over the cliques of $\mathcal{G}$.

$$p(x_1, x_2, \ldots, x_n) = \frac{1}{Z} \prod_k \phi_k(\{x_k\})$$

where, $\{x_k\}$ is the state of the k^{th} clique. Markov networks are often represented in *log-linear models* where each of the clique potentials is an exponential weighted sum of the features of its state.

$$p(x_1, \ldots, x_n) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(D_i)\right) \tag{32}$$

where, w_i is a real-valued weight and $f_i(D_i)$ is a feature function from clique D_i to $\mathbb{R}$ [RD06].

"*A Markov logic network is a first-order knowledge base with a weight attached to each formula, and can be viewed as a template for constructing Markov networks.*" [RD06]. Intuitively, a markov logic network (MLN) is a way for expressing logical expressions/rules, annotated with weights which denote its confidence for holding true. It is considered a template for creating markov networks. under a given set of constant values, it can produce a number of markov networks of varying size but having a similar pattern, especially with respect to the formula weights given by the MLN. Each network can be grounded and essentially defines a world x. The probability distribution of x is given by [RD06],

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right) \tag{33}$$

where, $n_i(x)$ is the number of times the first order formula is true in the world x.

The complete idea of creating multiple markov networks can be illustrated with a simple example. Let us model a general statement "*everybody who runs is fit*". This

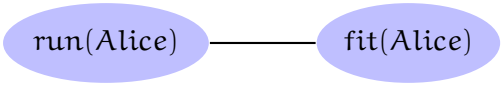

Figure 20: A ground markov network for the example.

statement is not an universally correct statement, so we can and we should attach some degree of uncertainty with it. This is formally expressed in first order logic as,

$$w: \quad (\forall x) run(x) \Rightarrow fit(x) \tag{34}$$

where w is a weight denoting the confidence of this formula. In this example, it is not important to exactly state the value of this weight, but higher values denote higher confidence. $run(x)$ and $fit(x)$ are atoms. And x is a variable and when each atom is replaced by actual constant values, it is called ground atom. For instance, hypothetically, if there is just one person $\{Alice\}$, then $fit(Alice)$ is a ground atom. And the ground instance of the formula looks like,

$$w: \quad run(Alice) \Rightarrow fit(Alice) \tag{35}$$

Usually each of these ground atoms are boolean variables, i.e. either a person runs or doesn't, then the following markov network, as shown in Figure 20, can exist in 4 (possibilities 0 or 1 for each of the 2 variables, = 2^2) possible forms.

In MLN terminology, each such variable combination state is called a *world*. Hence, $\{run(Alice) = 1, fit(Alice) = 0\}$ is a world[1]. We have 4 possible worlds in this simple example. The whole idea of MLN is to make a world less likely if the formula defining the world fails to hold and likewise make those worlds more likely where the first order formula holds. We use a simple truth table (Table 19) to capture this scenario and work out the probabilities for each of the worlds. We employ the probability distribution given by Equation 33. It is to be noted that the random variable X is the variable $(run(Alice), fit(Alice))$, which assumes 4 values as shown in the table. $n_i(x)$ denotes the number of times the Formula 35 evaluates to true in that world. Replacing the values in Equation 33, i determines the number of formulae, hence it is 1 here. Note that to evaluate the formulae we

1 Suppose there was one more person "Bob", then a particular world might have looked like, $\{run(Alice) = 1, fit(Alice) = 0, run(Bob) = 1, fit(Bob) = 1\}$, giving us 4^2 possible worlds. Observe the exponential growth in the possible worlds with growing number of variables. The variable X would have been $(run(Alice), fit(Alice), run(Bob), fit(Bob))$

X		Clausal form		
run(Alice)	fit(Alice)	¬run(Alice) ∨ fit(Alice)	n	$e^{n(x)*w}$
0	0	1	1	e^w
0	1	1	1	e^w
1	0	0	0	1
1	1	1	1	e^w

Table 19: Probability distribution of the possible worlds for the MLN $w: (\forall x)\ \mathrm{run}(x) \Rightarrow \mathrm{fit}(x)$ and $x \in \{\mathrm{Alice}\}$

express it in its clausal form. Hence, for x ∈ {(0,0), (0,1), (1,0), (1,1)}, we have,

$$P(0,0) = \frac{e^w}{Z} \tag{36}$$

$$P(0,1) = \frac{e^w}{Z} \tag{37}$$

$$P(1,0) = \frac{1}{Z} \tag{38}$$

$$P(1,1) = \frac{e^w}{Z} \tag{39}$$

As defined earlier, Z is the partition function and simply given by, $e^w + e^w + e^0 + e^w = 3e^w + 1$. It also clarifies the reason for Z being referred to as the normalizing factor. The most important message in this simple example is that MLN does not make the world (1,0) completely improbable even though the formula fails to hold. Rather, it is probabilistically evaluated to be less likely than the other 3 worlds. Ideally, our models are not as simple as this, rather contains a set of formulae with varying weights and a large number of constraints. This creates more complex networks with several magnitudes larger size and ground states.

Apart from formalism and modeling, we are actually interested in inference tasks. In general, there are two broad inference in graphical models. First, marginal inference, i.e. finding the probability of a particular ground atom, like P(fit(Alice)). This is calculated easily by a probability sum of the worlds where fit(Alice) holds 2 (rows 2 and 4 in Table 19). Second MAP or Maximum a posteriori estimation, which gives the variable X for which the probability of a particular world maximizes i.e.

$$\operatorname*{argmax}_x \ P(X = x)$$

In the above example, there are multiple configurations which maximizes the probability of a world, e.g. $x = \{(o,o)\}$. Our instance matching task is more concerned in finding such a maximum possible world. We have a comparatively huge network, with thousands of variables like $map(albert, Albert\_Einstein)$, $hasType(Person, Albert\_Einstein)$, and so on. These are mentioned in Section 6.2.3. Our modeling targets to make as many as map atoms true simultaneously. This is our final mapping set.

C

RESOURCES

Source Code

We release the source code as an open source project hosted on Github. The entire project is available as documented maven project at the link https://github.com/kraktos/ESKO The project has been named ESKO: an acronym for **E**nriching **S**tructured **K**nowledge Base from **O**pen Information Extraction. Please refer to the README file associated project for a detailed account of the execution steps and configurations. The project has few dependancies which have been mentioned in the project README. Furthermore, a documented configuration file CONFIG.cfg is used which set the important variables and parameters required for the workflows.

Data Sets

This work generated and used some data sets which are useful in general for the community. We provide the download links for each one of those.

- Wikipedia most frequent sense data. It consists over 9million distinct surface forms to Wikipedia articles and along with the count of number of links from each surface form to the article.
  http://web.informatik.uni-mannheim.de/adutta/wikiPrep.tar.gz
- Annotated set of NELL instance mappings.
  https://madata.bib.uni-mannheim.de/65/
- Instance and relations mappings for REVERB. Three files have been compressed and added into this .gz file. One is a gold standard for REVERB instances, the other for relations and a README file describing the individual file formats.
  http://web.informatik.uni-mannheim.de/adutta/REVERB_GS.tar.gz
- In our configuration, we often have to query for the type of a DBPEDIA instance. The usual way is through a SPARQL endpoint, but a network call can

be slow and time consuming at times, especially, when a large number of instances are queried for. This is a sql dump file which contains the DBpedia instances and its type information as a relational table. To use this file in tandem with the source code, one must set the variable "RELOAD_TYPE" in Config.cfg to false.

http://web.informatik.uni-mannheim.de/adutta/DBPTypes.tar.gz

BIBLIOGRAPHY

[ABK+07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proc. of the 6th International Semantic Web Conference*, pages 722–735, 2007.

[ACG02] Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 586–597. VLDB Endowment, 2002.

[AdS09] Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. Knowledge-based WSD on specific domains: performing better than generic supervised WSD. In *Proc. of IJCAI-09*, 2009.

[AG00] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 85–94, New York, NY, USA, 2000. ACM.

[AMC14] Isabelle Augenstein, Diana Maynard, and Fabio Ciravegna. Relation extraction from the web using distant supervision. In *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings*, pages 26–41, 2014.

[AVH+12] Alan Akbik, Larysa Visengeriyeva, Priska Herger, Holmer Hemsen, and Alexander Löser. Unsupervised discovery of relations and discriminative extraction patterns. In Martin Kay and Christian Boitet, editors, *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 17–32. Indian Institute of Technology Bombay, 2012.

[Bar11] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 04-2011 edition, 2011. In press.

[BCS+07] Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the Web. In *Proc. of IJCAI-07*, pages 2670–2676, 2007.

[BE08] Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In Kathleen McKeown, Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, editors, *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 28–36. The Association for Computer Linguistics, 2008.

[BEP+08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. of the 2008 ACM SIGMOD international conference on Management of data*, 2008.

[BLK+09] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. {DBpedia} – {A} Crystallization Point for the Web of Data. *Journal of Web Semantics*, 7(3), 2009.

[BM05] Razvan C. Bunescu and Raymond J. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 724–731, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[BMC+03] Mikhail Bilenko, Raymond Mooney, William Cohen, Pradeep Ravikumar, and Stephen Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, September 2003.

[BP06] Razvan Bunescu and Marius Paşca. Using encyclopedic knowledge for named entity disambiguation. In *Proc. of EACL-06*, 2006.

[Bri98] Sergey Brin. Extracting patterns and relations from the world wide web. In *In WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT 98*, pages 172–183, 1998.

[BvH06] Sylvain Brohée and Jacques van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7(1), 2006.

[CBK+10] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proc. of AAAI*, 2010.

[Coh98] William W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 201–212, New York, NY, USA, 1998. ACM.

[CSZ10] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 1st edition, 2010.

[Cuco7] Silviu Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of EMNLP-CoNLL-07*, 2007.

[DJ79] Richard Dubes and Anil K. Jain. Validity studies in clustering methodologies. *Pattern Recognition*, 11(4):235–254, 1979.

[dLL13] Oier Lopez de Lacalle and Mirella Lapata. Unsupervised relation extraction with general domain knowledge. In *EMNLP*, pages 415–425. ACL, 2013.

[DMP14] Arnab Dutta, Christian Meilicke, and Simone Paolo Ponzetto. A probabilistic approach for integrating heterogeneous knowledge sources. In *ESWC-14*, pages 286–301, 2014.

[DMR+10] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. Entity disambiguation for knowledge base population. In *Proc. of COLING-10*, 2010.

[DMS14] Arnab Dutta, Christian Meilicke, and Heiner Stuckenschmidt. Semantifying triples from open information extraction systems. In *Frontiers in Artificial Intelligence and Applications*, volume 264. IOS Press, 2014.

[DMS15] Arnab Dutta, Christian Meilicke, and Heiner Stuckenschmidt. Enriching structured knowledge with open information. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 267–277, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[DNMP13] Arnab Dutta, Mathias Niepert, Christian Meilicke, and Simone Paolo Ponzetto. Integrating open and closed information extraction : Challenges and first steps. In *Proc. of the ISWC-13 NLP and DBpedia workshop*, 2013.

[DRST12] Lucas Drumond, Steffen Rendle, and Lars Schmidt-Thieme. Predicting rdf triples in incomplete knowledge bases with tensor factorization. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 326–331, New York, NY, USA, 2012. ACM.

[DS14] Arnab Dutta and Michael Schuhmacher. Entity linking for open information extraction. In Elisabeth MÃ©tais, Mathieu Roche, and Maguelonne Teisseire, editors, *Natural Language Processing and Information Systems*, volume 8455 of *Lecture Notes in Computer Science*, pages 75–80. Springer International Publishing, 2014.

[dSMSB13] Filipe de Sá Mesquita, Jordan Schmidek, and Denilson Barbosa. Effectiveness and efficiency of open relation extraction. In *EMNLP*, pages 447–457. ACL, 2013.

[DWL+04] B. Dorow, D. Widdows, K. Ling, J.-P. Eckmann, D. Sergi, and E. Moses. Using Curvature and Markov Clustering in Graphs for Lexical Acquisition and Word Sense Discrimination. *eprint arXiv:cond-mat/0403693*, March 2004.

[ECD+04] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in KnowItAll (Preliminary results). In *WWW*, 2004.

[ECD+05] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91 – 134, 2005.

[EFC+11] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: The second generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pages 3–10. AAAI Press, 2011.

[Etz11] Oren Etzioni. Search needs a shake-up. *Nature*, 476(7358):25–26, August 2011.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *IN SIGCOMM*, pages 251–262, 1999.

[FSE11] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[FSEC] Anthony Fader, Stephen Soderland, Oren Etzioni, and Turing Center. Scaling wikipedia-based named entity disambiguation to arbitrary web text. *User-Contributed Knowledge and Artificial Intelligence: An Evolving Synergy*, page 21.

[FSSS09] Thomas Franz, Antje Schultz, Sergej Sizov, and Steffen Staab. Triplerank: Ranking semantic web data by tensor decomposition. In *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, pages 213–228, 2009.

[GBVR03] Lifang Gu, Rohan Baxter, Deanne Vickers, and Chris Rainsford. Record linkage: Current practice and future directions. Technical report, CSIRO Mathematical and Information Sciences, 2003.

[GLG+13] Abhishek Gattani, Digvijay S. Lamba, Nikesh Garera, Mitul Tiwari, Xiaoyong Chai, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Entity extraction, linking, classification, and tagging for social media: A wikipedia-based approach. *Proc. VLDB Endow.*, 6(11):1126–1137, August 2013.

[GM06] Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proc. of AAAI-06*, 2006.

[GM07] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. of IJCAI-07*, 2007.

[HKF+13] Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. Umbc ebiquity-core: Semantic textual similarity systems. *2nd Joint Conf. on Lexical and Computational Semantics, Association for Computational Linguistics*, page 44, 2013.

[HNP13] Eduard Hovy, Roberto Navigli, and Simone Paolo Ponzetto. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27, 2013.

[HSG04] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discovering relations among named entities from large corpora. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[HTN+14] Yi Huang, Volker Tresp, Maximilian Nickel, Achim Rettinger, and Hans-Peter Kriegel. A scalable approach for statistical learning in semantic graphs. *Semantic Web*, 5(1):5–22, 2014.

[HW79] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[HYB+11] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proc. of EMNLP'11*, 2011.

[JG11] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Proc. of ACL-11*, 2011.

[JH10] Xin Jin and Jiawei Han. K-medoids clustering. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 564–565. Springer, 2010.

[JLM03] Liang Jin, Chen Li, and Sharad Mehrotra. Efficient record linkage in large data sets. In *Proceedings of the Eighth International Conference on Database Systems for Advanced Applications*, DASFAA '03, pages 137–, Washington, DC, USA, 2003. IEEE Computer Society.

[JTHN12] Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. Link prediction in multi-relational graphs using additive models. In Marco

de Gemmis, Tommaso Di Noia, Pasquale Lops, Thomas Lukasiewicz, and Giovanni Semeraro, editors, *SeRSy*, volume 919 of *CEUR Workshop Proceedings*, pages 1–12. CEUR-WS.org, 2012.

[Kam04] Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[KBB+12] Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. In *NIPS Workshop on Probabilistic Programming: Foundations and Applications*, 2012.

[KD08] Stanley Kok and Pedro Domingos. Extracting semantic networks from text via relational clustering. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pages 624–639, Berlin, Heidelberg, 2008. Springer-Verlag.

[KHY+14] Abhay Kashyap, Lushan Han, Roberto Yus, Jennifer Sleeman, Taneeya Satyapanich, Sunil Gandhi, and Tim Finin. Meerkat mafia: Multilingual and cross-level semantic textual similarity systems. *SemEval 2014*, page 416, 2014.

[Lar10] Ray R. Larson. Information retrieval: Searching in the 21st century; human information retrieval. *Journal of the American Society for Information Science and Technology*, 61(11):2370–2372, 2010.

[LC01] K. Lai and Narciso Cerpa. Support vs confidence in association rule algorithms. In *OPTIMA*, 2001.

[LLZ+13] Fang Liu, Yang Liu, Guangyou Zhou, Kang Liu, and Jun Zhao. Determining relation semantics by mapping relation phrases to knowledge base. In *2nd IAPR Asian Conference on Pattern Recognition, ACPR 2013, Naha, Japan, November 5-8, 2013*, pages 420–424. IEEE, 2013.

[LME12a] Thomas Lin, Mausam, and Oren Etzioni. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages

84–88, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[LME12b] Thomas Lin, Mausam, and Oren Etzioni. No noun phrase left behind: Detecting and typing unlinkable entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 893–903, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[MBSJ09] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 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: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Mil95] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[MJGSB11] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia Spotlight : Shedding light on the web of documents. In *Proc. of I-Semantics'11*, 2011.

[MN13] Andrea Moro and Roberto Navigli. Integrating syntactic and semantic analysis into the open information extraction paradigm. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2148–2154. AAAI Press, 2013.

[MSB+12] Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. Open language learning for information extraction. In *Proc. of (EMNLP-CONLL)*, 2012.

[MW08] David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 509–518, New York, NY, USA, 2008. ACM.

[Nav09] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69, February 2009.

[NNS13] Jan Noessner, Mathias Niepert, and Heiner Stuckenschmidt. RockIt: Exploiting parallelism and symmetry for map inference in statistical relational models. In *Proc. of AAAI-13*, 2013.

[NP12] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[NTK11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 809–816. ACM, 2011.

[NWS12] Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1135–1145, 2012.

[PD09] Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 1–10, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Pea05] K. A. R. L. Pearson. The Problem of the Random Walk. *Nature*, 72(1867):342, August 1905.

[PMGC13] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Large-scale knowledge graph identification using psl. In *AAAI Fall Symposium on Semantics for Big Data*, 2013.

[PN10] Simone Paolo Ponzetto and Roberto Navigli. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1522–1531, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[PVD08] Marius Paşca and Benjamin Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from Web documents and query logs. In *Proc. of ACL-08*, 2008.

[PY13] Jakub Piskorski and Roman Yangarber. Information extraction: Past, present and future. In Thierry Poibeau, Horacio Saggion, Jakub Piskorski, and Roman Yangarber, editors, *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 23–49. Springer Berlin Heidelberg, 2013.

[RD06] Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, February 2006.

[RL99] Bhavani Raskutti and Christopher Leckie. An evaluation of criteria for measuring the quality of clusters. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, pages 905–910, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[RNX+12] Shu Rong, Xing Niu, EvanWei Xiang, Haofen Wang, Qiang Yang, and Yong Yu. A machine learning approach for instance matching based on similarity metrics. In *The Semantic Web – ISWC 2012*, volume 7649 of *Lecture Notes in Computer Science*, pages 460–475. Springer, Berlin and Heidelberg, 2012.

[RR07] Leonard Richardson and Sam Ruby. *Restful Web Services*. O'Reilly, first edition, 2007.

[RYMM13] Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. Relation extraction with matrix factorization and universal schemas. In *(HLT-NAACL '13)*, 2013.

[SAS11] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. Paris: probabilistic alignment of relations, instances, and schema. *Proc. VLDB Endow.*, 5(3):157–168, November 2011.

[SC12] Valentin I. Spitkovsky and Angel X. Chang. A cross-lingual dictionary for english wikipedia concepts. In *Proc of LREC-12*, pages 3168–3175, 2012.

[SG10] Ang Sun and Ralph Grishman. Semi-supervised semantic pattern discovery with guidance from unsupervised pattern clusters. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1194–1202, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[SKW07] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *Proc. of WWW-07*, 2007.

[SM07] Stephen Soderland and Bhushan Mandhani. Moving from textual relations to ontologized relations. In *AAAI Spring Symposium: Machine Reading*, pages 85–90. AAAI, 2007.

[SP14] Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-based graph document modeling. In Ben Carterette, Fernando Diaz, Carlos Castillo, and Donald Metzler, editors, *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, pages 543–552. ACM, 2014.

[SRQ+10] Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Mausam, and Oren Etzioni. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102, 2010.

[SSW09] Fabian Suchanek, Mauro Sozio, and Gerhard Weikum. SOFIE: A self-organizing framework for information extraction. In *WWW'09 : proceedings of the 18th International World Wide Web Conference*, pages 631–640, Madrid, Spain, 2009. Association for Computing Machinery (ACM), ACM.

[STNM12] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 455–465, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[SWH15] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *Knowledge and Data Engineering, IEEE Transactions on*, 27(2):443–460, Feb 2015.

[TSN11] Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. Probabilistic matrix factorization leveraging contexts for unsupervised relation extraction. In Joshua Zhexue Huang, Longbing Cao, and Jaideep Srivastava, editors, *PAKDD (1)*, volume 6634 of *Lecture Notes in Computer Science*, pages 87–99. Springer, 2011.

[VBGK09] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk - A Link Discovery Framework for the Web of Data. In *Proc. of LDOW '09*, 2009.

[vDoo] Stijn van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, Utrecht, May 2000.

[Vel12] T Velmurugan. Efficiency of k-means and k-medoids algorithms for clustering arbitrary data points. *International Journal of Computer Technology & Applications*, 3(5):1758–1764, 2012.

[WCG+12] Daisy Zhe Wang, Yang Chen, Sean Goldberg, Christan Grant, and Kun Li. Automatic knowledge base construction using probabilistic extraction, deductive reasoning, and human feedback. In *Proceedings of the Joint Workshop on*, AKBC-WEKEX '12, pages 106–110, 2012.

[Win95] William E. Winkler. Matching and record linkage. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(5):313–325, 1995.

[WW07] Fei Wu and Daniel S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 41–50, New York, NY, USA, 2007. ACM.

[WW10] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[YC13] Daisy Zhe Wang Yang Chen. Web-scale knowledge inference using markov logic networks. *ICML workshop on Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*, June 2013.

[YEo9] Alexander Yates and Oren Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *J. Artif. Intell. Res. (JAIR)*, 34:255–296, 2009.

[ZG05] Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 419–426, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[ZHW12] Congle Zhang, Raphael Hoffmann, and Daniel S. Weld. Ontological smoothing for relation extraction with minimal supervision. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.* AAAI Press, 2012.

[ZNL+09] Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. Statsnowball: A statistical approach to extracting entity relationships. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 101–110, New York, NY, USA, 2009. ACM.

EHRENWÖRTLICHE ERKLÄRUNG

Ich versichere, dass ich die beiliegende Doktorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Arnab Kumar Dutta,
Mannheim, Deutschland,
March 3, 2016