Roma Tre University

Department of Computer Science and Engineering
Ph.D. in Computer Science and Engineering

Augmenting Knowledge Graphs with Natural Language Evidence

Candidate

Matteo Cannaviccio

Advisor

Prof. Paolo Merialdo

Co-advisor

Prof. Denilson Barbosa

Big Data and Databases Research Group

March 2018

XXX Ph.D Cycle

Augmenting Knowledge Graphs with Natural Language Evidence

A thesis presented by

Matteo Cannaviccio

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Computer Science and Engineering

Roma Tre University

Department of Computer Science and Engineering

Committee:

prof. Paolo Merialdo (Roma Tre University)

prof. Denilson Barbosa (University of Alberta)

Reviewers:

prof. Martin Theobald (University of Luxembourgy)

prof. Luciano Barbosa (Universidade Federal de Pernambuco)

Ph.D. Course Coordinator:

prof. Stefano Panzieri (Roma Tre University)

To myself

Acknowledgements

These years have been very important both for my professional and personal growth, defining a significant chapter of my life. For this reason, a number of people deserve to be acknowledged for the support they gave me.

First of all, I would like to express my gratitude to my advisor, Paolo Merialdo, for his passion and guidance that have been fundamental for me during these years. I sincerely thank Denilson Barbosa as well, who trusted in me from the very beginning and introduced me to research on these very interesting topics. His continued enthusiasm during my master thesis inspired me to begin this journey.

I would like to thank my colleagues, both at Roma Tre and at the UofA, and all the students that I guided during these years which contributed to this result. A particular thank goes to Antonio, for all the valuable discussions that taught me a lot. Thanks to Nicola, Andrea, Marco, Federico and Bruno, for being always by my side in every single adventure of my life.

Most of all, I am grateful to my family for supporting me over these years. To Claudia and Marco who believed in me and provided support in difficult moments. Special thanks go to Adri, who always reminds me the importance of smiling. I want to thank my father for encouraging me everyday and my mother, whose life instilled in me the strength to face every possible challenge.

Finally, I thank Silvia, for her love, her incredible patience and her encouragements throughout this not always easy process.

I could not have done this without any of you.

Abstract

In last ten years, massive amounts of world knowledge have been accumulated into large knowledge graphs (KGs). These knowledge repositories store millions of facts about the world, such as information about people, places and organizations, and have become a powerful asset for semantic applications such as search, analytics, recommendations, and data integration. Several approaches have been proposed to create KGs from Wikipedia, as in the cases of YAGO and DBpedia, or collaboratively as for Freebase and Wikidata. Despite their seemingly huge size, these knowledge graphs are greatly incomplete and approaches to populate them automatically are needed to increase their coverage.

This thesis describes principled methods to model knowledge graph relations with natural language. These models allow the extraction of facts from text or to annotate web tables with KG relations, with the aim of populating state-of-the-art KGs. The first contribution is a pattern-based extraction system which can extract automatically high-quality facts from the text of Wikipedia articles. Indeed, the approaches used to derive KGs from Wikipedia are focused only on its structured components like the info-boxes. Although valuable, they represent only a fraction of the actual information expressed in the articles. We experiment our system on five different languages, showing that it can extract a large number of facts that are out of reach of common infobox-based extractions. The second contribution is an approach that uses language models, derived from a Web-scale corpus, to rank KG relations that hold over pairs of entities juxtaposed in tables or structured lists. Our experimental evaluation shows the effectiveness of the approach in predicting KG relations even when entities are missing from the graph and thus represents a significant advancement of the state-of-the-art.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays an increasing amount of information is generated through the World Wide Web and expressed on a large variety of sources such as textual articles, data-intensive web sites, social media, to name a few. A user that is looking for some piece of information has to explore them manually in order to arrive to the answer. In this task, search engines helped a lot in driving the user to explore Web documents toward most relevant sources, but the process is still very chaotic.

In this scenario, the idea of Web of Data was to organize all the information contained on the Web creating a giant structured open repository. Based on Tim-Barnes Lee's vision, users would have to publish contents with semantic markup, in a way that not only humans but also machine could read and organize such information. This approach allowed for a rapid growth of the Semantic Web linked open data cloud (LOD)[1].

Despite this great idea most of the Web contents remain unstructured, but the Web of Data inspired the research on different aspects of knowledge-oriented systems. In 2012, Google announced the creation of the Google Knowledge Graph, a graph-based repository that encodes diverse areas of human knowledge about the real world. In this way they could provide directly the information needed by the user instead of potentially relevant sources. The Google Knowledge Graph bears considerable impact in improving the quality and user-experience of Web search. Also, it enables many

[1] http://lod-cloud.net/

other AI applications, such as question-answering, data analytics, user recommendation, etc. For this reason many prominent Web-based companies started to built their own knowledge-based repositories: Facebook created the Graph Search[2] to associated users with real-word entities, Microsoft developed Satori to assist the user across its different applications, increasing the hype around knowledge graph technologies.

In the past decades many research challenges have been faced in different areas of Artificial Intelligence, Information Extraction and Natural Language Processing in order to make machines able to mine knowledge from unstructured sources. Thus, instead of rewriting the Web in a way that the machines can understand, the direction of the research in these fields is to enable the machines to create and populate very large knowledge graphs automatically.

1.1 From Information to Knowledge

Today different kind of knowledge graphs have been created. Some of them represent domain specific knowledge, for example the Gene Ontology[3], or industry-oriented such the LinkedIn Economic Graph[4]. Cross-domain knowledge graphs, instead, are focused on *common knowledge*, that are the information that humans generally know about the world, for example “Rome is the capital of Italy" or “the director of Blade Runner", etc.. Differently from domain-specific information, this kind of knowledge is easily expressed in different Web pages through text, tables or other kind of unstructured sources.

The largest available KGs that contain this kind of knowledge are built collaboratively from communities of volunteers, for example Freebase [2] and Wikidata [3], using manual or semi-automatic approaches. Furthermore, there has been several important and mostly academic projects that aim to build large knowledge graphs automatically from Wikipedia, as in the case of YAGO [4] and DBpedia [5], or from the Web, such as NELL [6], PROSPERA [7] or the Google Knolwedge Vault [8].

[2] http://www.facebook.com/about/graphsearch
[3] http://www.geneontology.org/
[4] https://www.linkedin.com/economic-graph

We use the term **open-world** to indicate this kind of knowledge graphs, which cover a wide range of domains into a single schema. The representation of such knowledge is based on the concepts of **entities**, **types**, **relations** and **facts**. The types represent classed of objects, for example cities, companies, actors, and the entities are instances or those classes. For example “Quentin Tarantino", “Pulp Fiction" and “Amsterdam" are all entities, from different domains. Entities can be linked each others, through different type of relations. For example the place of birth of a person or the director of a movie. Usually, both the relations and the types are represented by unique global identifiers and they are part of a predefined schema. Finally, the facts are instances of relations and are expressed by a pair of entities linked by a specific relation. For example, a fact can record that “Quentin Tarantino" is the director of “Pulp Fiction".

1.2 Challenges

The construction of large knowledge graphs has become a major research field in the last ten years. Many knowledge graphs have been created over the semi-structured knowledge that is exposed in Wikipedia[5], the largest available online encyclopedia. Others approaches follow collaborative processes that require an high level of human effort from a community of volunteers. Despite their large size in terms of facts contained, all those knowledge graphs are still very incomplete and there are many open problems in obtaining methods to scale their automatic population.

Some recent works in this area [9, 10] show the limited coverage of Freebase, a KG obtained using a collaborative approach. For example, over 70% of the people contained in the graph are not associated with their place of birth, and 99% have no information about their ethnicity [10]. In the case of Freebase, the missing information are probably due to the large number of entities that have been added automatically into the graph for which people do not have any knowledge about them, signaling a clear limit of collaborative approaches in the long run.

[5] https://www.wikipedia.org

A similar situation occurs with DBpedia, which is a knowledge graph automatically extracted from the information in Wikipedia info-boxes. In this case 58% of scientists do not have a fact that describe what they are known for, and 40% of the countries do not have the association with their capital city [11]. These numbers reveal important limits on the ways these knowledge graphs are built, which inevitably miss out most of the real world knowledge that exist on the Web [12].

1.2.1 Knowledge Harvesting from Text

Harvesting knowledge from textual sources is crucial to populate an incomplete knowledge graphs and obtain an high coverage. Techniques used to face the task have evolved over time, from *supervised* approaches which formulate the problem of relation extraction as a binary classification problem, to *semi-supervised* bootstrapping techniques which only require a small set of tagged seed instances (pairs of entities involved in a relation) to start an iterative extraction process. On the one hand, the systems built with supervised approaches usually obtain the best performance, but they were difficult to apply in the case of knowledge graphs because of the manual labeled examples required for each relation. On the other hand, the bootstrapping approaches suffer the problem of *semantic drift*, a tendency to move too far away from the target relation to harvest which may require additional human intervention during the learning process.

With the increasing popularity of the knowledge graphs a technique called *distant supervision* has been used to train relation extraction classifiers aligning a knowledge graph with textual documents: if a sentence contain entities that are involved in a relation (from the KG) it become a positive example for that relation. The approach can be used to generate very large noisily annotated corpora. The noise is present essentially for two reasons: (a) at labelling time we can not know if the sentences is actually expressing the relation; (b) with the KG we have only positive examples, without the possibility to obtain negative labels. In general, the current state-of-the-art approaches for the task of populating a knowledge graph are based on a combination

of distant supervision and supervised approach[6]. However, it is still difficult to achieve successful results at Web-scale leaving it as a currently open problem.

Speaking of Wikipedia, most of the knowledge is offered in the text of the articles, with only the main information structured in the info-boxes, but the approaches that has been proposed to build knowledge graphs from it focused only on those structured information. Despite this, the encyclopedic nature of Wikipedia ensures that the language used to express facts in the text is very homogeneous (compared to other common web documents) which is an advantage for training knowledge extraction systems. In **Chapter 4** we proposed a pattern-based method that relies on a soft distant supervision for training. The extraction system is able to extract very accurate facts from different languages from the text of Wikipedia articles and use them to augment state-of-the-art KGs such as Freebase and DBpedia.

1.2.2 Understanding Web Tables

In 2011, an influential research from Google revealed that the Web contained around 154 million high-quality HTML tables [13]. These tables are widely used structure to express relational information on a variety of domains, which could be precious in improving search quality [14] or for the task of populating existing knowledge graphs. In both the cases, the task is to *understand* the table or, in other words, to align its content with the KG in order to handle the information provided. Generally, there are three steps associated with understanding Web tables: (1) identifying the “entity column" of the table; (2) assigning KG types for every column in the table; and (3) labeling the relationships involving entities in the “entity column" and the other columns.

Recent works on Web table understanding, which rely on existent knowledge graphs to perform the necessary steps, annotate the cells of a given table with entity identifiers, and pairs of columns with relations from the KG that hold over the linked entities. The techniques vary from using probabilistic graphical models against the YAGO knowledge

[6] https://kbpo.stanford.edu/

graph [15] or matching each row of the table against possible DBpedia entities taking into account how well the table headers match classes in the DBPedia ontology [16, 17]. In both the cases, given the incompleteness problem of modern KGs, matching the span of text contained in the cells with the exact entities in the KG is a very hard problem, especially where is not enough context that can be exploited for disambiguation. In **Chapter 5** we describe an alternative method to predict a KG relation given a list of entity pairs extracted from a table. It aims to rank relations from the knowledge graph using generative language models derived from Web-scale corpora instead of entity linking. As such, it can produce high quality results even when the entities in the table are missing from the graph.

1.3 Contribution and Novelty of this Thesis

This thesis focuses on investigating possible approaches to turn the information contained in unstructured sources, such as text or tabular structures, into a graphical form using the schema provided from an existent KG. The main challenge is to associate natural language expressions with the relations that they describe in the reference KG. Once we obtain them, we can use their evidence to harvest structured information from textual document or align tabular data to the KG, with the final goal of augmenting it with new information.

Some parts of this thesis has been published in international conferences. For instance, the **Lector** system, an extraction tool that is able to harvest facts from the text of Wikipedia articles and use them to augment Freebase is described in [18]. The process extracts facts with high-accuracy (over 95%) using a patterns-based approach.

Together with the tool, we also provide a dataset of ∼230 K extracted facts[7] which is a precious resource to test relation extraction approaches since they are missing from Freebase, but also Wikipedia info-boxes, YAGO and DBpedia.

[7] Available at: http://downloads.dbpedia.org/2016-04/ext/lector_facts/

The tool is later extended to DBpedia, resulting the winning system for the **2017 DBpedia TextExt Challenge**[8], a competition organized by DBpedia with the goal of exploring knowledge extraction from Wikipedia article text. The tool, which is called Lector+[9], was able to extract ∼1,8 M new facts from the text of the English Wikipedia, with an accuracy of 0.81. However, the system can be used to extract facts from the text of different language editions and will be integrated as a standard component of the DBpedia Extraction Framework[10]. The general architecture is described in [19].

Finally, we propose **Stup**, an approach that uses language models to represent KG relations and rank them based on how they represent pairs of entities juxtaposed in a table or structured list. The approach is an alternative to the existing solutions, which are hampered by the incompleteness and uneven coverage in even the best KGs available today, and can be use to predict relations even when entities are missing from the KG. We presented this contribution in [20].

Overview This dissertaiton is structured as follow:

- Chapter 2 gives an introduction to knowledge graphs, with the basic notions and the potential applications that can be enabled; it outlines possible approaches used to built knowledge graphs automatically from the Web and quantifies the level of incompleteness of some of the largest open-world KGs.
- Chapter 3 gives an overview of the state-of-the-art for extracting knowledge from unstructured sources. In particular it describes possible techniques used to populate knowledge graphs from textual documents and web tables.
- Chapter 4 describes Lector and the whole process used to process Wikipedia articles, detect the entities and associate natural language patterns to the relations. This chapter provides a complete evaluation of both **Lector** and **Lector+**, focusing on the improvements that we obtain with the second version of the tool.

[8] TextExt Challenge: http://wiki.dbpedia.org/textext
[9] Available at: https://gitlab.com/Rm3UofA/Lector/Code
[10] DBpedia Extraction Framework: https://github.com/dbpedia/extraction-framework/

- Chapter 5 describes **Stup**, our approach used to rank KG relations for a given set of entity pairs. The chapter describes the process used to create language models for the relations and provide a complete experimental validation showing that our approach is robust and effective in practice.
- Chapter 6 summarizes the work of this thesis.

Chapter 2

Knowledge Graphs

In this section we provide an exhaustive description of Knowledge Graphs (KG), describing the motivations that lead to rapidly increase their popularity the last years. We define the general concepts of open-world KGs, which are used to store knowledge about the world, but similar considerations are valid for more domain-specific ones. Finally we will talk about the main challenges in knowledge extraction, motivated by the natural incompleteness of every KG.

2.1 Open-World Knowledge Graphs

A Knowledge Graph can be seen as a structured repository in which we can store facts about entities that exist in the world. **Entities** are real-world objects (e.g. Sean Connery, Scotland, etc.) and represent the nodes of the graph. Those nodes usually have a precise identifier (e.g. Sean_Connery) and can be labelled with one or more textual expressions that represent their name, their title or alternative names used to to refer to them in the real-world (e.g. “Thomas Sean Connery", “Big Tam"). They can also have other textual or numeric **attributes**, for example the height, the age or the date of birth in case of person. Additionally, entities in schema-based knoweldge graphs are generally typed. The **types** are pre-defined classes from the schema, used to describe their nature and can be organized in an ontology. For example, an entity

can be a person, a location, or more specifically an actor, a book, a city, etc.

Entities can have different kind of relationships with other entities. For example, an actor can be related with the movies she acted in, or a town can be related with the country where it is located. With respect to a graph-based definition, the edges that link the nodes can have different labels that are called **relations** (e.g. <birthPlace>, <starring>, etc.). Finally, we define a **fact** as a single unit of information expressed by the graph. A fact is defined by a triple composed of a pair of entities (generally a subject and an object) and a relation. For example, the triple:

<Sean_Connery> <birthPlace> <Edinburgh>

is a fact describing that <Edinburgh> (object) is the place of birth of <Sean_Connery> (subject). Facts of a specific relation can be viewed as instances of that relation or labeled edges of the graph. Fig. 2.1 shows a snapshot of a possible knowledge graph describing information related to Sean Connery.

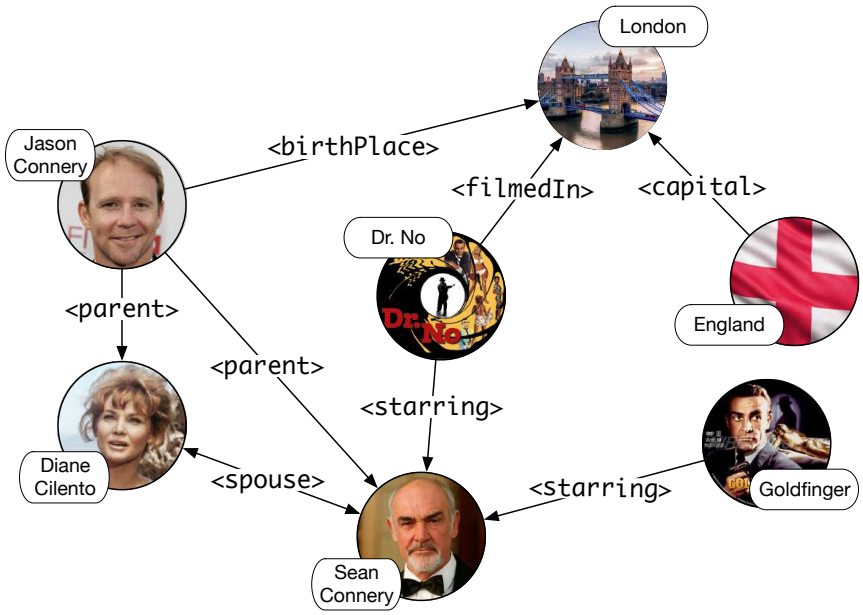


Figure 2.1: This is a snapshot of a possible open world KG.

2.2 Applications

Knowledge Graphs have gained much interest in the last years because they are key assets that enable a variety of semantic applications in different areas [21].

Web Search Knowledge Graphs are improving Web Search because they allow to search for entities and their relations with other entities, instead of keywords inside documents. With the help of a KG a search engine can disambiguate the intention of the user and provide more rich results which hopefully satisfy the information needs. For example, as in Fig. 2.2, Google search engine provides a box with all the relevant facts about the entity that has been looked-up.

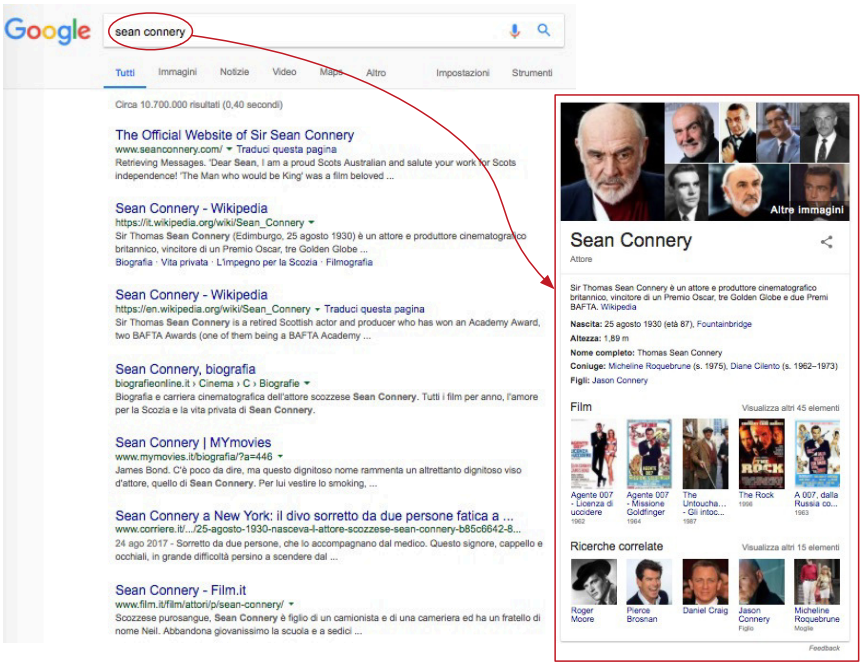


Figure 2.2: Result of the query “Sean Connery" by Google Search Engine.

Question-Answering This structured knowledge has become very useful to enable reasoning and inference, necessary for question-answering system. As showed in Fig. 2.3 with the Google search engine, a natural language expression can be translated into a structured query over the graph obtaining single facts or a list of entities as a result.

Much more work is still necessary both in natural language understanding and knowledge extraction to allow Knowledge Graphs to provide answer for very complicated questions but the direction is correct as confirmed by the result obtained by IBM Watson system at the Jeopardy game.

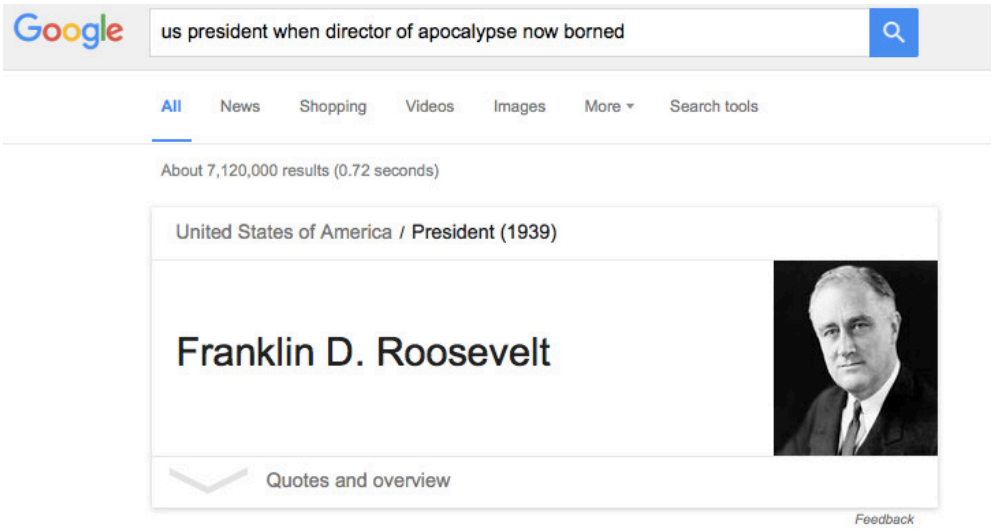


Figure 2.3: Result of a complex query to Google Search Engine.

2.3 Definition of a Knowledge Graph

We can define a knowledge graph as a labeled, directed multi-graph **KG=(N, E, L)** where **N** is the set of nodes, **E** is the set of edges, and **L** is the set of labels. The nodes are used represent entities and types, the labels are the relations and the edges are instances of relations or facts.

Most of the open-world KGs like YAGO [4], DBpedia [5] and Freebase [2] expose their information following the W3C Resource description Framework (RDF) standard. Moreover, these knowledge resources can be semantically interlinked at the entity level, contributing to the growth of the Linked Opne Data cloud. In this section we define the main concepts that are in common for many KGs using a simplified notation, but we give some references to their relative RDF notations.

Entities Entities are real-world objects such as person, location, movies, etc. and are represented using unique identifiers, internal to the KG. For example Freebase [2] uses "m-ids" (e.g., `m.03dgdg`), Wikidata [3] adopts internal IDs ("`Q`" followed by a number, e.g. `Q2096`) in order to be language-agnostic, while DBpedia and YAGO provide human-readable IDs (e.g. `Edmonton,_Alberta`). Since most of open-world KGs are represented using RDF standard, entities do not only have unique IDs, but URIs (Uniform Resource Identifiers) by which they can be referred to (e.g. `http://rdf.freebase.com/m/03dgdg` or `http://dbpedia.org/resource/Edmonton,_Alberta`). We will refer to entities using a human-readable notation, i.e. <Sean_Connery>.

Types Types are nodes of the graph that are used to assign classes to entities. As for the entities, a type has a unique internal identifier and a global URI which represent it as a resource. In schema-based KGs the types can be organized with an ontology that is part of the schema. For example, in DBpedia the ontology of types is defined in a hierarchy[1] similar to Fig. 2.4: in this case entities are labeled with a single type, not necessarily a leaf of the tree. In Freebase instead, the types describe the different perspectives that can be associated to an entity and do not have sub-types[2]: in this case an entity can be associated with any number of types. For example, in Fig. 2.5 the entity Barack Obama is both a person and a politician. We will refer to types using a human-readable notation, i.e. [Actor].

Literals Literals are particular nodes of the graph: they do not have an identifier and do not expose any outgoing connections with other nodes. Literals are used to describe a node (entities or types) with values such as strings, numbers and dates by means of a lexical representation (e.g. "*Big Apple*", "*1972-01-01*"). Most open-world KGs support simple data-types for literals such as a subset of the XML Schema (e.g. `xsd:date`).

Relations Relations are the labels used to specify the interconnections between two nodes. A relation is used to label the link between two entities, but also the link between

[1] DBpedia Ontology, http://mappings.dbpedia.org/server/ontology/classes/
[2] Freebase Basic Concepts, https://developers.google.com/freebase/guide/basic_concepts

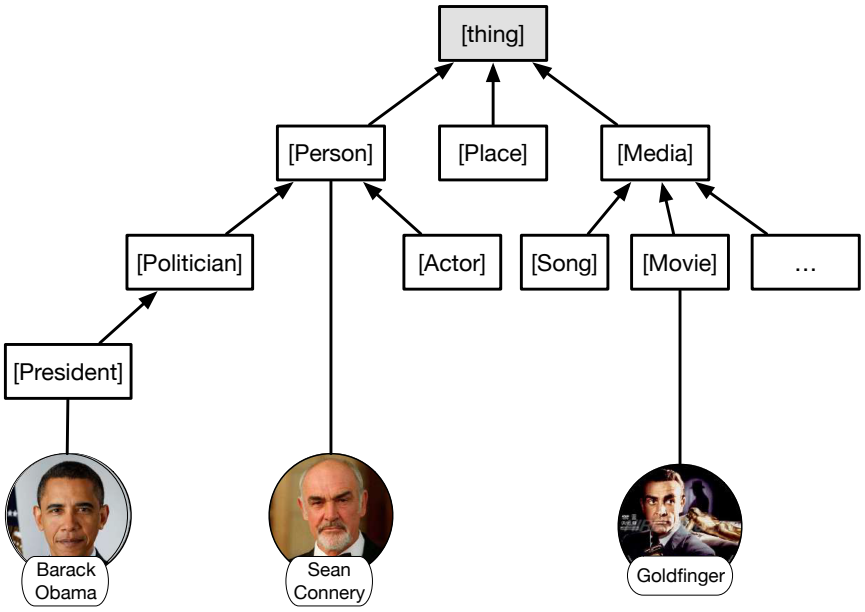


Figure 2.4: An example of DBpedia Ontology. Entities can be associated to intermediate nodes of the hierarchy.

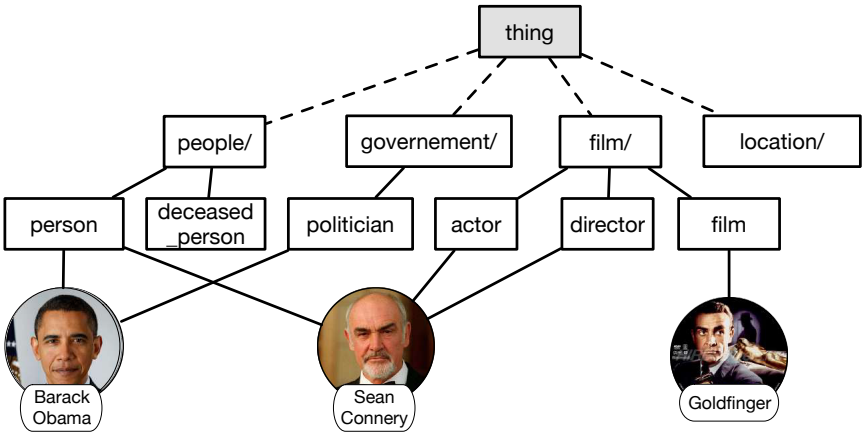


Figure 2.5: An example of the different domains of types in Freebase. Entities can be associated to multiple types in different domains.

an entity and a type and an entity with a literal. Relations in schema-based KGs are typed: they exists only between a specific `domain` (the type of the subject entity) and `range` (the type of the object entity). For example, the relation film/director/film in Freebase connects an entity of type film/director with an entity of type film/film[3]. We will refer to them using a simplified notation, i.e. <birthPlace>.

Facts Edges in $\mathbf{E} \subseteq \mathbf{N} \times \mathbf{L} \times \mathbf{N}$ are often called facts[4] and the effective links that record relationships between entities, types and literals. Most of open-world KGs store their facts in the form of RDF-style subject-predicate-object (SPO) triples. Table 2.1 lists most of the facts that are used to represent the KG snapshot of Fig. 2.1 and 2.4 with all the information described so far.

subject	relation	object
<Sean_Connery>	<spouse>	<Diane_Cilento>
<Diane_Cilento>	<spouse>	<Sean_Connery>
<Jason_Connery>	<parent>	<Diane_Cilento>
<Jason_Connery>	<parent>	<Sean_Connery>
<Dr._No>	<starring>	<Sean_Connery>
<Goldfinger>	<starring>	<Sean_Connery>
<Goldfinger>	<filmedIn>	<London>
<England>	<capital>	<London>
<Sean_Connery>	<label>	"*Sean Connery*"
<Diane_Cilento>	<label>	"*Diane Cilento*"
<Jason_Connery>	<dateOfBirth>	"*1963-11-01*"
<England>	<areaKm>	"*130279*"
	...	
<Sean_Connery>	<type>	[Actor]
<Diane_Cilento>	<type>	[Person]
<England>	<type>	[Country]
<London>	<type>	[City]
	...	
[Actor]	<subClassOf>	[Person]
[City]	<subClassOf>	[Settlement]
	...	

Table 2.1: A KG snapshot represented in form of facts, or RDF-triples.

[3] In Freebase each type provides a different set of relations that can be used to associate the entity with other entities

[4] Sometimes they are referred with *statements* (in Wikidata) or *triples* (following RDF-style)

Complex Relationships Facts are used to represented binary relations between entities. However, in the real world many complex relationships can exists between the entities. In order to represent such n-ary relations different solutions have been adopted. The standard way is called "reification" and consists in using a further resource (i.e. node) to denote a fact, then using other facts having the node as subject to add more information. An other option is to use a resource to represent those complex relations (e.g. "complex objects" used in Freebase). Instead of stating that a subject has a given value, the model states that the subject is involved in a relationship, and that that relationship has a value and some qualifiers.

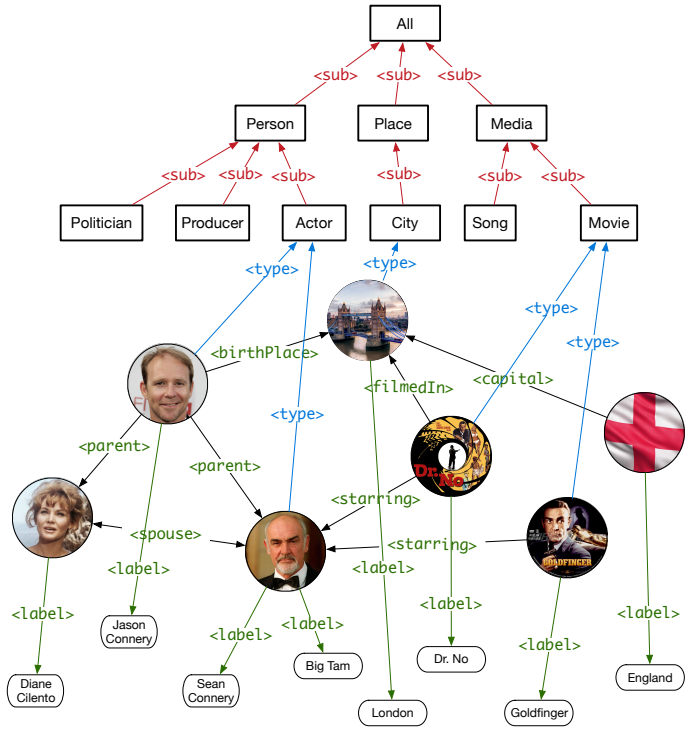


Figure 2.6: A KG snapshot represented in form of graph.

In this thesis we consider a KG as a set of binary relations between resources. It can be represented in the form of a graph, as in Fig. 2.6, where all the different nodes are connected each other through labelled edges, or using a set of RDF triples, as in Table 2.1. Both the representations describe the same information about <Sean_Connery>.

2.4 Knowledge Graphs Construction

The construction of large KGs has become a major research field in the last ten years. So far, KGs have been derived from accurate encyclopedic sources [5, 4] or following collaborative approaches that require an high level of human effort [2, 22]. This way of building knowledge graphs has been precious, but inevitably misses out most of the real world knowledge that exist on the Web [12]. Several attempts have been made in order to push the construction at the Web-scale, extracting facts from arbitrary Web pages and natural-language texts. Despite great advances in these regards, there are still many challenges regarding different aspects of knowledge extraction [21, 12]. In this section we briefly describe the evolution of such KGs construction methodologies.

2.4.1 Manually-curated Approaches

The first knowledge bases[5] have been built manually, typically compiled by domain experts, as in the cases of Cyc [23] and WordNet [24]. These knowledge sources served to support decision-making process of humans or machines, satisfying the highest quality expectations. For example, WordNet [24] groups nouns, verbs, and adjectives into sets of synonyms based on their senses and relationships[6] representing an important resource for many NLP applications. However, manual approaches are not a scalable, suffer from low coverage, and can be employed to create limited KGs that do not need to be updated frequently.

2.4.2 Collaborative Approaches

This kind of approaches are based on large community of voluntaries that contribute to insert and maintain the information in the graph. As in the case of Wikipedia, the largest on-line encyclopedia, this method can lead to obtain large KGs depending on the size of the community. Also, the content of collaborative KGs are biased on the

[5] The first knowledge repositories did not have any graph-based structure.
[6] Available at: http://wordnet.princeton.edu

popularity of the topics and the interests of the people that contribute. Freebase [2] and Wikidata [3] are two examples of large collaborative open-world KG.

Freebase Freebase is a collaborative knowledge graph created in 2007 by Metaweb. The information contained in Freebase are directly added by a large community of contributors which provide a minimum supervision. Many facts have been inserted automatically, moving information from external databases such as MusicBrainz[7], IMDb[8], and many others. In this way they obtain a very large knowledge graph that has been acquired by Google in 2010 and is the core of Google Knowledge Graph cite and Knowledge Vault [8] (see next section). In 2015 Google has retired Freebase and help the community with the transfer of the content to Wikidata [22].

Wikidata Wikidata is a project launched in October 2012 as part of the Wikimedia Foundation[9]. It has been presented as the "Wikipedia for data" [3] with the goal of structure the knowledge contained in Wikipedia through facts. An interesting possibility offered by Wikidata is that the facts are not necessarily true, but have to be referenced to the source. For example, border conflicts can be expressed from different political points of view [22]. In this way, two facts can contradict each other but, in the spirit of collaborative approaches, it is possible to encode different perspective of the same information.

2.4.3 Automatically-derived from Wikipedia

The automatic construction of knowledge graphs started around 2006 with YAGO [4] and DBpedia [5]. Those two academic projects aim to build a knowledge graph deriving information from Wikipedia. Both the projects exploit the semi-structured components of Wikipedia articles (info-boxes, categories, metadata, etc.). which expose a large number of facts. With the help of some hand-curated extraction rules they gather most

[7] https://musicbrainz.org/
[8] http://www.imdb.com/
[9] https://wikimediafoundation.org/

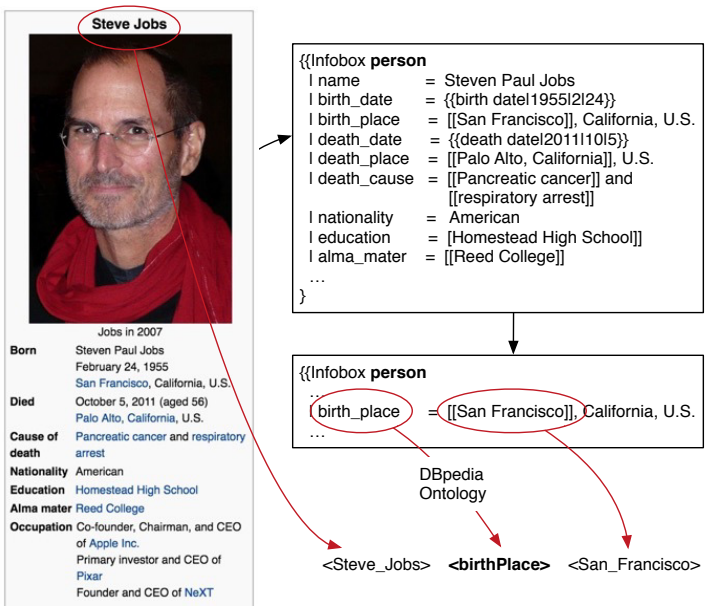


Figure 2.7: Example of a fact extracted from Steve Jobs info-box in Wikipedia.

of the information that are present and obtain large knowledge graphs. As we will discuss in the next section, both the KGs are still largely incomplete.

DBpedia DBpedia is a project started in 2006 as a collaboration of Free University of Berlin, Leipzig University and University of Mannheim, with the goal of extract structured information from Wikipedia. It has been a key factor in the growth of the Linked Open Data in that years so that it became its central hub[10].

An important step is represented by the creation of the DBpedia Ontology, that is the schema of classes and relations. The ontology is maintained and extended by the community through the DBpedia Mappings Wiki [11]. They allow to disambiguate attributes from Wikipedia info-boxes and align them to canonical relations and classes (example in Fig. 2.7).

[10] http://lod-cloud.net/
[11] http://mappings.dbpedia.org/

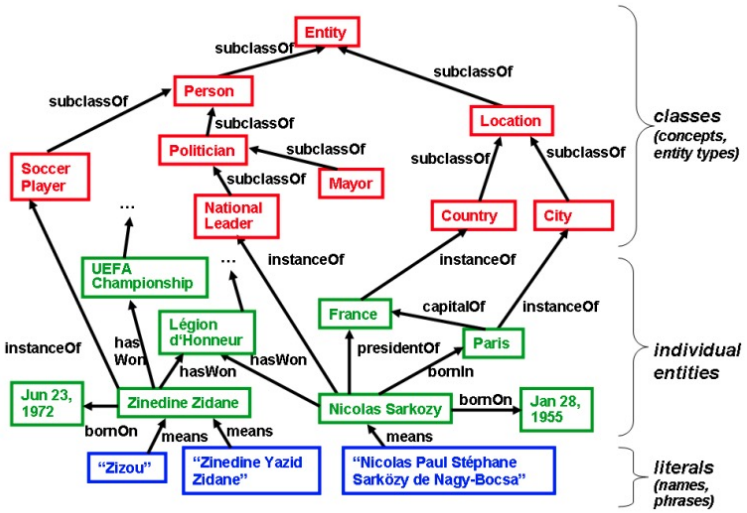


Figure 2.8: Excerpt of the YAGO Knowledge Base, from [25]

YAGO YAGO (Yet Another Great Ontology) is a complete knowledge graph automatically generated from Wikipedia. The project, which started in 2006 at the Max-Plank Institute, is based on the following observation: Wikipedia contains a large number of entities in many different domains but the category system is not well organized; on the other hand, WordNet contains a clean ontology of concepts but a very few entities as leafs of the hierarchy. Thus, other than extracting facts from Wikipedia similarly to what DBpedia did, the brilliant idea of YAGO was to map Wikipedia categories to WordNet concepts and obtain a complete knowledge graph with entities and types (an example is in Fig. 2.8). Differently from DBpedia, which extracts an RDF representation of Wikipedia's infoboxes, in YAGO only a few relations are represented: in this way they can keep an high-level of accuracy that has been evaluated around 95% [4].

	Freebase	**Wikidata**	**YAGO**	**DBpedia**
nodes	50 M	16 M	4.6 M	4.8 M
edges	3 B	66 M	26 M	176 M

Table 2.2: Number of relations and types for different open-world KGs

2.4.4 Automatically-built from the Web

Here we describe some projects that aim to extract knowledge at Web-scale. All of them are composed by an extraction phase, where a single or multiple extractors are used to harvest new facts from Web-scale sources; a reasoning phase, where the facts are aggregated and weighted in order to ensure correctness of the extractions.

Knowledge Vault The Knowledge Vault (KV)[8] is a research project by Google which attempts to build a very large knowledge graph automatically from the web. It relies on Freebase[2] as a reference schema (using the 4469 most common Freebase relations) and as "prior" to compute the probability of the facts extracted.

The framework consists in three steps (Fig. 2.9 provides a high level overview of the complete architecture). In the first step, facts are extracted from different Web sources:

- **text documents** (with distant supervision[26] techniques, more details in Sec. 3.1.3)
- **tabular data** (using entity linking approaches[14], more details in Sec. 3.2)
- **page structure** (training a classifier with lexicalized paths from the DOM tree);
- **human annotations** harvested from the HTML page[12].

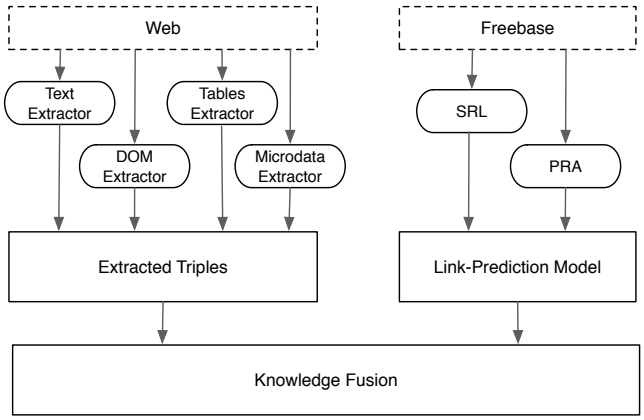


Figure 2.9: Architecture of Knowledge Vault (KV).

[12]They define a mapping from schema.org to Freebase schema for these different predicates

Second, a statistical relational learning model[27] is trained on Freebase to serve as a "prior" for computing the probability of (new) edges[13]. In particular, they use a combination of latent and observable models to predict links in a knowledge graph:

- a multi-layer perceptrons model (**MLP** [27]) is used to measure the latent semantic similarity of different relations;
- the Path Ranking Algorithm (**PRA** [28]) as a graph feature model to predict possible links;

They combine the two models, creating a single link prediction model that achieves an area under the ROC curve of 0.911 [8]. Finally, the confidence in the automatically extracted facts is evaluated using both the extraction scores and the link-prediction model. They also insert some knowledge fusion features such as the confidence of the extractor and the number of (de-duplicated) Web pages containing the evidence for the extraction. This is an example from [27] that shows the benefits of combining the prior with the extractors (i.e., Knowledge Fusion Layer in Figure 2.9). Consider an extracted triple corresponding to the following relation:

<Barry_Richter> <attended> <University_of_Wisconsin-Madison>

The extraction confidence for this triple (obtained by fusing multiple extraction techniques) is just 0.14, since it was based on the following two rather indirect statements:

> *In the fall of 1989, Richter accepted a scholarship to the University of Wisconsin, where he played for four years and earned numerous accolades...*

and:

> *The Polar Caps cause has been helped by the impact of important coaches such as Andringa, Byce and UW teammates Chris Tancill and Barry Richter.*

[13] Knowledge Vault extracts facts about entities that are already in Freebase.

However, from facts already in Freebase, Barry Richter was born and raised in Madison, Wisconsin. According to the prior model, people who were born and raised in a particular city often tend to study in the same city. This increases the prior belief that Richter went to school there, resulting in a final fused belief of 0.61. Combining both the models they are able to increase the number of high confidence triples from 100M (based on extractors alone) to 271M (based on extractors plus prior).

PROSPERA PROSPERA [7] is a project developed at Max-Planck Institute (MPI) which attempts to create an high-quality knowledge harvesting system from textual documents. The goal of PROSPERA is to offer a scalable framework that relies on lexico-syntactic patterns and logical reasoning, ensuring the quality and the validity of fact candidates. PROSPERA is built over SOFIE [29] and is organized to support distributed processing, which allow to scale to Web-size corpora. SOFIE starts with a seed-set for the target relations and learns lexico-syntactic patterns that are used to extract new facts that were not in the ontology. Then, using several hand-crafted consistency rules (e.g. checking the types of the subject/object of triples, check for functional constraints, etc.) they can construct Horn clauses from existent facts and evaluate the newly acquired knowledge as a (weighted) MaxSat problem.

ReadTheWeb/NELL The Never Ending Language Learning (NELL) project [6, 30] developed from Carnegie Mellon University (CMU) is a web-scale approach for extracting a knowledge graph from unstructured data, based on the principle of patterns and relations duality and use the redundancy of the information in the Web for learning mechanism. Differently from other approaches, it continuously learns new facts and corrects itself over time as it learns to better understand natural language. Indeed, it learns textual pattern from a large-scale corpus of web sites and exploits a coupled process which associates patterns, types and relation assertions and applies them to extract new entities and relations.

In order to keep an high level of accuracy they exploit reasoning which removes inconsistent triples and keeps them consistent. The system is still running today, con-

tinuously extending its knowledge base: one of the most recent version (i.e., the 945th iteration) contains roughly 2 million entities and 433,000 relations between those. The NELL ontology defines 285 classes and 425 relations.

Name	# Entity types	# Entity instances	# Relation types	# Confident facts (relation instances)
Knowledge Vault (KV)	1100	45M	4469	271M
DeepDive [32]	4	2.7M	34	7M[a]
NELL [8]	271	5.19M	306	0.435M[b]
PROSPERA [30]	11	N/A	14	0.1M
YAGO2 [19]	350,000	9.8M	100	4M[c]
Freebase [4]	1,500	40M	35,000	637M[d]
Knowledge Graph (KG)	1,500	570M	35,000	18,000M[e]

Figure 2.10: Comparison of automatic knowledge graphs construction compared to human curation ones, from [8]. The table reporst only confident facts (reliability>0.9).

2.5 Incompleteness of Knowledge Graphs

Large Knowledge graphs contain millions of entities, organized in hundreds to hundred thousands of semantic classes, and hundred millions of relational facts between them [21]. Despite this, all of them inevitable rely on the information provided by communities of people even directly, as in the case of Freebase and Wikidata, or through Wikipedia, which can be problematic in the long run. Take Wikipedia for instance, which is a perfect example of collaborative approach: it is maintained by thousands of contributors that update and review articles on a daily-basis, but recently it has been observed that its growth has been slowing down [31]. This is happening probably because with the increasing of dimensions the opportunity to contribute inevitably decreases. For the same reason, the growth of knowledge graphs that rely on collaborative approaches is bound to keep them incomplete especially for tail knowledge.

In Table 2.3 we provide some numbers to describe the order of incompleteness in Freebase and DBpedia for several relations in the domain of people. The percentages for DBpedia are obtained using the mapping-based properties (clean facts), while the numbers for Freebase come from [10]. As we can see from the table both the KGs are largely incomplete. The situation slightly improves when we consider only popular

entities (top-K ordered by number of outgoing links) where for some relations we obtain an acceptable coverage.

Relation	Percentage unknown			
	Freebase		DBpedia	
	All 3M	Top 100K	All 500K	Top 10K
profession	68%	24%	87%	36%
place of birth	71%	13%	63%	4%
nationality	75%	21%	92%	50%
education	91%	63%	94%	48%
spouses	92%	68%	94%	73%
parents	94%	77%	98%	85%
children	94%	80%	98%	83%
siblings	96%	83%	98%	84%
ethnicity	99%	86%	98%	83%

Table 2.3: Incompleteness of some relations in Freebase and DBpedia about entities in the domain of person. Left-hand part of the table is from [10].

The percentages obtained for DBpedia reflect the presence of those information in the info-boxes (the main sources of information for both DBpedia and YAGO). However, while such structured resources provide direct access to some information of the article, they have been shown to have low coverage: in 2008, 56%. of Wikipedia articles had no infobox, while in 2010 that fraction grew to 66% [14]. These numbers explain the reason why 63% of the entities classified as person are missing a place of birth (which instead is likely to be present in the vast majority of info-boxes about person) and the general high level of incompleteness. On the other hand, every Wikipedia article has natural language text expressing many human-readable facts that are missing from the info-boxes and, as we will show in this work, can be exploited for KG augmentation.

2.5.1 Heterogeneous Granularity

The goal of open-world KGs is to describe knowledge about entities of the real world and modelling it into a graph. As we saw, all those graphs (at least the *schema-based* ones) rely on a set of predefined classes and relations that can be used to classify entities

[14] https://en.wikipedia.org/wiki/Infobox

or label their connections. As reported in Table 2.4, the schema of such KGs, which are supposed to encapsulate knowledge about the world, is usually very rich.

	Freebase	**Wikidata**	**YAGO**	**DBpedia**
relations	38,000	1,600	77	2,800
types	27,000	23,000	488,000	753

Table 2.4: Size of the schema in popular KGs. All the numbers come from [32].

Despite large size, different schema are often very unbalanced. Indeed, it is not rare to find entities with very narrow types. For example, in YAGO, where the types are derived from Wikipedia categories, there is a type that applies to actresses who played as "Bond Girl" in 007 movies, while in Freebase there are types to describe Particle Physics elements, and in DBpedia there is the type "LightNovel", sub-type of "Novel".

The problem exists for relations as well, and the differences in terms of granularity depends on the domain of the information. For example, in the domain of geography, Freebase has a relation to list the capital cities of Brazilian states, or for sports, where some relations are created to keep statistics about athletes of specific sports. Other relations are more broad, encompassing more diverse entity pairs. The extreme example are the relations `<contains>` and `<containedBy>`, which are present in all the above KGs and apply to a broad swath of entity pairs. Also, in many cases, existent dependencies across relations are not defined in the underlying schema, as generally happens for the types. For example, there is no entailment between the relations `<ceo>` and `<employer>`, but the first is effectively subsumed into the second.

These unbalanced schema are an inevitable consequence of the process used to derive the graphs, which reflect the granularity used to express such information at the source. For example, The info-boxes of Wikipedia contain only a subset of the possible information about the entities, just as the contributors of collaborative KGs are interested only on particular aspects of the real world entities. In general, real world is not described following a specific schema, thus a perfect alignment is simply impossible.

Chapter 3

Related Works

In this chapter we provide a deeper description of the state-of-the-art on *knowledge* extraction from unstructured sources. We describe the main methodologies used to extract relations from text, which underlie several Web-scale automatic approaches for KGs automatic construction (described in Section 2.4.4). In particular, we focus a little bit on the advantages and the problems derived from introducing KGs as integral part of the supervision process. Finally, we describe some of the approaches used to annotate tables with information from the KG (i.e. understanding web tables) for the task of knowledge graphs augmentation.

3.1 Relation Extraction From Text

Harvesting knowledge from textual sources is crucial to build knowledge graphs with high coverage, but the problem has been faced since a long time. Indeed, the goal of early information extraction systems developed for Message Understanding Conference [1] was to use textual documents to fill the fields of specific *templates* (i.e. tables), for example the cause, the agent, the time and place of an event. Today, after thirty years (the first MUC has been organized in 1987), the same task has evolved in populating an incomplete knowledge graph (e.g., Freebase) with facts extracted from a large

[1]The Message Understanding Conferences (MUC) were financed by DARPA (Defense Advanced Research Projects Agency) to encourage research on information extraction methods.

corpus of text (e.g., Wikipedia) [2]. However, a strong problem for these relation extraction approaches is represented by the heterogeneity of KGs schemes (discussed also in Section 2.5.1). Since a standard set of relations and types is missing[3], the solutions proposed often depend on the specific schema adopted, introducing difficulties in the comparison of different approaches.

3.1.1 Supervised Approaches

Supervised approaches are currently the best performing methods for the task for relation extraction. The process is simple: they create a corpus where the sentences are annotated with relations, extract features at different levels, then train a model (SVM, MaxEnt, Deep Neural Network) and predict relations on a test-set.

Features are usually obtained from the NLP processing and can be POS-tags, NER types of the entities involved, lammas on the dependency parse, or simply n-grams of words between and around entities. Some approaches proposed to use latent-features such as words embeddings [33], which learn high-dimensional representations of labeled data and eliminate the need for feature engineering.

However, the problem is that these approaches are difficult to scale due to the amount of labeled training data required. For this reason, they can not be applied to the task of augmenting large knowledge graphs, which usually contain a large number of possible relations (see Table 2.4).

3.1.2 Pattern-based Bootstrapping Approaches

With the rapid growth of the Web came the first systems to extract information from Web sources. The pattern-based bootstrapping approaches, pioneered by DIPRE [34] and Snowball [35] can extract new facts for a target relation with a very minimal supervision. Indeed, they require only a few training facts of the target relation and a large collections of text documents. They start looking for a list of reliable lexico-syntactic

[2]Knowledge Base Population (KBP): https://nlp.stanford.edu/projects/kbp/
[3]Differently from the task of Named Entity Recognition or Part-of-Speech tagging, for example

patterns that are used to express the facts and so instantiate the target relation. They use those patterns to extract more pairs from text document, discovering new patterns and increasing the number of facts that can be extracted over multiple iterations. For example, DIPRE [34] who initially proposed the idea, starts with only 5 pairs of authors and their books, looking for new tuples of an hypotetical relation “writtenBy". The main problem of such bootstrapping approach is that it suffers from semantic drift, that occur when the patterns used lost their exact focus. The problem can be partially limited evaluating the quality of the new facts and the reliability of the new patterns during the expansion phase [36, 37]. Many works start from this principle to build their extraction systems, for example in Espresso [38], which uses *hypernymy* (is-a), *meronymy* (part-whole) relations, or KnowItAll [39], a Web-scale extraction tool, which relies on the redundancy offered in the Web (e.g. number of sources offering such information) to validate the tuples extracted and limitate semantic drift. The same idea is used in NELL (described in Section 2.4.4) which learns to extract tuples from text, but also tables and lists, focusing both on ontological relations (entity-class) and factual relations between entities. Finally, the principle has been used to increase the content of existing KGs from Wikipedia article texts as in [40, 41] or [29, 7] which use posterior logical reasoning to validate facts extracted.

3.1.3 Distant Supervision

In 1999 Craven and Kumlien presented an approach to train information extractors for textual documents in the biological domain. They rely on existing databases containing biological information, which provide “weakly" labeled documents that can be involved for training. They define this form of training data “weakly labeled" because each instance consists not of a precisely marked document, but instead of a fact to be extracted along with a document that may assert the fact [42]. The technique allows to create large annotated training set without manual supervision and, for this reason ten years later, thanks to the increasing availability of large knowledge graphs, it has been adapted in the task of relation extraction under the name of Distant Supervision

(DS) [26]. The paradigm consists in aligning all the facts already present in a KG with the text that mentions the entity pairs, obtaining very large 'weakly" annotated datasets. It combines the advantages of both supervised an unsupervised learning algorithms since it requires a set of canonical relations to use as labels for a supervised classifier (e.g. Naive Bayes, SVM, MaxEnt), but no manual effort is required to label training data.

Mintz et al. Approach [26] Many works that are based on distant supervision are built over [26]. It consists on a logistic multi-class classifier which uses features extracted from the sentences. In particular, it uses either lexical features (span of text between and around entities, part-of-speech tags, etc.), syntactic features (dependency path between the entities) and semantic features (NER types of the entities) obtaining a language-dependent classifier.

With this approach they can extract many facts that were missing from Freebase but the general accuracy is lower than for common supervised approaches due to incorrectly labelled training examples. Indeed, to train the classifier it assumes that all the sentences that mention a pair of entities express the relations that exist between them in the KG [26]. This inevitable introduces many false associations between sentences and relations, for example when there are multiple relations that hold between the entities but the sentence expresses only one of them (likely, if there are not overlaps or inclusions between the relations).

An other source of the noise is the **incompleteness of the KG**, which arises when the sentence expresses a non-existent relation (*schema incompleteness*) or the two entities are not related in the KG (*facts incompleteness*). Improving the automatic labelling approach has been the main focus of the following DS research [43, 44, 45, 9, 46].

Noise Reduction In order to relax the initial strong assumption a possible solution is by using a multi-instance model [43]. In this case the assumption is that at-least-one of the sentences mentioning a pair of entities might express the target relations and this has been modeled using a factor graph between the mentions (sentences) and the relations.

This model is further improved to support multiple relations expressed by different sentences mentioning a pair of entities (Multi-Instance Multi-Label, MIML) [44, 45]. Other ways to reduce the noise of the assignations using some heuristics [47] or relying on patterns correlation [46] (see noise reduction strategies [48])).

Modeling the Negative Evidence A problem of the data sets labeled with distant supervision is the absence of negative samples, which can not be obtained with incomplete knowledge graphs. Indeed, even large KGs do not contain facts that are explicitly false. The works introduced so far about distant supervised relation extraction model an extra ***negative*** **relation** which aggregates features extracted from the sentences where the entity pairs that are not related in the KG. In this way they would produce a massive amount of negative examples, containing many false positive due to the natural incompleteness of the KGs. In order to alleviate the problem they under-sample the negative instances: 1% is used in [26], 10% in [43] and 5% in [45]: this number is choose based on the best results in development for all models. Other works manually select relation-specific cases [49]. A different approach to reduce the false positives in the negative examples is adopted in [9] introducing a further expectation maximization training step on the unlabeled sentences. However, there is not a way to obtain clean negative examples and all these approaches are based on heuristics which aim to estimate them[4].

3.1.4 Open Relation Extraction (OpenRE)

With the massive amount of heterogeneous text documents available from the Web it is difficult to obtain labelled data necessary to train extractors for every single relation. In this scenario, unsupervised methods which do not require a fixed schema became very popular for exploring and harvesting knowledge from text creating the paradigm of Open Relation Extraction (OpenRE). Existing approaches can be grouped according to the level of sophistication of the NLP techniques they rely upon: (1) shallow methods, (2) dependency parsing and (3) semantic role labelling (SRL).

[4]DeepDive's guidelines on how to obtain negative evidence for RE: http://deepdive.stanford.edu/generating_negative_examples

Shallow OpenRE Shallow methods annotate the sentences with part-of-speech (POS) tags and then identify relations by matching patterns over such tags. The most representative examples are TextRunner [50] and its successor ReVerb [51], which are based on the idea that most relations are expressed using few syntactic patterns (e.g. "verb", "verb+preposition" and "verb+noun+preposition"). A similar approach is used by SONEX [52], which extends ReVerb by detecting more rich patterns.

OpenRE via Dependency Parsing Dependency parsing allow to identify whole sub-trees connecting two entities, that are the arguments of a relation predicate. The most representative OpenRE approach in this category are PATTY [53] and Ollie [54]. The former extracts textual patterns from sentences by looking at the shortest path that connects the two named entities in the dependency graph. The latter also extracts relations between two entities applying pattern templates over the dependency sub-tree containing pairs of entities. Those pattern templates are learned automatically from a large training set that is bootstrapped from high confidence extractions from ReVerb.

OpenRE via Semantic Role Labelling Semantic role labeling (SRL) systems aim to recover the semantic in a sentence, expressing the exact role that the arguments can take in the event, defined by the predicate. The approach is based on lexicon that describe predicates and their corresponding roles: for example PropBank[5], where roles are specific to individual verbs, or FrameNet[6], which introduce the concept of *frame*.

Overall, OpenRE methods do not rely on canonical relations from a specific schema thus can not be used to augment an existent knowledge graph. However, since they are all unsupervised, they could be used to extract verb-centric relations on large corpora such as the Web. In some cases their computation cost become high, for example when they require the dependency or semantic parsing, and it does not always pays off with higher effectiveness [55].

[5] https://propbank.github.io/
[6] https://framenet.icsi.berkeley.edu

3.2 Web Tables Understanding

An other potential source of information that can be used in augmenting the content of existent Knowledge Graph is represented by HTML tables, which are massively used in the Web. According to [13], in 2009 there were a total of 14 billion raw HTML tables on the Web and, among those, 154 million high-quality relational-style tables contained potential useful information about entities. Indeed, even if the vast majority of them are used for formatting purposes, the Web offers enough tables to consider them one of the richest available data sources. Given the wide range of the topics covered, tables have been involved in many processes for automatically create Knowledge Graphs [8, 6].

Unfortunately, relation extraction techniques developed for text can not be directly applied to extract facts from tables, because the relation between two entities is not mentioned but can be hidden in the structure of the table or expressed by using maximum some words (i.e. header of the columns). Also, since Web tables are usually formatted for human consumption, there are many challenges that need to be solved to understand the schema and the format before to extract knowledge from them.

Hurst [56] divides the problem of processing Web tables into four sub-tasks: (1) tables have to be located and extracted from the Web documents; (2) tables should be parsed and normalized into known structures; (3) tables used for page layout need to be discarded and, finally, (4) the tables could be used for recovering their semantic and process their content.

In this part of the thesis, we initially describe some of the approaches used to perform the first three sub-tasks and then we focus on the fourth sub-task, giving an overview of state-of-the-art approaches proposed for recovering the semantic of Web tables with the goal of augmenting an existent Knowledge Graph.

3.2.1 Obtaining Relational Web Tables

The Web can be considered as a corpus of unstructured documents. Many of those documents contain potential information that are often expressed in the form of surface HTML tables. As done in many works [13, 57, 58], those tables can be detected from

the documents using the tag `<table>`. A common problem is that such HTML tag is often used for structuring the content of the page (e.g. visualize HTML forms, calendar, etc.), instead of containing relational information between entities or values. For example, Web tables are improperly used to (*a*) render **HTML forms** with multiple fields that users can fill, (*b*) to **structure the layout** of the whole/part of a page, or (*c*) create horizontal and vertical **navigational bars**. Differentiating weather a table is used to aggregate information or other purposes is difficult without understanding the data exposed. This task has been faced using machine learning solutions [59, 13, 60]. Usually, classifiers are trained on a samples of hand-labeled tables and the features exploited consist in the table layout (e.g., # rows, # columns, average cell length) and content consistency (e.g., # columns containing non-string basic data types, such as integer and date) [59, 57]. Based on [13], using such classifier on a large collection of tables harvested from the Web allows to filter out 98.9% of tables that are used for layout purposes. Fortunately, the remaining 1.1% is still a big number, since it contains 154 millions of tables that can be used to extract useful knowledge.

The Case of Wikipedia The set of Wikipedia tables is a small part of the Web tables, consisting on 1.4 million of tables [61]. A lot of works use them, as it is believed that almost every table is relational, well edited and contain valuable data, making it an ideal table corpus to study. In this case, tables are generally extracted from the Wiki Markup using the `wikitable` class [61, 62] and than parsed to obtain the relative HTML format [63]. However, even in the case of Wikipedia tables, there are still many problems required to be solved due to the variety of formats used to express information.

3.2.2 Kinds of Relational Tables

The information contained in Web tables may be exposed in many different ways. The variation in formatting is one of the key reasons that make table understanding an hard task. Indeed, in order to extract the knowledge encoded in those sources, one has to understand the underlying schema. In [64], a fine-grained table schema taxonomy has

been realized through a census of HTML tables on a large crawl of the Web. In Table 3.1 we reported a simplification of that taxonomy.

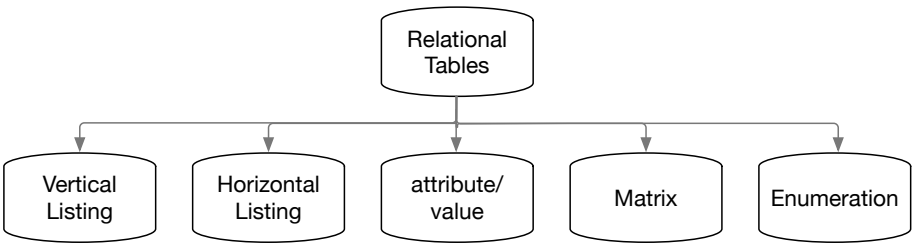


Figure 3.1: Taxonomy of Web Tables, deirved from [64]

Even if we consider only tables that contain high-quality relational data they are not formatted by only rows and columns, but can be of very different types. The main kind of tables are:

- **vertical listings**: listings are used to show one or more attributes for a set of similar entities. For example, a table that lists the capital city and population for several countries. When they are vertical there is a single entry per line. These tables are ideal to be used with the goal of augmenting a KG because they contain all the information needed to extract triples from them. One of the challenges is the identification of the subject column, and the columns that list the attributes. Often, the subject is the first column, but other more sophisticated solutions have been proposed [13, 16, 65].

- **horizontal listings**: in this case the listing is horizontal, namely there is one entry per column. This kind of tables are not often used for the task of augmenting a KG because they are widely used for comparison of different items, e.g. products.

- **attribute/value**: these tables are similar to listings but the subject is not contained in the table. This format is used to express some facts about an external entity, usually the subject of the page [64]. This format is also called *specifications* [66]. For example, a table containing specifications of a digital camera,

where the table does not contain the actual camera name. Wikipedia info-boxes might be a good examples of this kind of tables.

- **matrix**: these tables contain information that are meaningful for a combination of entities. The value is expressed in the cell at the junction of a row and a column. For example, a table giving the number of traffic accidents per month (rows) and per state (columns) is of type matrix. These tables are frequently used to provide temporal information, e.g. timeline, calendar, etc.

- **enumeration**: these are particular tables containing a list of similar entities, generally with a single header [67]. For example, a list of the best museums in Rome might consist on a list of names.

Other great source of relational data consists in spreadsheets [68, 69, 70], which are largely used to contain valuable data. Eventually, tabular data can be obtained from the very large amount of data-intensive sources offering data that can be harvested exploiting the regularity of the DOM trees [71, 72]. All these kind of tabular data could be considered a small “database" containing useful information, but the problem is the absence of a uniform schema: header rows exist in few cases, and even when they do, the attribute names are typically noisy.

3.2.3 Recovering the Semantic for Search

Without knowing the inherent semantic of a table, it is very difficult to leverage its content. Indeed, the meaning of each table is rarely explicit from the table itself but is interpreted by humans who look at it. The task of Web tables understanding consists in adding semantic annotations to tables, which can be used for example to facilitate operations such as web-search and finding related tables [14]. Clearly, the process depends on the kind of table involved. For the sake of simplicity, we discuss the main methodologies used to annotate vertical listing tables. Indeed, this kind of tables are extremely common on the Web, and store exhaustive information about pair of entities

which can be of interest for a user. For example, in Fig. 3.2 there is a vertical listing that describe actors and filming locations of the movies listed in the first column.

Film	Main Actor	Location
Dr. No	Sean Connery	Jamaica
From Russia With Love	Sean Connery	Yugoslavia
The World Is Not Enough	Pierce Brosnan	Kazakhstan
Moonraker	Roger Moore	Brazil

Figure 3.2: Example of a vertical listing table.

With only the names of the entities a user that is interested in “filming location of James Bond movies" does not have the possibility to retrieve the table. For this reason, adding information on the entities described (*movies* or *007 movies*, in the first column), the types of the entities (*actors*, *person* and *locations* or more specifically *countries*) and the relationships among them can provide a bettere description of the table.

A first solution that has been proposed aimed to facilitate web-search [14]. They annotate columns with keywords reflecting the classes of the entities from an “is-a” database (if a sufficient number of the values in the column are identified with that keyword in the database) and relationship between the subject column and other columns with *key-phrases* frequently occurring with the entities in the table. In both the cases they mine classes and key-phrases using lexico-syntactic patterns (e.g. Hearst patterns [73] for classes) on textual documents from the Web and assign them using a maximum likelihood inference model.

This work is an example of how text on the Web can be used to recover the semantics of structured data on the Web. The annotations provide a good impact in improving table search but, on the other hand, they are only textual keywords, which are not part of a predefined set of semantic annotations. This represents the main drawback of this approach, which can be improved relying on annotations from a knowledge graph.

3.2.4 Recovering the Semantic with a KG

Annotating tables with knowledge graph information allow to leverage their content in many different ways. Indeed, other than search, tables could be used to augment existent knowledge graphs. In this case some more work is necessary: (1) the content of each cell has to be linked to the relative entity in the KG (if exists), (2) columns need to be annotated with KG types describing the entities and (3) pairs of columns are annotated with KG relations that hold over the respective entities. Performing these three tasks on a given table allows to create triples with entities and relations described, which can be used to augment existing KGs.

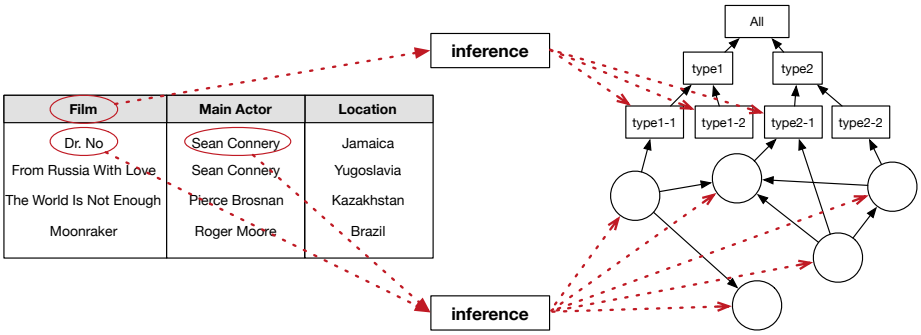


Figure 3.3: Recovering semantic of a table with a KG.

The first step is the tricky one, while the others two might be derived from it. It requires to link cells to entities, essentially matching their textual content against entity labels. This step is not trivial, especially in the context we consider here where there is little or no text connecting the named entities. Indeed, the task is similar to the entity linking problem (see [74] for a survey) where entities mentioned in textual document need to be linked with node of the graph. In that case features extracted from text can help in describing the context of the document and improve disambiguation. Many possible solutions have been proposed, with the aim of building a representative context that can be used for disambiguate entities, for example using the main components of the table (i.e. cell content, header) but also external features such as the title of the table, the captions, and the text that surrounds the table.

The first two solutions are similar: given a table, they identify the "entity column" (i.e. the column that contain the entities described in the table) and try to match each row with an entity of the KG. Both the solutions consist in iterative algorithms that match and refine possible entities considering the coherence of their types. Suppose for example that they correctly match some title in the first column of Table 3.3 with the movies and some other rows with the titles of the albums containing the respective soundtracks or even something more different. A refining step would consider the types of the matched entities to drive the matching in the second iteration.

TableMiner [65] The first step consists in identifying the "subject column" of the table. Then, the process is based on two steps: (1) (*LEARNING*) a first pass performs column classification and entity disambiguation using an incremental and mutually recursive approach. For each column of the table they annotate cells with KG entities (they starts from unambiguous cells) using the features discussed below and the relative type as the type of the column; (2) (*UPDATE*) in a second pass they revise the annotations by enforcing interdependence between columns, and between the types of the columns and the relative candidate entities for the cells.

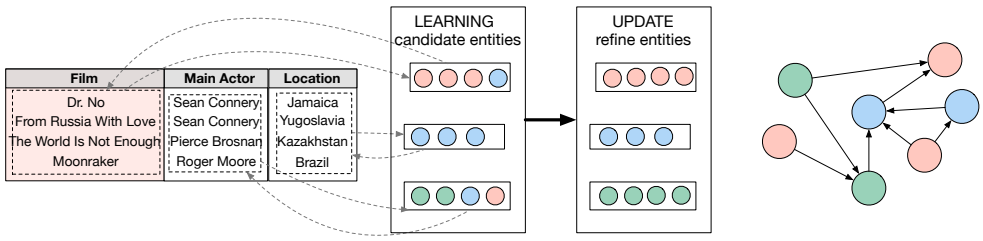


Figure 3.4: Two-steps approach of TableMiner. On the right side there is the KG.

Finally a relation enumeration step discovers binary relations between the "subject column" and other entity columns. Differently from other works, TableMiner uses features from both in-table (approximate matching of cell content) and out-table context such as captions, webpage title, surrounding paragraphs, semantic markups inserted within webpages (if any). They experiment the system with Freebase as a reference KG.

T2K Framework [16] A similar idea is to consider a table as a collection of similar entities, modelling each row as a set of (possibly multi-valued) attributes describing a single entity in the KG. The method assigns a global type to each Web table (e.g. [Film] in the previous example), a KG entity to each row and a property to each column (if possible). In this case the algorithm iteratively refines the assignments relying on a mutual influence between candidate entity types and schema-level matching (one is used to weight the similarities of the other).

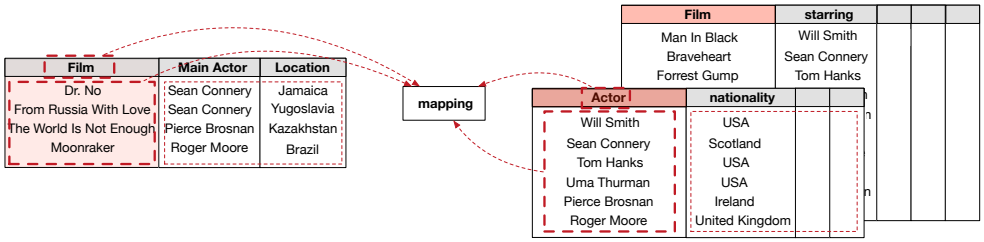


Figure 3.5: Data integration approach of T2K. On the right side there is the collection of tables that represent the KG.

In practice, the KG is viewed as a collection of tables, in which each table lists all the entities and the properties of a specific type. For example, the table of the type [Film] contains all the entities of such type, described with all their properties in the columns (e.g. starring, director, producer, etc.). The method can be considered a data integration approach: given a table, it has to find the most similar table of the KG, linking the records and, potentially, filling missing values.

Munoz et al. Approach [62] They propose a framework to extract DBpedia triples from tables in Wikipedia. Differently from other works described here, they skip the entity linking step since Wikipedia tables already contain delimited and unambiguous entities (i.e. wiki-links). The approach is based on the assumption that if two entities co-occurs in a Wikipedia table, it is likely that they should share an edge in the knowledge graph. For this reason, they extract a large set of candidates from all the tables in Wikipedia, using all possible relations that hold between at least one pair of entities in two columns. Then, based on a labeled subset of that extraction, they apply

classification using various features to identify those relations that should actually hold in the knowledge graph.

Limaye et al. Approach [15] An earlier work in this area uses a graphical model to describe the dependencies among all the components of the table, collectively annotating: cells with entity identifiers, columns with schema types and pairs of columns with KG relations. They use YAGO as a reference KG.

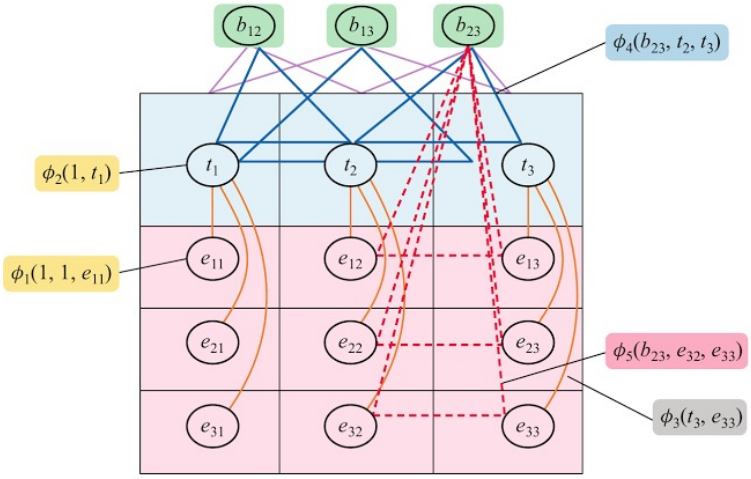


Figure 3.6: Graphical model for annotating a 3x3 Web table, from [15]

More specifically, the probabilistic graphical model in Fig. 3.6 relies on a number of interrelated random variables:

- $\phi_1(r, c, e_{rc})$ expresses the matching between the cell $(r,\ c)$ matches the entity e_{rc}
- $\phi_2(c, t_c)$ expresses weather the header text for column c describes the type t_c
- $\phi_3(t_c, e_{rc})$ expresses the coherence between entity e_{rc} and type t_c
- $\phi_4(b_{cc'}, t_c, t_{c'})$ expresses weather the relation $b_{cc'}$ between columns c and c' is compatible with types t_c and $t_{c'}$
- $\phi_4(b_{cc'}, e_{rc}, e_{rc'})$ expresses weather the relation $b_{cc'}$ between the cells (r, c) and (r, c') is compatible with the entities e_{rc} and $e_{rc'}$

The model is trained to learn the weight used to combine those features with the goal to maximize the the joint probability of their assignments.

Chapter 4

Accurate Fact Harvesting from Wikipedia Text

Many approaches have been introduced recently to automatically create or augment knowledge graphs with facts extracted from Wikipedia as done for example for YAGO [4] and DBpedia [5]. In both projects, different techniques are used to extract facts from the *infoboxes* and the article *category* hierarchy. Although these structured parts are valuable, they represent only a fraction of the actual information expressed in the articles. Indeed, every Wikipedia article has natural language text expressing many human-readable facts about the entity being described, which can be exploited for KG augmentation.

In this chapter we quantify the number of highly accurate facts that can be harvested with high precision from the text of Wikipedia. In particular, we describe **Lector**, an extraction system that is able to harvest facts from the text of Wikipedia articles and augment the content of a reference KG. The approach relies on the duality between patterns and relations, a well-know principle in Information Extraction (IE) [34, 35]. The tool is able to select the most common *patterns* that represent instances of KG relations in text and, applying them again, it collects many facts that are ready to be harvested from the articles.

In its preliminary version [18], which uses Freebase as reference KG, Lector extracts more than 230K new facts for 12 relations in the domain of people, augmenting the content of some of them by more than 10% with facts whose accuracy are over 0.95. As we show, many of these facts are missing from the info-boxes and the KGs derived from them (other than Freebase), indicating our approach is complementary to the state-of-the-art of automatic KGs construction.

For this reason, we propose **Lector+** [19] in which we adapt the model to use DBpedia and scale to a larger number of relations. We experiment the tool from several language-editions of Wikipedia (namely English, Italian, Spanish, German and French). As we show, the system extracts more than 2M of new facts with an accuracy estimated around 0.8, for a large number of DBpedia relations.

In this chapter we describe Lector+ and its improvements over the initial version, which let the tool to won the **2017 DBpedia TextExt Challenge**[1].

4.1 Overview of Lector Approach

Lector relies on the well-known principle of patterns and relations duality [34, 35, 36]. The patterns are represented by *typed-phrases*, i.e. a concatenation of entity types and fragments of text that are commonly used to describe instances of KG relations in natural language. For example, the sentence "*Ben Ripley is a graduate of Stanford University*" is an evidence that the subject entity "*Ben Ripley*" has studied at "*Stanford University*" which is an instance of the DBpedia relation <dbo:education>.

From that sentence we can extract the typed-phrase "[Person] *is a graduate of* [University]" which represents a *pattern* that is frequently used to describe instances of such relation in Wikipedia articles.

In order to extract facts that are not present in DBpedia, Lector follows a two-phase approach. First, it processes all the Wikipedia articles to extract the most common typed-phrases used to express facts (i.e. instances of relations) already known in DBpe-

[1] http://wiki.dbpedia.org/textext

dia. For example, the following are other very common typed-phrases used to describe instances of the relation <dbo:education> over the text of Wikipedia articles:

"[Person] *attended* [University]"
"[Person] *graduated from* [University]"
"[Person] *entered* [University]"
"[Person] *was educated at* [University]"
"[Person] *studied at* [University]"
"[Person] *then attended* [University]"
"[Person] *studied law at the* [University]"
"[Person] *enrolled at* [University]"
...

After collecting all the typed-phrases for a large number of relations, the system looks for their possible occurrences between pairs of entities that are not yet linked in DBpedia, yielding new facts for the relations.

To obtain facts with high accuracy, the main challenge of Lector consists in associating only the typed-phrases that effectively represent DBpedia relations and discard other generic or ambiguous expressions. For example, the following typed-phrases do not describe specifically the relation <dbo:education> even if they can be found often between entities linked with that relation in DBpedia:

"[Person] *went to* [University]"
"[Person] *at* [University]"
"[Person] *from the* [University]"
"[Person] *returned to* [University]"

Some of these typed-phrases appear very frequently between pair of non-related entities and they do not express any particular relation in DBpedia. Thus, extracting new facts from them would be very risky for the general accuracy of the KG.

4.1.1 Extraction Tool

Lector starts from a Wikipedia dump and produces, as output, a list with all the new facts that can be added to DBpedia. The whole process can be summarized in four steps. We briefly describe them with reference to the architecture of the tool, in Fig. 4.1.

1. **Article Parser**: which reads the input dump provided using the WikiMarkup formatting language and extracts the clean text from all the articles describing DBpedia entities. We adapt this step for all the experimented language-editions. More details are given in Section 4.2.

2. **Entity Detection**: which detects all the entities from the articles linking them to the relative DBpedia resource. For example, consider the following fragment of <Steve_Jobs> article:

   > Jobs founded Apple on April 1, 1976.
   > He met Steve Wozniak and Ronald Wayne in 1970.

   we detect all the entities:

   > <Steve_Jobs> founded <Apple_Inc.> on April 1, 1976.
   > <Steve_Jobs> met <Steve_Wozniak> and <Ronald_Wayne> in 1970.

   In this step we take advantage of the entities that are already annotated through the wiki-links (i.e., the HTML links used to highlight concepts and link them to Wikipedia articles) on the page but we also adopt other techniques to detect other kind of entities. This step is described in Section 4.3.

3. **Typed-phrase Labelling**: which extracts the text between each pair of consecutive entities and convert it to a typed-phrase. Continuing with the previous example, we extract the following typed-phrases:

   <Steve_Jobs> founded <Apple_Inc.> ⟶ [Person] *founded* [Company]
   <Steve_Jobs> met <Steve_Wozniak> ⟶ [Person] *met* [Person]
   <Steve_Wozniak> and <Ronald_Wayne> ⟶ [Person] *and* [Person]

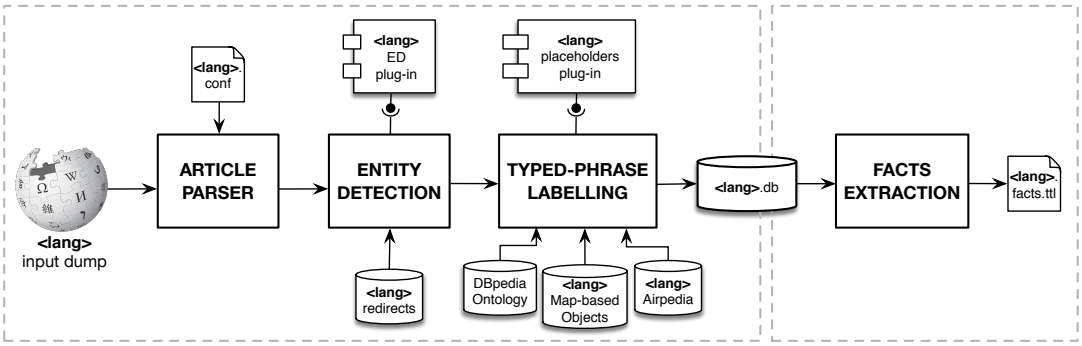


Figure 4.1: Complete architecture of Lector+.

This step is described in Section 4.4 and the statistical method used to associate typed-phrases to DBpedia relations is described in Section 4.4.2.

4. **Facts Extraction**: which extracts new facts that can be directly added to DBpedia by exploiting, from one side, the typed-phrases that have been associated to DBpedia relations and, on the other side, their instances between entities that are not related in the graph.

In order to keep a flexible architecture we created a pipeline that uses language-agnostic components. In this way, we can adapt the process to different languages with a minimum effort. We applied the tool on five different language-edition of June 2017 Wikipedia (English, Italian, Spanish, German and French) extracting almost 2 million of facts for a large number of relations in DBpedia.

The tool is available[2] and can be executed on more recent versions of Wikipedia dump. Each components could be improved modifying the customized plug-ins to perform language-specific tasks, as showed in the architecture in Fig. 4.1.

4.2 Article Parser

The first step consists on a parser which read the WikiMarkup formatted pages and extract the full text of the articles. Here we perform an initial filtering of the articles,

[2] https://gitlab.com/Rm3UofA/Lector/Code

	en	de	fr	it	es
factual articles	5.12	2.04	2.40	1.28	1.32
hub-pages					
↳ category	1.63	0.31	–	0.31	0.34
↳ list	0.09	0.05	0.01	–	–
↳ disambiguation	0.26	0.02	0.99	0.10	0.05
meta-pages					
↳ redirect	8.26	1.42	1.49	0.67	1.70
↳ template	0.48	0.06	–	0.13	0.02
↳ portal	1.06	0.08	0.05	0.13	0.04
other	0.93	–	0.05	–	–

Table 4.1: Statistics of the article parser, with the counts (in million of pages) obtained for each category of article. We use only factual articles for the further steps.

trying to keep only the ones containing facts (i.e. factual articles) and discarding meta-articles such as template pages, discussion pages, project pages, etc. This task can be partially done by checking the namespace[3] of the pages (for example for templates or projects) but, in most of the cases we need to examine some cues from the markup, for example to differentiate hub pages from normal articles[4], penalizing a bit the accuracy of the filtering. As reported by the statistics in Table 4.1 this step allows to eliminate most of the pages from the original dump. From now on, we use only the factual articles.

For each article, we first extract all the original wiki-links and then remove all the contents that surround the text. As we will describe in the next section, we exploit all the wiki-links of the page for the entity detection. We then eliminate noisy markup tags and, with the help of a small configuration file customized for each language, we recognize and eliminate footer sections such as “external reading", “notes", “bibliography", etc. and the structured parts of the articles such as info-box, tables, lists, images and other media that are not necessary in our approach.

The article parsing results in a collection of structured articles, including only the sections and the paragraphs with clean textual content[5] and a set of original wiki-links associated with their anchor texts.

[3] https://en.wikipedia.org/wiki/Wikipedia:Namespace

[4] We filter out pages like https://en.wikipedia.org/wiki/Forrest_Gump_(disambiguation) or https://en.wikipedia.org/wiki/List_of_Italian_dishes

[5] The output is similar to the NIF dataset: http://wiki.dbpedia.org/downloads-2016-10

4.3 Entity Detection

The second step consists in detecting all the DBpedia entities that are mentioned in the text of the articles. For general Web documents, this task is known as *entity linking* [74] and sophisticated techniques are used to disambiguate named-entities and link them to the KG by relying on several features extracted from their context [75, 76]. However, in the case of Wikipedia articles the situation can be different.

Most of the articles are known to describe a specific named-entity, that we call **primary entity (PE)**. For example, the entity <Steve_Jobs> is the primary entity in the article about him (i.e. the article identified by /wiki/Steve_Jobs). Those PEs are widely described in their article but their mentions are often subsumed in the text or use alternative expressions such as pronouns, aliases, abbreviations, etc. which are out-of-reach from common entity linking tools.

On the other hand, each article contains many **secondary entities (SE)**: for example, the entity <Steve_Wozniak> is a secondary entity in the article about Steve Jobs. In this case, those entities are usually annotated with a wiki-link to their relative page. However, as suggested by the Wikipedia guidelines[6], in order to increase the readability of the pages many secondary entities are annotated only in their first occurrence. To give an example, at the time of writing, only 7 over 30 instances of <Steve_Wozniak> are annotated in the article about Steve Jobs.

The entity detection approach adopted by Lector aims to capture most of PEs and SEs instances from the articles. For the former, we take advantage from its encyclopaedic style and create some co-reference heuristics which allow to discover most of instances that are not explicitly mentioned in the text. For the latter, we rely on the original wiki-links on the page and augment their instances by harvesting most of the missing wiki-links over the article. As we will see, our approach is quite robust, especially for some specific category of articles.

[6] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style

4.3.1 Detection of Primary Entity

Our approach consists in finding, for each article, most of the expressions that are used to refer the PE in its article (i.e. *lexicalizations*). For example, some of the lexicalizations are based on the information that we can capture from the abstract of the article:

- **title** – after removing further specifications into parenthesis;
- **alias** – are the alternative names used for the primary entity, which are usually expressed with bold names in the first sentence. For example, in the sentence "Microsoft Corporation, commonly referred as Microsoft or MS ... " we capture both "Microsoft" and "MS" as possible aliases;
- **lexical class** – a widely used form to refer the PE is represented by its class, which can be extracted by using simple finite-state machine on the first sentence (as in Fig. 4.2). Capturing the class C ="*company*" from the sentence "*Apple is an American multinational company head-quartered in [...]*" let us to harvest many other instances of <Apple_Inc.> in its article looking for the expression "*the C*". Indeed, we avoid many false positives adding the token "*the*", which is generally used to mark a noun as indicating the best-known in the article.

We harvest lexicalizations from the articles without using any external resource produced by similar approaches[7]. We introduce two other techniques to extract other lexicalizations of the primary entity based on their frequency over the whole article[8]:

- **part of the title** – we extract a part of the title if it is very frequent over the article. Indeed, we look for frequent longest common sub-string between the title and all the sentences. It allows to detect the last name of person or the main part of the title (e.g. "Thames" in the article of <River_Thames>).
- **pronoun** – the technique is similar to the previous one. In this case we extract a third-person pronoun if it is very frequent over the article, compared to the others.

[7] For example, the Lexicalization data-set (http://wiki.dbpedia.org/lexicalizations), included into DBpedia Spotlight, provides a list of names used to refer the entities including redirects and anchor texts.

[8] For the techniques that involve statistics on the whole article we set a threshold to 0.5.

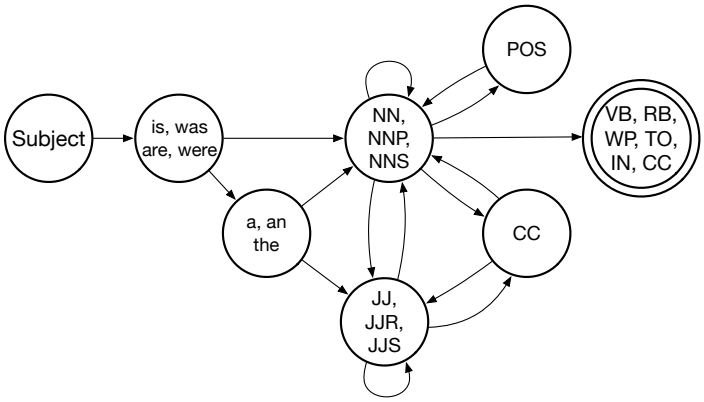


Figure 4.2: Possible English finite-state machine used to find the entity-type in the first sentence, expressed using Penn Treebank POS tags. The types are all the nouns obtained before to enter the final state.

> Since “it” is a widely used token in English, we use only third-person pronouns referring to people (“he”/“she”) avoiding many false positives.

For each article we apply all these techniques to create a list of PE lexicalizations. Then, after ordering them by specificity (i.e. length), we look for their occurrence in the text: when they occur we replace that span of text with a wiki-link pointing to the PE of the article. Also, in order to avoid partial or wrong replacements, we check that the span of text that has to be replaced is not surrounded by any proper nouns. In this way we can reduce the number of false positives that can occur for example when we replace the last name of people that belong to the same family of the primary entity. For example, if we associate the lexicalization “Obama" to the entity <Barack_Obama> we can not match the same text against the span of text “Michelle Obama". Only in a second moment we use an existent entity linking tool (namely DBpedia Spotlight [77]) to harvest possible remaining instances or other entities.

Evaluation While these techniques are heuristic, they work well in Wikipedia because of its editorial guidelines. Also, we observe that the primary entity is expressed differently on different kind of article. For this reason, we evaluate our approach on different categories such as articles about Person, Location, Orgnization or Creative

(articles)	Person (1.1 M)		Place (788 K)		Organization (260 K)		Creative Work (436 K)	
	inst.	acc.	inst.	acc.	inst.	acc.	inst.	acc.
PE	**17 M**		**3.8 M**		**2.5 M**		**2.2 M**	
↳ title	7 %	.96 ± .02	39 %	.97 ± .02	40 %	.98 ± .01	10 %	.77 ± .06
↳ aliases	1 %	.95 ± .03	13 %	.96 ± .02	10 %	.96 ± .02	1 %	.78 ± .06
↳ lex. class	1 %	.65 ± .07	34 %	.92 ± .04	30 %	.98 ± .01	45 %	.98 ± .01
↳ p, of title	38 %	.98 ± .01	11 %	.93 ± .03	14 %	.90 ± .04	15 %	.22 ± .06
↳ pronoun	52 %	.93 ± .03	1 %	–	1 %	–	27 %	–
↳ spotlight	1 %	.84 ± .05	2 %	.78 ± .06	5 %	.81 ± .06	2 %	.27 ± .07

Table 4.2: Effectiveness and relative accuracy of PEs detection techniques on different categories of articles. Each accuracy is estimated over a random sample of 100 instances.

Work, based on the DBpedia type of the PE described. In particular, we can associate ∼2.6M of articles with a category (the other 50% of articles describe entities with missing types or out of these four categories). From these, we manually evaluate a sample of 100 random instances of PEs detected with each technique in each category, thus we estimate the accuracy using Wilson score interval for $\alpha = 5\%$.

From the numbers in Table 4.2 it is evident that in the articles describing persons the primary entity is often mentioned using the last name (part of the title) or the third-person pronoun; in both cases our technique obtains a very high accuracy. However, using the third-person pronoun is inevitably wrong with other category of articles such as organizations or locations, but the small number of false positives in these articles confirms their different nature. A different situation happens in the articles about creative works such as movies, books, fictional characters, etc. In this case, most of the 27% of the instances are false positives due to person that are described in the article but are not the primary entities. For example, the plots of movies and books often mention the main character using the person pronoun. An other problem that arise with such category of articles is that the title often contains mentions to other entities that are described in the text. This problem clearly affects negatively the accuracy of the techniques that are based on the title, for example the part of the title, but also the DBpedia Spotlight. For example, the article about the movie “Alice in Wonderland” contains many mention of “Alice” that should refer to the character and not to the

primary entity. Fortunately, we are able to harvest correctly most of the PEs in this category of articles relying on the lexical class (45%). The same technique works very well even in articles about locations (34% with an accuracy around .92) or organizations (30% with an accuracy around .98). In these two categories even the techniques based on the aliases are very useful, albeit for different reasons: for some locations that title can be too specific (it contains the region where it is located, e.g. “Lehigh, Alberta") making it difficult to occur in the text, while the alias usually expresses the name that occur in the text (i.e. “Lehigh"); on the other hand, in some articles describing organizations the aliases usually express acronyms that are widely used to refer the PE: for example “University of California, Los Angeles" is mentioned more than 300 times as “UCLA" in its article.

Overall, we are able to detect from ∼5 PEs per article (in the case of articles about locations) to ∼15 PEs per article (in the case of articles describing persons). After using all the lexicalizations, DBpedia Spotlight, which associates token to entities based on a probability estimated on the anchor texts of the whole Wikipedia corpus, does not contribute too much. This is primarily because we run it after all the other techniques, thus most of the PEs mentioned with the title or aliases are already detected. However, such tool can not identifies PEs expressed with the lexical classes or pronouns, for which sophisticated co-reference approaches are necessary.

4.3.2 Detection of Secondary Entities

For the secondary entities we apply a similar technique which harvests most of the missing wiki-links over the article. Indeed, in this case we consider as possible lexicalizations the anchor text of the existent wiki-links (not only in text but also in info-box and tables). Using this technique we are able to increase the instances of the secondary entities by 40%, that we would miss considering only the original wiki-links. We evaluate 100 random SEs detected from articles of all the categories, obtaining an accuracy around 0.88 ± 0.04.

4.4 Typed-phrases Labelling

Having detected all the entities in the articles, we can now extract the *patterns* and associate them to possible relations in DBpedia. For this reason, we briefly describe how we obtain typed-phrases from the sentences and then we discuss the statistical approach used by Lector to associate them to the correct DBpedia relations, trying to discard the ones that do not describe any relation.

4.4.1 Extract Typed-phrases from Sentences

To extract typed-phrases we focus exclusively on the sentences that contain at least two entities. For each pair of two consecutive entities we create a typed-phrase by generalizing the text between them (essentially removing numerical values) and filtering out generic expressions (i.e. conjunctions); thus we concatenate it with the fine-grained types of the two entities in DBpedia.

Filter Out Generic Phrases In order to consider only phrases that are likely to express a relation between two entities in the first version of Lector [18] we removed the spans of text that are not coherent with well-known *relational* patterns. However, that filter was very strict discarding many useful fragments often because of the presence of further specifications, like numbers or dates for example. In order to ensure more flexibility but also to extend the approach to other languages, in Lector+ we keep every span of text but the conjunctions, which are clearly used to separate two different clauses of the same sentence. For example, we discard the fragments between two entities if they contain only colon, semicolon, or other special characters. On the other hand, we do not remove any stop-words or prepositions, because they are important to differentiate peculiar ways of expressing certain relationships [78].

Placeholder Generalization Finally, we generalize the text on numerical values which are used to specify distances, dates, ordinal numbers but do not change the meaning of the sentence (recall we are interested in facts between named entities, excluding

Concept	Pl.	Regex	Original/Generalized Phrase
month	#M#	(January|...|December)	<A> released in September for <B> <A> released in #M# for <B>
day	#D#	#M# [1-2]?[0-9]	<A> drafted on June 21 from <B> <A> drafted on #D# from <B>
years	#Y#	[1-2][0-9][0-9][0-9]	<A> 's 2013 documentary film <B> <A> 's #Y# documentary film <B>
full dates	#FD#	#D#, #Y#	<A> was born on May 30, 1971 <B> <A> was born on #FD# <B>
nationalities	#N#	(Afghan|...|Zambian)	<A> written by Italian author <B> <A> written by #N# author <B>
ordinals	#O#	(first|second|third)	<A> is the second movie by <B> <A> is the #O# movie by <B>
length	#L#	[0-9]+ km|mi|ft|yd|m	<A> is located 62 mi north of <B>
positions	#P#	(north|...|east)	<A> is located #L# #P# of <B>

Table 4.3: Placeholders used to generalize phrases.

relations that involve literal vlaues). For the same reason we extend the generalization to adjectives expressing nationalities and positions. Table 4.3 shows examples of the placeholders used in English. However, as reported by the architecture in Fig. 4.1, we created the regular expressions even for the other experimented languages.

Fine-grained Types We introduce the types in order to specify the evidence available from the text, which is fundamental given the high number of fine-grained relations. Indeed, it is possible that the fragment of text between two entities is not enough to describe one relation rather than another. For example the phrase "*joined the*" can be used to describe instances of many different relations: it can describe the relation <dbo:associatedMusicalArtist> if the entities are a singer and a band; the relation <dbo:team> if is found between an athlete and a sport team or even the relation <dbo:tributary> if the surrounding entities are two rivers; etc. In all these cases, introducing the types of the entities can help in differentiate many ambiguous instances and allow to increase the quality of extracted facts. In our solution we discard the typed-phrases for which we can not provide a complete pair of DBpedia type but an interesting future work will be to allow missing types and generalize them relying on the hierarchy that is provided by the DBpedia Ontology.

In order to keep the process mostly language-agnostic we adopt very shallow natural language techniques to extract the *patterns* from the sentences. However, many improvements can be done in order to improve the recall of the whole system. For example, one can consider more sophisticated techniques such as identifying the precise connections between two entities in a sentence using the dependency tree, as has been done largely in many open relation extraction approaches [78, 79] discussed in Sec. 3.1.4.

4.4.2 Associate Typed-phrases to KG Relations

We take advantage from the the high number of facts that are already in DBpedia to associates typed-phrases to the relations. Indeed, our approach consists in obtaining a set of typed-phrases, for each DBpedia relation, which are commonly used to describe relative instances in the text of the articles. In the next pages we describe the principled method used to obtain such associations and, at the same time, discard the typed-phrases that do not provide any evidence of a relation from the KG.

Distantly Supervised Labeling We start from associate typed-phrases to relations in DBpedia following the standard distant supervision assumption [80, 26]. Specifically, each typed-phrase that we obtain between two entities in text is associated to all the DBpedia relations that hold between them in the graph. If more than one relation exists, we replicate the typed-phrases creating multiple associations; on the other hand, if no relations exist between the entities in DBpedia we associate the typed-phrase to a further <UNKNOWN> relation:

[Person] *founded* [Company]	⟷	<occupation>
[Person] *founded* [Company]	⟷	<foundedBy(-1)>
[Person] *met* [Person]	⟷	<UNKNOWN>
[Person] *and* [Person]	⟷	<UNKNOWN>

Continuing with our running example, we label two times the typed-phrase between <Steve_Jobs> and <Apple_Inc.> (they are linked with multiple relations in DBpedia) and once each of the other two typed-phrases because the relative entities are not related in

DBPedia. Due to the strong assumption used, this labeling process inevitably produces many noisy labels. We differentiate two categories of errors:

① we associate a typed-phrase to the wrong DBpedia relation. This is frequent when the entities hold multiple relations but the text describes only one aspect. For example, the typed phrase "[Person] *founded* [Company]" does not really describe that <Apple_Inc.> is the occupation of <Steve_Jobs>;

② we associate a typed-phrase which does not effectively support any of the existent relations. This is due to an *incompleteness of DBpedia schema*. For example, the typed-phrase "[Person] *met* [Person]" does not describe any relation of the graph, but we can found it labeled many times with the relation <spouse>;

In Table 4.4 we give some statistics on the numbers that we obtain applying this labelling approach on all the typed-phrases from different Wikipedia language-editions. Considering the English version, we obtain 44 M of instances of typed-phrases between consecutive entities in text. From them, we label 8.9 M instances with some existing DBpedia relations, replicating only 1.1 M (14%) to different relations. The remaining 36.2 M instances (80% of all the original typed-phrases) are labelled with the *dummy relation* <UNKNOWN>. Although the vast majority of them do not describe any specific relationship between the entities involved, many express evidence of some DBpedia relations but are there given that DBpedia is incomplete: those instances are exactly the ones that we will use to extract new facts.

	en	**de**	**it**	**fr**	**es**
all typed-phrases	44.0	15.4	5.5	8.1	5.8
labeled	8.9	1.1	0.6	1.6	0.8
↳ (replicated)	1.1	0.2	0.1	0.3	0.2
<UNKNOWN>	36.2	14.5	4.9	6.8	5.1

Table 4.4: Statistics of all the typed-phrase harvested for each language. All the numbers are expressed in millions (M).

Specificity of Typed-phrases and Relations After the labelling step, each typed-phrase is associated a certain number of times with one or more relations $r_1, ..., r_n$. To measure the strength of the association within a typed-phrase (tp) and a relation (r) we use a probability derived from their witness counts:

$$P(r|tp) = \frac{c(tp, r)}{\sum_{k \in R} c(tp, r_k)} \tag{4.1}$$

where $c(tp, r_i)$ is the count of typed-phrase tp with relation r_i and R is the set of all the relations in the KG. The probability describes the specificity of a typed-phrase with a relation and, by normalizing with all the relations in the KG, intuitively penalizes the typed-phrases that are used to express many different concepts. As we will see from the experimental evaluation, the high number of facts that are already present in the KG prevents many errors of type ①: if the typed-phrase describes a relation of the KG it is likely to be associated correctly. Indeed, we do not replicate many typed-phrases across different relations (Table 4.4). On the other hand, preventing bad associations derived from errors of type ② is more difficult because of the absence of any counter-evidence (aka. *negative examples*).

Considering Top-K from Positive Examples In the first version of Lector [18] we select the most significant typed-phrases for each relation. Intuitively, the most significant typed-phrases for a relation are the most common typed-phrases that are labelled mostly with it and not with other. Thus, the score of a typed-phrase tp with a relation r_i includes also the logarithm of its witness count, and become:

$$score(tp, r_i) = \log c(tp, r_i) \cdot P(r_i|tp) \tag{4.2}$$

As reported in the experiment section, we associated the typed-phrases to a singe relation by using a threshold on the probability value (Eq. 4.1). We rank the typed-phrases by their score and experiment the extractions increasing the number of typed-phrases for each relation. As we will see from the results, this approach is affected by the presence of noisy typed-phrases that, given their frequency, penalize a lot the accuracy of several relations.

Considering Unknown Examples In order to discard the typed-phrases that do not describe the relations of the KG we need to introduce some kind of negative examples. Indeed, we need counter evidences to balance the presence of the many labels that are obtained *by chance* (error of type ②). However, treating all the unknown typed-phrases, i.e. labelled with <UNKNOWN>, as negative examples would decrease a lot the recall because of the presence of many false positive (i.e. all the instances of the correct facts that we will extract). For this reason, we have to look for a trade-off.

To explain it with an example, suppose that the box in Fig. 4.3 below contains all the 44M instances of typed-phrases: the left side of the box contains the instances that are associated with a DBpedia relation, while the right side of the box contains all the unknown instances (i.e. <UNKNOWN>). The bars represent the instances of the typed-phrases listed on the left side of the figure (i.e. A,B and C), and are segmented based on the number of associations with each relation (for simplicity expressed with a number, e.g. R9) and the ones that are unknown (<UNKNOWN>).

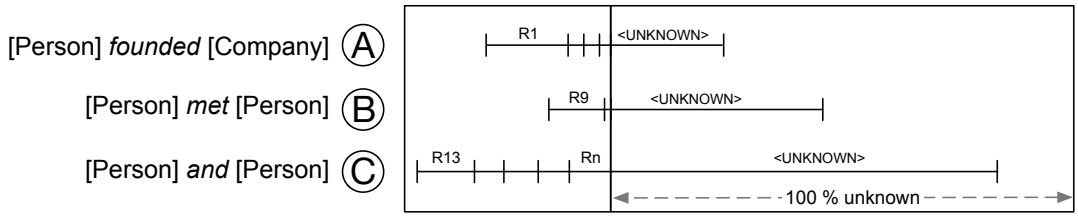


Figure 4.3: Examples of labels obtained from the standard distant supervision approach.

If we consider only the positive examples (i.e. instances associated to DBpedia relations) as done for Eq. 4.2, we have to look at the left side of the box. In this case, our specificity estimation (Eq. 4.1) would associate the typed-phrase A to the relation R1 (i.e. <dbo:foundedBy(-1)>) because it occurs most of the times with it, but it would fail with the typed-phrases B and C, associating them with the relations R9 (i.e. <dbo:spouse>) and R13 (i.e. <dbo:successor>) respectively, although with different values of specificity. This example confirms that the specificity (and the score, Eq. 4.2) is robust to the wrong labels originated by errors of type ①, but inevitably fails when the typed-phrases do not describe any particular relations, as in the cases of

the typed-phrases B and C. Furthermore, as we will see in the experiments, when the typed-phrases describe concepts that are *related* to the real relations (e.g. the typed-phrase "[Person] *met* [Person]" and the relation <dbo:spouse>, or "[Person] *grew up in* [Location]" and the relation <dbo:birthPlace>) they obtain high values of specificity.

However, as visible from the same figure, the ratio of unknown instances versus the ones that are labelled with a relation of the KG can be used to estimate if a typed-phrase expresses or not the relation. Thus, we treat <UNKNOWN> as the other relations and, for each typed-phrase, we associate the relation ($\hat{r}$) that maximizes its specificity:

$$\hat{r} = \arg\max_{r \in R^+} P(r|tp) \tag{4.3}$$

where R^+ is the set of relations that we consider, which include all the relation in DBpedia plus the unknown relation (<UNKNOWN>). In this way we associate a DBpedia relation only to the typed-phrases that are mostly labelled with it; on the contrary, when we associate the relation <UNKNOWN> we are discarding the typed-phrase. In order to balance the presence of unknown examples (recall they contain also many false positives) we *sample* them by picking random subsets of different percentages. With high percentages we are considering most of the unknown typed-phrases as negative examples, thus we are associating a minor number of typed-phrases with a relation in DBpedia. For example, we can not associate any typed-phrases of the example in Fig. 4.3 with a relation in DBpedia since all of them are mostly associated to <UNKNOWN>.

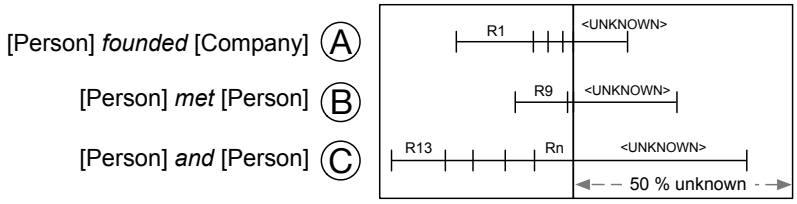


Figure 4.4: Examples of labels obtained using a random subset of unknown examples.

When we decrease the percentage used, some the of typed-phrases that were associated to <UNKNOWN> gradually become associated with a DBpedia relation. For example, if we consider the 50% of the whole set of unknown instances (Fig. 4.4) we are

able to associate the typed-phrase A to the relation R1 and avoid to associate the other two typed-phrases. Finally, when we do not use any unknown example, we are always associating typed-phrases with relations from DBpedia even with a small evidence.

In the experimental section, we evaluate Lector+ showing the number of facts and their accuracy that we can extract considering different percentages on unknown instances (unk). Intuitively, the higher is the percentage of unknown examples the greater are the results in terms of accuracy but at the cost of a lower recall.

4.5 Harvesting new facts

For each relation we use the associated typed-phrases to harvest new facts that are not present in DBpedia. In particular, we look for sentences that mentions pairs of entities that exists in the DBpedia but are not yet related, and compare the typed-phrases that we extract from them with the ones that we select for each relation. In order to augment the KG with a new fact $\langle s, r_i, o \rangle$, we check if the typed-phrase matches exactly one of the typed-phrases associated with the relations. More sophisticated matching methods such as approximate match of the text or the types are left for future work.

4.6 Experimental Evaluation

In this section we present the experimental evaluation that we perform on Lector and Lector+. In both the cases we cannot compare against a reliable ground truth. For this reason we follow the state-of-the-art [4] and manually evaluate a sample of facts extracted. In the next sections we discuss the results obtained.

4.6.1 Lector Evaluation

In its first version, the system used Freebase as reference KG, but the process used to extract the typed-phrases is comparable with the one described so far. For the evaluation we restrict the system to work only with relations in the domain of person

Relation in Freebase	Facts in Freebase	Lector			
		#new facts	*#eval.*	*accuracy*	*in infobox*
../place_of_birth*	662,192	57,140	347	.89 ± .03	25.4%
../place_of_death	178,849	18,458	104	.80 ± .07	17.3%
../nationality	584,792	50,234	290	.95 ± .02	21.7%
../teams	145,080	49,809	286	.96 ± .02	82.9%
../education	378,043	46,342	286	.98 ± .01	10.5%
../spouse	130,425	14,939	97	.31 ± .09	15.5%
../parents	123,747	5,648	50	.77 ± .10	8.0%
../children	141,860	3,149	50	.38 ± .12	10.0%
../ethnicity	39,869	2,989	50	.92 ± .06	2.0%
../religion	47,016	1,437	50	.94 ± .05	14.0%
../awards_won	98,625	1,934	50	.96 ± .03	18.0%
../party	65,300	3,684	50	.94 ± .05	10.0%

* A manual inspection, shows that removing one noisy fragment drastically increases the accuracy (see text)

Table 4.5: Number of new facts extracted by Lector compared to Freebase contents.

and, in order to quantify how many facts are missing also in YAGO and DBpedia (other than Freebase) we consider only the entities for which there exists a LOD link (through an `owl:sameAs` predicate) between Freebase and the other KGs. This resulted in approximately 977K entities. The facts, however, are extracted from all articles in Wikipedia. To choose the relations for our analysis, we considered the type signatures of the corresponding Freebase relations, focusing on general facts associated with every instance of `people/person` (e.g., nationality, birthplace and religion), as well as on facts pertaining to specific sub-types (e.g., the teams that sport athletes play for and the parties that politicians are affiliated to).

Parameter setting To associate typed-phrases to Freebase relations Lector considers the best-ranked typed-phrases according to their score (Eq. 4.2). The score is composed by two parameters: (1) a probability (Eq. 4.1), which defines the specificity of a typed-phrase for a relation; and (2) a maximum number K of phrases that can be associated with each relation, ranking them by a score (Eq. 4.2) that privileges frequent typed-phrases. For the purpose of this evaluation we set the minimum probability to 0.5 and $K = 20$. We experimented with other values of K, as reported below.

Results We first took a sample of 1500 random facts across all relations, which resulted in some relations having a small number of facts. In order to obtain a more significant confidence interval (below $\pm.1$), we re-sampled facts so that we could evaluate at least 50 facts for all relations. (The lines in Tables 4.5 and 4.6 separate the relations for which we had to re-sample). Besides the accuracy of the extractions, judged by reading the Wikipedia articles from which they were produced, we asked the human judges to determine whether the facts were given in the corresponding info-box (if the article had no info-box we counted that as a negative answer).

Table 4.5 reports both the number of facts found by Lector that are new to Freebase as well as their accuracy. As one can see, our approach can augment many relations in Freebase by 10% or more, and that, on average, less than 20% of Lector's new facts could have (and yet have not) been extracted from the info-boxes directly. In other words, the infoboxes could grow Freebase by 2% only.

Not only can Lector harvest several thousand facts, but also for most relations its accuracy is very high (in fact, comparable to the state-of-the-art [4]). For the cases where the accuracy is low, a manual inspection revealed that most errors were due to a small number of ambiguous phrases. For example, most of the errors in the relation `/spouse` are relative to the typed-phrase "[./person] *met* [./person]"[9] which naturally can be used to describe many other relations between two people (e.g., co-workers and teammates meet each other). A similar case happens for relation `/children` and the typed-phrase "[./person] *was succeeded by* [./person]". Removing such phrases increases accuracy: removing the typed-phrase "[./person] *grew up in* [./location]" from the relation `/place_of_birth` brings the accuracy up to 97.24% $\pm$ 1.49%.

Impact of K We performed an experiment to show how the choice of K affects the accuracy of the extracted facts. We evaluated the method for $K \in \{1, 5, 10, 15, 20\}$ and in terms of estimated precision and *relative* recall using $K = 20$ as the point of reference. The results are in Fig. 4.5, where the solid line represents the result considering all relations in Table 4.5 while the dotted line is the results without considering the three

[9] Note here we use the Freebase notation to express types and the relative typed-phrases.

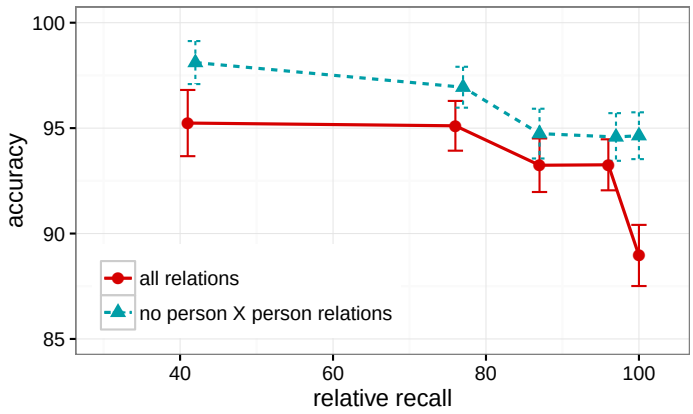


Figure 4.5: Accuracy vs (relative) recall varying K.

noisiest relations between two entities of type `people/person` (spouse, parents, and children). From left-to-right, the plots correspond to increasing K from 1 to 20, showing the expected trade-off between precision and recall, where the maximum recall is relative to the case of K=20. The higher precision on the less ambiguous relations indicates that proper filtering may lead to 95% accuracy without sacrificing recall.

Comparing to other KGs The facts extracted by Lector are absent in Freebase but could be present in other available KGs. For this reason we compare our extractions against DBpedia[10] and YAGO[11]. Table 4.6 reports the number of facts contained in both KGs as well as the number of new facts extracted by Lector. Only a small fraction of the new facts were already present in the other KGs, with the exception of the relation <dbo:team> in DBpedia (`playsFor` in YAGO) which is well populated due to the good coverage provided by the info-boxes. In any case, it is clear that Lector can be used to substantially grow these and other resources without affect their accuracy[12].

Our preliminary results shows that filtering ambiguous typed-phrases can lead to substantial improvements in accuracy. For this reason, we provide the evaluation of Lector+ in which we use the typed-phrases involving pairs of entities that are *not* related to clean the associations.

[10]We use the latest English mapping-based dataset (2015-04)
[11]We use English facts from YAGO3 (version 3.0.2)
[12]The dataset with the extracted facts is available at: http://dx.doi.org/10.7939/DVN/10795

Relation in DBpedia	Facts in DBpedia	Lector *#new facts*	Relation in YAGO	Facts in YAGO	Lector *#new facts*
<dbo:birthPlace>[1]	880,384	48,314	wasBornIn	281,067	55,577
<dbo:deathPlace>	243,415	15,818	diedIn	94,747	18,014
<dbo:nationality>	79,679	48,125	isCitizenOf	36,257	49,977
<dbo:team>	590,082	23,640	playsFor	525,374	35,013
<dbo:almaMater>	62,426	45,585	graduatedFrom	51,510	46,095
<dbo:spouse>	22,964	14,662	isMarriedTo	34,229	14,573
<dbo:parent>	19,508	5,631	-	-	-
<dbo:child>	10,219	3,140	hasChild	42,335	2,958
<dbo:ethnicity>	4,713	2,890	-	-	-
<dbo:religion>	26,021	1,368	-	-	-
<dbo:award>	65,874	1,655	hasWonPrize	120,529	1,370
<dbo:party>	36,866	3,594	isPoliticianOf	32,796	3,684

[1] The high number of facts is because the relation in DBpedia is not functional.

Table 4.6: Number of new facts extracted compared to state-of-the-art KGs

4.6.2 Lector+ Evaluation

Here we evaluate the effectiveness of Lector+ in extracting facts that can be used to augment DBpedia with new instances of many relations. In this case we do not limit the system to work with a restricted number of relations and typed-phrases, as happened in the previous evaluation. On the other hand, in order to limit the presence of noisy typed-phrases, which describe out-of-KG relations and penalize the overall accuracy of the system, we associate typed-phrases to relations considering also some *negative* instances. They are obtained picking random typed-phrases that involve pairs of entities that are *not* related in the KG.

In order to balance the presence of positive and *negative* examples we run the system varying the percentages of unknown examples to include in the model (unk), from 0% to 100%. Intuitively, the higher is the percentage of unknown examples the greater are the results in terms of accuracy but at the cost of a very low recall. On the other hand, a model which uses only positive examples (unk=0%) is able to associate larger amount of typed-phrases, thus extracting more facts at the cost of a lower accuracy.

We perform two different evaluations: (*a*) an internal evaluation, which shows the accuracy of the model in associating the typed-phrases with the DBpedia relation that

they express; (*b*) a manual evaluation, in which we can evaluate also the accuracy of the system in discarding ambiguous typed-phrases; with it we show the consequences of estimating negative examples on increasing percentages of unknown typed-phrases. Finally, we provide statistics on the facts that can be extracted from the different categories of articles, the same used in Sec. 4.3 for the entity detection evaluation.

Internal Evaluation In this experiment we follow the so-called *silver standard* evaluation approach [32] and we build a test-set directly from the facts in DBpedia. We split all of the 44M of entity pairs (i.e. instances of typed-phrases, Fig. 4.4) from all the articles in Wikipedia in five folds, evaluating the system via cross-validation. In particular, we use four of the five folds to associate possible typed-phrases to DBpedia relations, and one fold, containing the remaining entity pairs (and their relative typed-phrases), as a test set. Given a typed-phrase from the test set we check that the relation predicted by the model is already used to link the two entities in DBpedia.

We consider precision and recall to compare the performances of the system. Precision is calculated as the ratio between the number of entity pairs predicted correctly (i.e. we can associate their typed-phrases exactly with one of the relations that hold between them in DBpedia) and the total number of pairs predicted; recall is estimated using the same numerator but divided by the total number of pairs in the fold that are related in DBpedia. Since DBpedia is incomplete the approach is not reliable in measuring the recall, but we can use it as a rough estimation of the accuracy to associate the correct relation. Table 4.7 reports the results on different percentages of unknown typed-phrases and for each language, computed over the averages of the five folds.

model	en		de		it		fr		es	
	p.	**r.**	**p.**	**r.**	**p.**	**r.**	**p.**	**r.**	**p.**	**r.**
unk=0%	0.92	0.49	0.89	0.50	0.96	0.56	0.96	0.49	0.94	0.41
unk=25%	0.94	0.45	0.92	0.42	0.97	0.53	0.97	0.46	0.96	0.38
unk=100%	0.96	0.38	0.94	0.32	0.98	0.48	0.98	0.42	0.97	0.30

Table 4.7: Average Precision (p.) and Recall (r.) from the 5-folds cross-validation.

This experiment is useful to show that the system is able to associate most of the typed-phrases to the correct relations from the ones available in DBpedia, independently from the amount of unknown typed-phrases used. However, the experiment can not measure the real recall obtained by the system because of the presence of ambiguous typed-phrases that should not be used for the extractions. For example, if we have "[Person] *met* [Person]" as a possible typed-phrase in the test set, the model which predict the relation <dbo:spouse> would have a benefit (in recall) compared to the one that – correctly – abstain from the prediction.

Complete Extraction of Facts We extract all the facts considering 21 different models, built on increasing percentages of unknown examples (from 0% to 100%). Table 4.8 reports, for each model, the number of typed-phrases that can be associated to the relations and ones that are effectively used to extract new facts. Together with the number of facts that can be extracted we also provide an estimation of their accuracy (the same is shown in Fig. 4.6) that we discuss later on this section.

The numbers obtained confirm what we discuss in Sec. 4.4.2: high percentages of unknown examples allow to associate less typed-phrases to DBpedia relations, bringing fewer new facts. In these cases, the models effectively abstain from predict a relation on the sentences whom typed-phrases do not have enough evidence. Instead, decreasing the percentage, we associate a larger number of typed-phrases with DBpedia relations, thus extracting new facts.

Quantitatively speaking, considering all the unknown examples (unk=100%) we can extract ∼787 K facts from a bit more than ∼61 K typed-phrases. Just considering 5% less of unknown example (unk=95%) we can add ∼33 K new facts (∼820 K in total) using less than ∼1,5 K new typed-phrases. On the other hand, the model that consider only positive examples (unk=0%) is the most relaxed one: using ∼111 K typed-phrases it extracts more than ∼5,4 M new facts (42% more than the previous model with unk=5%) but, as we will see in the evaluation, with a very low accuracy.

unk	Associated t-p (# *rel.*)	Used t-p (# *rel.*)	#*new facts*	*accuracy*
0%	227,461 (464)	111,237 (390)	5,431,082	0.53
5%	223,785 (458)	107,633 (385)	3,828,150	0.65
10%	219,529 (456)	103,423 (382)	3,069,571	0.70
15%	215,748 (455)	99,569 (378)	2,617,919	0.74
20%	212,343 (455)	96,053 (379)	2,312,371	0.78
25%	209,006 (452)	92,842 (370)	2,053,454	0.80
30%	205,985 (455)	89,853 (375)	1,851,339	0.80
35%	203,226 (453)	87,114 (371)	1,708,812	0.82
40%	200,555 (452)	84,299 (367)	1,581,636	0.84
45%	198,085 (448)	81,911 (368)	1,479,000	0.84
50%	195,843 (449)	79,610 (364)	1,374,362	0.85
55%	193,580 (450)	77,465 (366)	1,300,389	0.87
60%	191,617 (448)	75,383 (362)	1,224,023	0.88
65%	189,452 (449)	73,448 (357)	1,159,365	0.88
70%	187,614 (448)	71,568 (358)	1,059,690	0.88
75%	185,763 (447)	69,608 (356)	1,011,861	0.89
80%	184,004 (448)	67,993 (354)	970,129	0.89
85%	182,419 (448)	66,229 (353)	921,570	0.90
90%	180,787 (447)	64,715 (352)	867,827	0.91
95%	179,243 (447)	63,189 (350)	820,262	0.91
100%	177,775 (447)	61,705 (349)	787,639	0.91

Table 4.8: Typed-phrases associated to DBpedia relations and used for the extractions.

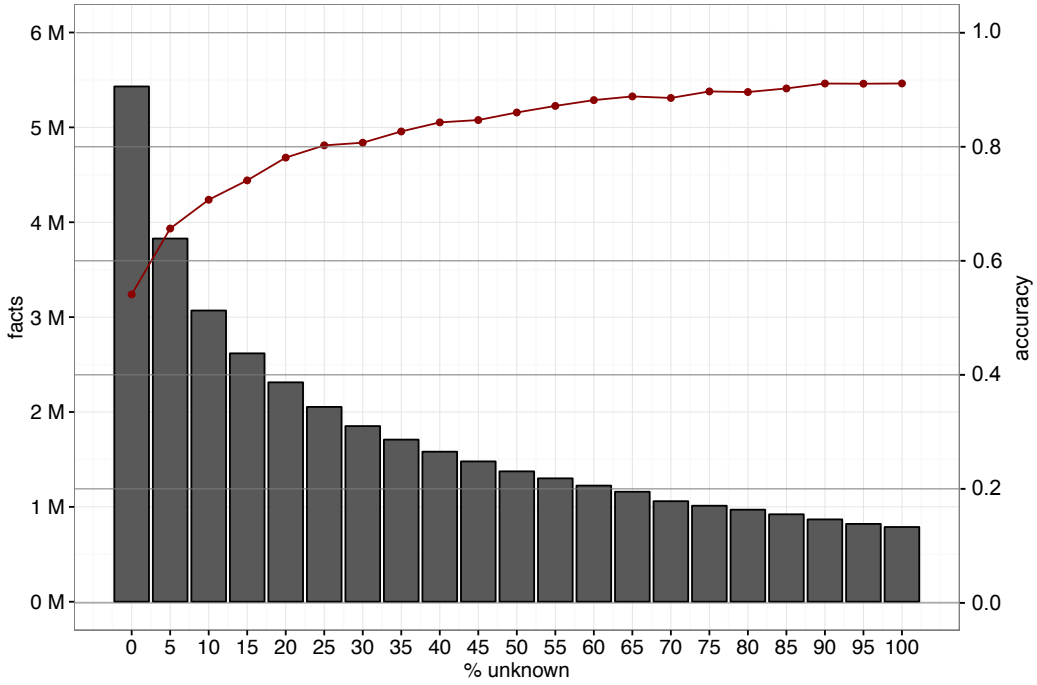


Figure 4.6: Facts extracted and accuracy obtained by each model.

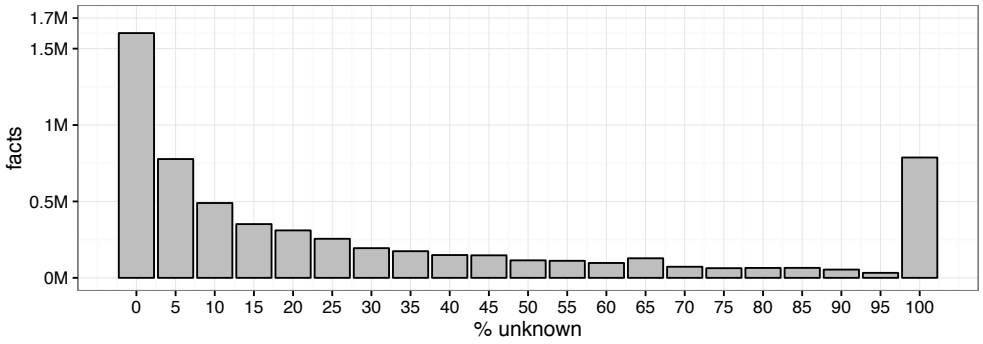


Figure 4.7: Estimated accuracy for the facts extracted in each level of unknown.

Manual Evaluation As we did for Lector, we estimate the accuracy obtained by the system evaluating – manually – a sample of the facts extracted. Since we want to measure the effectiveness of the system on varying the percentage of unknown examples, we evaluate – for each model – random facts extracted with the typed-phrases that become associated to the relations starting by that model. Fig. 4.7 shows better the situation: each bar represents the number of facts that can be extracted by the relative model, that were not extracted by the next model on its right. For example, a model with unk=100% extracts ∼787 K facts; as described before, decreasing the percentage to unk=95% we add ∼33 K new facts; ∼47 K new facts with unk=90%; etc. up to the model with unk=0% which extracts ∼1,6 M facts more than the model with unk=5%.

From each of these groups we pick 150 random facts, obtaining a total of 3,150 facts. The judges evaluate the correctness of the extractions based only on the evidence from the sentence, without any influence from their background knowledge. For example, the fact <Mahatma_Gandhi> <dbo:birthPlace> <India> extracted from the sentence "*Gandhi grew up in India before to move in England*" should be evaluated as wrong even if the fact is effectively true. In this way, we can measure the real effectiveness of the typed-phrases to describe exactly the relations from the KG.

For each sample we calculate the precision, from the number of facts that are evaluated correct, and estimate a confidence interval using the Wilson score for $\alpha = 5\%$. We report all the accuracies in Fig. 4.8. As visible from the results, the accuracy obtained by the model created on lower percentages of unknown examples is low.

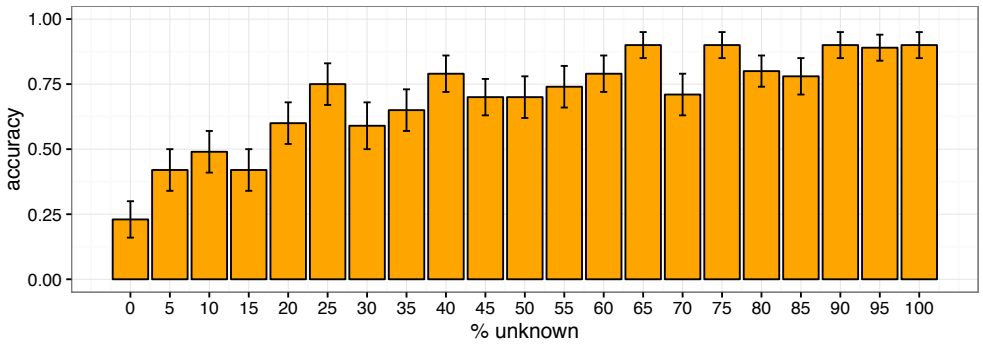


Figure 4.8: Number of facts extracted with the typed-phrases associated at each percentage of unknown.

At unk=0% most of the facts are obtained by very generic typed-phrases such as conjunctions or prepositions, which are common to separate entities but do not describe any particular relationships between them. From unk=5% to unk=15% the results are still bad, but here we found many typed-phrases which describe relations between entities that do not exists in KG. For example, the typed-phrases "[Person] *met* [Person]" associated to <dbo:spouse>, "[Person] *left* [Country]" associated with <dbo:birthPlace> or "[Person] *returned to* [Country]" with <dbo:deathPlace>. Even if all these typed-phrases describe good relationships between entities we can not use them to augment the content of DBpedia because of the incompleteness in the schema.

From unk=20% we start to obtain very good facts. In particular, the high accuracy obtained with unk=25% is due to the presence of many instances of the typed-phrase "[Person] *is a* [Country]" associated with <dbo:nationality>; the relation is not well populated in DBpedia but the typed-phrase is widely used in the text. A similar situation happens at unk=65% where we associate the typed-phrase "[Person] *was born in* [Settlement]" to the relation <dbo:birthPlace>; in that case many facts are correct resulting in a strong increment of the accuracy.

Finally, we obtain an accuracy around 0.9 using models with high percentages of unknown examples (unk=90%-100%). In these cases, the typed-phrases used are very descriptive of a DBpedia relation: "[BaseballPlayer] *was selected by* [BaseballTeam]" with the relation <dbo:team> or "[Film] *with* [Actor]" with the relation <dbo:starring>.

In order to calculate the accuracy obtained by Lector+ using a specific percentage of unknown examples (Fig. 4.6) we aggregate the accuracies of that model with the ones that use higher percentages of unknown examples, weighting them by the number of facts extracted from each of them. As we can see from its trend the accuracy constantly decreases with lower percentages of unknown examples. At unk=25% the accuracy reaches 0.8, but its general trend is still stable. After it, the accuracy of the system starts to decrease rapidly. For this reason, we consider unk=25% a good trade-off value for the percentage of unknown examples to use.

Where are the facts on the articles? We analyse all the facts that can be extracted by the system to quantify the advantages of (*a*) detecting instances of primary and secondary entities and (*b*) using the whole article, instead of only the abstract as done in previous works [81]. We reports such statistics for each extraction that we computed.

Fig. 4.9(a) shows, at each percentage of unknown examples, the number of facts that we can extract considering pair of different combination of primary (PE) and secondary (SE) entities that were already highlighted in the text with a wiki-link (*org*, for "original"), or have been detected with our methods described in Sec. 4.3 (*aug*, for "augmented"). It reveals that most of the facts involve entities that have been detected by the system. Indeed, using only the original secondary entities (which are the ones already in the articles) we could have extracted between the ∼30-40% of all the facts (the dark blue part of the bars in Fig. 4.9(a)) with each model. From the same figure, it is evident the importance of detecting the primary entities: their instances bring between the ∼35-45% of all the facts (the white and the light blue parts of the bars in Fig. 4.9(a)) across all the models.

Several works tried to extract facts from Wikipedia text considering only the abstract of the articles [81]. Indeed, this part of the page is well known to be dense of information that could be used for augmentation. However those information are also the ones that are likely to be already in DBpedia. We report statistics on the number of facts that we can extract from the articles in Fig. 4.9(b).

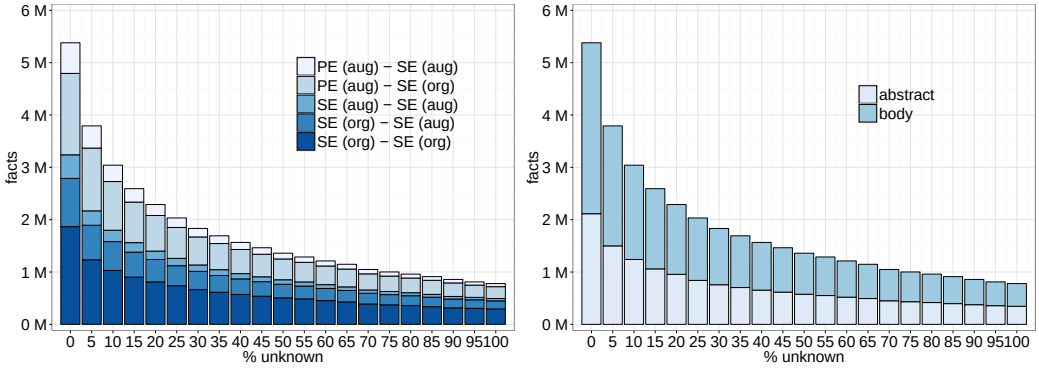


Figure 4.9: Statistics of the facts extracted by different combinations of entities (left) and from different section of the article (right).

Per Relation Evaluation To participate at the DBpedia TextExt Challenge we run the system using unk=25%. The system could extract ∼1.8 M of facts that are not present in DBpedia with an accuracy estimated at 0.8.

To make more sense of the DBpedia relations that we can augment we analyse the facts extracted from different kind of articles, as done in evaluating the primary entity detection methods (Sec. 4.3). Thus, from each article about Person, Places, Organizations and Creative Works (book, films, etc.), we select only the facts that involve the primary entities which, as we saw in the previous section, can be considered around the ∼35-45% of all the extractions. In Table 4.9 we report the number of facts for the top-5 relations that we can augment from each category of articles. As we can see, we can augment some relation such as <dbo: almaMater>, <dbo:nationality>, <dbo:hometown> by as much as 10%. Some relations, for example the ones describing geographical attributes such as <dbo:nationality>, <dbo:country>, <dbo:ground>, etc. could be augmented very well considering only the abstract of the articles: those information in fact are often expressed directly on the first sentence.

In Table 4.10 we report, for some relations, the most common typed-phrases (associated using unk=25%) that extract correct facts from the English Wikipedia. As we can see, the types of the entities help in cleaning many ambiguities derived from using common expressions. For example, the typed-phrases "[Station] *is on the* [RailwayLine]", "[MusicalArtist] *toured with* [MusicalBand]", "[Island] *is a* [Country]", etc.

Relation	Facts in DBpedia	# new facts	% in abstract
Top-5 relations from articles about **Person**:			
<dbo:birthPlace>	979.490	163.471	50 %
<dbo:team>	477.886	64.584	28 %
<dbo:deathPlace>	297.771	44.454	36 %
<dbo:almaMater>	115.724	22.095	25 %
<dbo:nationality>	97.161	17.346	94 %
Top-5 relations from articles about **Place**:			
<dbo:isPartOf>	884.489	36.773	77 %
<dbo:location>	247.377	13.790	79 %
<dbo:department>	28.551	8.814	99 %
<dbo:country>	550.066	6.328	72 %
<dbo:district>	46.737	4.259	87 %
Top-5 relations from articles about **Organization**:			
<dbo:city>	73.779	5.695	81 %
<dbo:ground>	29.383	5.147	95 %
<dbo:hometown>	54.354	5.016	77 %
<dbo:country>	72.808	4.575	89 %
<dbo:location>	95.056	3.527	72 %
Top-5 relations from articles about **Creative Work**:			
<dbo:starring>	365.470	2.084	65 %
<dbo:musicalArtist(-1)>	53.131	1.781	19 %
<dbo:country>	58.553	1.369	48 %
<dbo:writer>	158.596	1.295	63 %
<dbo:artist(-1)>	136.373	1.027	30 %

Table 4.9: Statistics and evaluation of the top-5 relations augmented for each domain in the English edition.

Multilingual Facts Extraction Other than English, we tested the tool on several other language-editions of Wikipedia: German, Italian, French and Spanish. Indeed, the advantage of our system is that it does not rely on sophisticated NLP techniques and so it can be adapted to any languages with a minimal effort. We run the system using the same percentage of unknown examples exploited for the English language edition (i.e. unk=25%), although we know that the correct trade-off could change across different language. In Table 4.11 we report some numbers on the facts extracted using the primary entities and different category of articles in the various languages. The difference in terms of *recall*, w.r.t. the English edition, is partially because we did not extend all the entity detection techniques, relying only on a subset of the lexicalizations used to detect instances of the primary entity. However, as from Table 4.9, there is also a big difference in terms of number of articles compared to the English edition.

Relation	Common Typed-phrases	Relation	Common Typed-phrases
<dbo:birthPlace>	"[Person] *was born in* [Settlement]" "[Person] *was born at* [Settlement]" "[Person] *was born in #YEAR# in* [City]" "[Person] *was born on #DATE# in* [Settlement]" "[Person] *'s hometown* [City]"	<dbo:team>	"[BaseballPlayer] *signed with* [BaseballTeam]" "[SoccerPlayer] *joined* [SoccerClub]" "[BaseballPlayer] *was drafted by* [BaseballTeam]" "[SoccerPlayer] *moved to* [SoccerClub]" "[RugbyPlayer] *played for the* [RugbyClub]"
<dbo:almaMater>	"[OfficeHolder] *graduated from* [University]" "[Writer] *attended* [University]" "[Person] *enrolled at* [University]" "[OfficeHolder] *was educated at* [University]" "[Scientist] *studied at the* [University]"	<dbo:associatedMusicalArtist>	"[MusicalArtist] *collaborated with* [MusicalArtist]" "[MusicalArtist] *joined* [MusicalBand]" "[MusicalArtist] *worked with* [MusicalArtist]" "[MusicalArtist] *toured with* [MusicalBand]" "[MusicalArtist] *co-wrote with* [MusicalArtist]"
<dbo:starring(-1)>	"[Person] *played* [Film]" "[Person] *joined the cast of* [TelevisionShow]" "[Person] *also appeared in* [Film]" "[Person] *starred in* [Film]" "[Person] *has appeared in* [Film]"	<dbo:isPartOf>	"[Town] *is a town in* [AdministrativeRegion]" "[City] *is a city located in* [AdministrativeRegion]" "[Settlement] *is a village in* [Settlement]" "[Settlement] *became part of* [Settlement]" "[Settlement] *is a commune in* [Settlement]"
<dbo:country>	"[Settlement] *became part of* [Country]" "[Settlement] *is a village in the* [Country]" "[Settlement] *is a district in the* [Country]" "[Island] *is a* [Country]" "[Settlement] *is a city in #POS#* [Country]"	<dbo:servingRailwayLine>	"[Station] *is on the* [RailwayLine]" "[Station] *is a station on the* [RailwayLine]" "[Station] *is served by the* [RailwayLine]" "[Station] *is located on the* [RailwayLine]" "[Station] *was a railway station on the* [RailwayLine]"
<dbo:league>	"[SoccerClub] *won the* [SoccerLeague]" "[SoccerClub] *joined* [SoccerLeague]" "[SoccerClub] *played in* [SoccerLeague]" "[SoccerClub] *was relegated to* [SoccerLeague]" "[SoccerClub] *finished #ORD# in* [SoccerLeague]"	<dbo:publisher>	"[VideoGame] *was published by* [Company]" "[Book] *was published by* [Publisher]" "[VideoGame] *was released by* [Company]" "[AcademicJournal] *is published by* [Publisher]" "[VideoGame] *'s publisher* [Company]"
<dbo:location>	"[Company] *is a* [Country]" "[Company] *is headquartered in* [City]" "[Company] *was founded in* [Settlement]" "[TelevisionStation] *channel #NUM# in* [City]" "[RugbyClub] *is a club based in* [Settlement]"	<dbo:recordLabel>	"[Band] *signed with* [RecordLabel]" "[Band] *left* [RecordLabel]" "[Band] *signed a contract with* [RecordLabel]" "[Band] *parted ways with* [RecordLabel]" "[Band] *'s label,* [RecordLabel]"

Table 4.10: Statistics on the most common typed-phrases that are used to extract correct facts from the English edition.

de		it		fr		es	
Relation	**# facts**	**Relation**	**# facts**	**Relation**	**# facts**	**Relation**	**# facts**
Top-5 relations from articles about **Person**:							
<dbo:birthPlace>	10.127	<dbo:birthPlace>	2.182	<dbo:birthPlace>	14.589	<dbo:birthPlace>	4.899
<dbo:deathPlace>	7.070	<dbo:parent>	1.055	<dbo:nationality>	2.811	<dbo:team>	2.068
<dbo:team>	2.228	<dbo:deathPlace>	467	<dbo:teamManager>	964	<dbo:hometown>	1.019
<dbo:starring>	622	<dbo:successor>	101	<dbo:team>	748	<dbo:deathPlace>	924
<dbo:predecessor>	605	<dbo:author>	98	<dbo:deathPlace>	446	<dbo:spouse>	360
Top-5 relations from articles about **Place**:							
<dbo:city>	5.612	<dbo:administrativeDistrict>	6.714	<dbo:country>	2.175	<dbo:province>	1.504
<dbo:administrativeDistrict>	1.925	<dbo:location>	613	<dbo:isPartOf>	914	<dbo:map>	1.306
<dbo:location>	1.586	<dbo:capital>	60	city	750	<dbo:location>	621
<dbo:county>	1.308	<dbo:routeStartLocation>	38	<dbo:region>	406	<dbo:type>	380
<dbo:province>	867	<dbo:servingRailwayLine>	16	<dbo:county>	379	<dbo:municipality>	275
Top-5 relations from articles about **Organisation**:							
<dbo:genre>	1.331	<dbo:location>	160	<dbo:headquarter>	1.433	<dbo:stadium>	902
<dbo:city>	1.090	<dbo:parentOrganisation>	89	<dbo:nationality>	389	<dbo:headquarter>	376
<dbo:league>	407	<dbo:foundationPlace>	54	<dbo:parentCompany>	233	<dbo:city>	131
<dbo:industry>	202	<dbo:childOrganisation>	35	<dbo:childOrganisation>	222	<dbo:tenant>	82
<dbo:type>	185	<dbo:foundedBy>	32	<dbo:country>	184	<dbo:aircraftUser>	58
Top-5 relations from articles about **Creative Works**:							
<dbo:genre>	509	<dbo:author>	44	<dbo:artist>	103	<dbo:type>	226
<dbo:category>	275	<dbo:artist>	34	<dbo:director>	95	<dbo:artist>	224
<dbo:musicType>	158	<dbo:country>	31	<dbo:programmingLanguage>	42	<dbo:author>	92
<dbo:operatingSystem>	141	<dbo:productionCompany>	22	<dbo:author>	38	<dbo:operatingSystem>	80
<dbo:developer>	111	<dbo:notableWork>	20	<dbo:publisher>	25	<dbo:musicalArtist>	54

Table 4.11: Statistics of the top-5 relations augmented for each domain in the various language editions.

Chapter 5

Towards Annotating Relational Data on the Web

Tables and structured lists on Web pages are a potential source of valuable information, and several methods have been proposed to annotate them with semantics that can be leveraged for search, question answering and information extraction.

Here we describe our approach to solve the problem of finding and ranking relations from a given Knowledge Graph (KG) that hold over pairs of entities juxtaposed in a table or structured list. The state-of-the-art for this task is to attempt to link the entities mentioned in the table cells to objects in the KG and rank the relations that hold for those linked objects. As a result, these methods are hampered by the incompleteness and uneven coverage in even the best knowledge graphs available today. The alternative described here does not require entity linking, relying instead on ranking relations using generative language models derived from Web-scale corpora. As such, it can produce quality results even when the entities in the table are missing in the KG. The experimental validation, designed to expose the challenges posed by KG incompleteness, shows that our approach is robust and effective in practice.

The work and the experimental evaluation described in this chapter are currently under revision for an international conference.

5.1 Introduction

The Web is a vast source of intrinsically relational knowledge expressed in hundreds of millions of tables and many more structured lists within billions of documents. Web-scale table corpora have found many applications, including search and question answering [82, 83, 14], knowledge graph construction [8, 84, 58, 85], schema understanding and auto-complete [82, 86], to name a few. However, unlike with documents in which information is encoded in text amenable to natural language understanding tools, the facts and relationships encoded in tables are *implicit*, and therefore hard to extract automatically. The various approaches for *understanding* Web tables amount to two main tasks: (1) identifying the *type* of each column in a table, and (2) identifying the *relationship* between pairs of columns in the table. As a motivating example, Fig. 5.1 shows a snippet of a prototypical table (from Wikipedia in this case) that can be easily extracted and parsed with tools like Google tables or an automatic wrapper induced from a set DOM-trees. It is clear to a human looking at the table that the second column has *actors* who played in the *movies* in the first column. With little effort (e.g., after reading the article with that table) a human can infer that the third column has *countries* that were *filming locations* of the *movies* in the first column. Yet, extracting those types and relationships is out of reach of text-based Information Extraction tools that rely on linguistic patterns used to encode knowledge [54].

The first methods for Web table understanding [14] were strictly lexical, using frequently occurring keywords and phrases as annotations, and are primarily useful for keyword-based table search. The prevalent approach, however, is to leverage existing Web-scale Knowledge Graphs (KGs) for *semantic* Web table understanding, whereby one annotates columns with *classes* of entities and pairs of columns with *relationships* from the KG ontology [17, 83, 16, 65, 87]. To do so, these methods attempt to disambiguate entities in the table by *linking* them to objects in the KG. If this can be done, one can immediately annotate each table column with the ontology type covering the entities in the column. Next, the relationship between a pair of columns can be inferred *ranking* KG relations based on their *coverage* of (entity pairs in the rows) of the table.

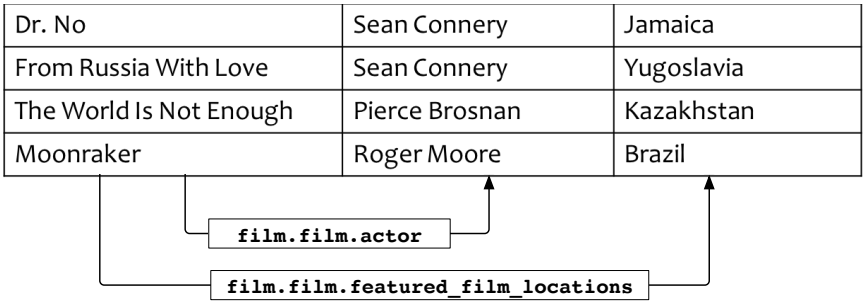


Figure 5.1: Web table with actors and filming locations of James Bond movies.

Although highly intuitive, the approach outlined above is hampered by the fact that even the best existing KGs are notoriously incomplete [10, 88, 89], missing many entities (not only obscure tail entities) as well as many relations among the entities. More precisely, KG incompleteness introduces two problems. First, if the Web table contains mostly entities that are not in the KG or cannot be easily linked, no table annotations are possible. It is worth mentioning that state-of-the-art entity linking methods rely on textual features (e.g., keyphrases) which are hardly available in the context of Web table understanding. Second, the KG coverage for specific relations is often biased, which can lead to unexpected results even if the entities can be correctly linked as illustrated next.

The example of Fig. 5.1 was chosen systematically to illustrate the problems caused by KG incompleteness. We picked a table with a column whose cells would: (1) be difficult to link to Freebase objects, and (2) participate in a partially populated relation of interest. In our example, the names of the movies are hard to distinguish from their respective *soundtrack* albums, even with sophisticated string matching methods (e.g., [90]). Moreover, although we can easily disambiguate the countries in the table, and although Freebase has a dedicated relation for filming locations of movies, the coverage of that relation is heavily biased towards recent movies, lacking the filming location of most Bond movies. Freebase is not as incomplete, however, in the music domain. In fact, it contains all countries were the soundtracks of the Bond movies were released. As a result, one would be biased towards annotating the relationship between

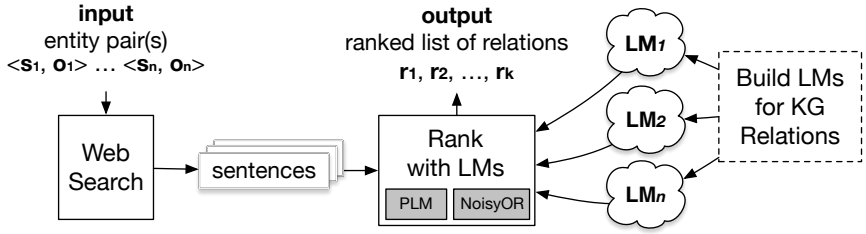


Figure 5.2: Overview of our approach.

the first and third columns of Fig. 5.1 with the predicate for the region where an album is released, which would be incorrect in this case. (In passing, at the time of writing, both YAGO and DBpedia also lack the filming locations of most Bond movies.)

5.1.1 Problem Statement

For convenience, we adopt the Freebase notation and terminology.

As customary, we model a **knowledge graph** as a labeled, directed multi-graph $KG = (N, E, L)$ where N, E, and L are sets of nodes, edges, and labels. Nodes can be entities, represented using unique identifiers called "m-ids" (e.g., `m/03_gd`), text literals in quotes (e.g., `"James Cameron"`), or types denoted as paths in an ontology (e.g., `film/director`). Labels in L define relation names (e.g., `film/director/film`). Edges in $E \subseteq N \times L \times N$ are called statements or triples and can be used to assign types to entities (e.g., $\langle$`m/03_gd`, `type/object/type`, `film/director`$\rangle$); to describe entities (e.g., $\langle$`m/03_gd`, `type/object/name`, `"James Cameron"`$\rangle$) or to relate pairs of entities (e.g., $\langle$`m/03_gd`, `film/director/film`, `m/0dr_4`$\rangle$). We assume the KG ontology specifies types for the domain and range of every relation.

This paper seeks to describe and evaluate a principled and effective way of predicting KG relations that hold over columns of Web tables. (Note that doing so allows one to annotate the columns themselves with the domain and range types from the KG ontology for those relations.) Predicting which relation(s) hold for pairs of entities amounts to *ranking* all KG relations for those entities followed by either thresholding or taking the top-k relations in the ranking. Thus, in this paper we focus on, and evaluate

phrase	freq.
was released in	0.23
topped the charts in	0.17
is available only in	0.14
...	...
shot in	0.04

(a) `music/release/region`

phrase	freq.
was filmed in	0.44
set in	0.26
shot in	0.14
...	...
was released in	0.03

(b) `.../featured_film_locations`

Figure 5.3: Generative language models for two relations between works of art and countries.

sentences	freq.
Dr. No filmed entirely in **Jamaica**	6
Dr. No location Ocho Rios **Jamaica**	3
Dr. No based and shot in **Jamaica**	2
Dr. No filmed in Kingston, **Jamaica**	1

Figure 5.4: Web search results for ⟨`"Dr. No"`,`"Jamaica"`⟩.

relation rankings instead of predictions. Without loss of generality, we assume the input to be a set of entity pairs, as one can always convert a multi-column table or a nested list into one or more sets of pairs. Also, we assume that all pairs in the set are similarly related; or, in other words, the input is not random. More precisely:

Definition 1. Given a set $I = \{\langle s_1, o_1\rangle, \ldots, \langle s_n, o_n\rangle\}$ of subject-object entity pairs, and a list L of relation names from a KG, produce a ranked list $r_1, \ldots, r_k$ of k relations from L that hold over pairs in I, sorted by decreasing relevance.

5.1.2 Overview of Our Approach

Fig. 5.2 illustrates our method, which is heavily inspired by the established Language Models (LMs) for IR approach [91], in which the goal is to model each document separately and score documents w.r.t. queries based on how likely the corresponding models are to generate the query. Keeping with this analogy, the KG relations in our setting play the role of the "documents" and the pairs of entities play the role of the "queries".

Models of KG Relations We model a KG relation distributionally using the *phrases* that are used to express it. We learn such models from the approximately 500M texts in English of the ClueWeb09 corpus, leveraging Google's FACC1 annotation corpus assigning m-ids of Freebase entities to the *mentions* in those texts where these entities appear. Following the state-of-the-art in open relation extraction [53], we gather phrases appearing *between* m-ids in the corpus and filter out phrases that describe ontological relations (e.g., "is a" and variants) and phrases that do not conform to known patterns defined at the level of parts-of-speech [1] tags. The details of the construction of LMs are given in Sec. 5.3.

Relation Ranking To rank KG relations for entity pair $\langle s_i, o_i \rangle$, we perform a Web search with those entities and extract *relational phrases* connecting the entities in the result of the Web search. Then, we score the KG relations based on how likely their corresponding models are to generate the set of phrases extracted from the Web search. Continuing with our running example, Fig. 5.3a shows some of the phrases of the LM associated with the relation associating albums to the countries they have been released, while Fig. 5.3b shows phrases for the relation for filming locations of movies. Fig. 5.4 shows some sentences from the Web search with entities $\langle$`"Dr. No","Jamaica"`$\rangle$. In this case, the relation about movie locations will rank higher than the relation about album releases.

Two models for KG relation scoring are considered here (see Sec. 5.4.2): (1) PLM, the "standard" *conjunctive* approach of maximizing the likelihood of all terms in the query, and (2) Noisy OR, a *disjunctive* approach where the ranking may be biased towards a small number of highly relevant phrases. Finally, note that when the input consists of multiple entity pairs, we need a way of finding aggregate scores for the relations relative to all such pairs. Again, two ways of doing so are discussed and evaluated here (Sec. 5.4.3): (*a*) a *global* strategy that combines all sentences of all pairs into a single *global* query used to rank the LMs (once), and (*b*) a *local* strategy where we produce a ranking for each pair, merging them to arrive at a final prediction.

Evaluation To the best of our knowledge, there are no benchmarks concerned with KG incompleteness and how they affect table understanding. Therefore, we designed two experiments to illustrate how our method can overcome this problem. In the first we use a synthetic benchmark with facts that are both true and known to be missing from Freebase, DBpedia, and YAGO (Sec. 5.5), following our previous work [18]. In the second we experiment with tables from Wikipedia for which a state-of-the-art web table annotation tool [17] fails to produce any output (Sec. 5.5.7).

Contribution We describe an effective method for ranking KG relations that applies to pairs of named entities that is less susceptible to KG incompleteness than the current state-of-the-art. Departing from previous work, our method does *not* require that the entities are linked to, or even exist in the KG. Instead, our method works whenever a Web search returns phrases describing how the entities are related. Our method is general: it is not specific to any KG, corpus or natural language, and the Web search can be replaced by a search on any large corpus. Finally, perhaps the biggest advantage of our method is that it predicts relations based on *corpus* statistics which are *independent* of, and not biased by the KG coverage. Our experimental evaluation, which has been conducted on publicly available datasets, indicates that our method can correctly predict relationships for in-KG and for out-of-KG entities, where state-of-the-art approaches fail, and thus can significantly contribute to improve previous work in this area.

5.2 Related Work

Language Models have found many uses in Information Retrieval beyond document ranking [92, 93, 91]. A state-of-the-art entity search method [94] is based on "entity LMs" that harness entity categories (i.e., semantic types) for ranking and filtering answers based on a desired *type* (e.g., movies, albums, etc.). Other applications include searching and ranking over RDF-structured Linked Data and knowledge graphs with queries that combine keywords and entity examples [95] or interpret so-called telegraphic

text queries [96] on the underlying structured data [97]. LMs have also been found useful in ranking the results of exact, relaxed and keyword-augmented graph-pattern queries over RDF graphs [98, 99], which has applications in translating natural language questions into SPARQL queries over KGs [100], among others. We use LMs for relation prediction.

Our work borrows from relation prediction methods that exploit the duality between KG relations and phrases occurring in text (e.g., [35, 54, 101, 102, 103]), except that we employ strict filters to remove non-relational and ontological patterns [1]. While we use LMs for prediction and achieve good results, other ranking models from Information Retrieval and/or other relation prediction models from the field of Information Extraction could be used for the same purpose. We leave as future work investigating other scoring models and how they fare in our setting.

We are motivated by the problem of understanding Web tables, widely recognized as a valuable knowledge source on the Web [82]. The first solution [14] is meant for search, annotating columns with keywords from an "is-a" database and relationships between columns with *keyphrases* frequently occurring with the entities in the table. On the other hand, by annotating pairs of columns with KG relations that hold over the respective entities, our method annotates the tables with semantic information. With our method, the columns can be annotated with the *expected* (semantic) types for the relations as per the KG schema.

Recent work on Web table understanding links entities in *table cells* to KG objects and pairs of columns with KG relations that hold over them [83, 16, 65, 87]. An earlier work in this area [83] learns a probabilistic graphical model that collectively annotates cells with entity identifiers, columns with KG types and pairs of columns with KG relations, maximizing the joint probability of the assignment. Another idea is to model each row of the table as a set of (possibly multi-valued) attributes describing a single entity in the KG [16]; each row of the table is then matched with entities in DBpedia, taking into account the table headers and how well they match classes in the DBpedia ontology. Unlike these works, our method does not require linking entities to

KG objects and, thus, should be less susceptible to KG incompleteness. We validate this hypothesis experimentally in two ways. First, we use a synthetic benchmark with facts that are both true and known to be missing from Freebase, DBpedia, and YAGO (Sec. 5.5), obtained from [18]. Second, we experiment with tables from Wikipedia for which a state-of-the-art web table annotation tool [17] fails to produce any output (Sec. 5.5.7). Our evaluation confirms our hypothesis and suggests our method can be *used in conjunction* with previous work to lead to better Web table understanding tools.

5.3 Building Language Models

Achieving accurate results in our setting requires LMs derived from a large corpus of phrases that are relational, grammatical, and frequent (so that they are likely to match evidence gathered at prediction time). Thus, we use the English subset of ClueWeb09 and the 5 billion annotations provided by Google's FACC1 corpus,[1] indicating which text spans contain mentions of entities known to Freebase, identified through their m-ids.

Replacing actual mentions to named entities in the text by their corresponding m-ids, we arrive at content such as the following:

`/m/06mr6` famously starred as `/m/06k5xq` besides `/m/0clpml`

Since in Freebase the entities `/m/06mr6` (actor Sir Sean Connery) and `/m/06k5xq` (fictional character Robin Hood) are related through relation `film/actor/.../character`, we add the phrase "famously starred as" to the language model of that relation (and also to the models of all other relations between these entities).

In a nutshell, building LMs boils down to: for every pair of m-ids that belong to a relation, extracting all phrases connecting those m-ids from the corpus, filtering uninformative phrases and aggregating the counts accordingly. Next, we explain the filtering steps we perform to increase the quality of our language models.

[1] http://lemurproject.org/clueweb09

5.3.1 Filtering Phrases

Our goal is to predict relations between *pairs of entities* such as family and romantic relationships between people, employment relationships between people and organizations, and business relationships among organizations. In order to keep our LMs highly focused we discard generic and uninformative phrases with the help of lightweight natural language processing tools.

Filtering Uninformative Phrases Not all phrases connecting entities are useful for relation prediction. For example, in the sentence above, the phrase "famously starred as" describes an actual relation between the surrounding entities while the phrase "besides" between `/m/06k5xq` (Robin Hood) and `/m/0clpml` (James Bond) does not. To filter out such noise, we parse the sentences containing pairs of m-ids and check if the phrases connecting the entities conform to known grammatical patterns that describe binary relations [1], discarding those that do not. This step eliminates the vast majority of the phrases but ensures our language models are grammatical and predictive.

Placeholder Generalization We often can (and should) generalize the phrases that reveal the same relation but differ in some detail. For example, phrases "starred in the 3rd movie of" and "starred in the first movie of" express the same relation, and are generalized into "starred in the ORD movie of", where "ORD" stands for any ordinal

Rule	Example
A? N ,?	*Apple* new leader *Cook*
, A? N P A?	*Cook* , new leader of *Apple*
,? C? S A? N P? A? ,?	*Apple* and its leader *Cook*
(, W)? A? V A?	*Cook* , who leads *Apple*
(, W)? A? V A? P A?	*Apple* is led by *Cook*
(, W)? A? V A? N A? P A?	*Cook* is the leader of *Apple*

Table 5.1: Rules for relational fragments from [1]. Category A contains a sequence of adjective, adverb, particle, modal and determiner; N is used to express a sequence of nouns; V is used for verb sequences; P contains a single preposition; S can be a possessive pronoun, wh-pronoun or ending; C is coordinating conjunction; W are Wh-pronoun ("who", "where").

number. We apply similar generalizations to instances of other common generic types such dates, distances and numbers. Fig. 5.8 summarizes the placeholders used with the relative frequency (i.e. number of phrases).

Further Filtering We are not interested in ontological relations that describe class membership of entities, such as that Sir Sean Connery is an actor and that James Bond is a fictional character. Other prominent ontological relations concern the ethnicity of people, the business segment of organizations, etc. Thus, we discard phrases that are variants of the "is a" pattern, often used in these cases (e.g., "is a British actor" or "was an American activist").

5.3.2 Phrase Statistics

We were able to find 19M distinct pairs of m-ids that are connected by a (filtered) relational phrase in the ClueWeb09 corpus. Only 1.4M of these pairs (8%) belong to one of the approximately 5K Freebase relations. In total, these 1.4M pairs are related through 2.36M distinct phrases in the corpus. Although we found some fairly long phrases, the majority of them are relatively short (4.3 tokens per phrase on average). As expected, we observed that the distribution of phrases by frequency in the corpus follows a power-law.

5.3.3 Specializing LMs Based on Type

The Freebase ontology specifies the *expected* types of the entities that can participate in any given relation. For example, relation `/film/film/subject`, that describes the subject of a movie, has domain `/film/film` and range `/film/film_subject`. Although somewhat informative, these types are fairly generic. For example, the subjects of biographical movies are people (and thus instances of `/people/person`), while the subjects of documentaries can be organizations or locations. Note however, that the LMs for these different kinds of movies are likely to be very different: the relational phrase "is the biography of" is appropriate for movies whose subject are people, while "portrays

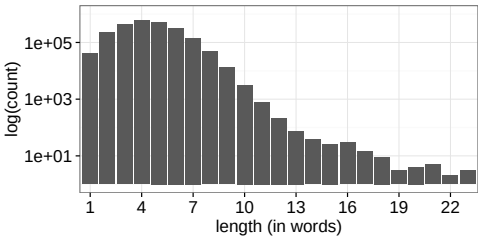


Figure 5.5: Histograms of phrase length (in words) as a function of their support in our phrase set.

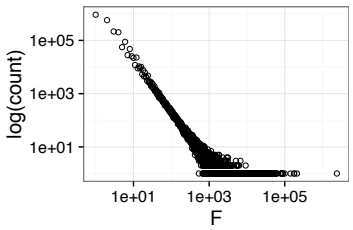


Figure 5.6: Phrases frequency distribution in the phrase set.

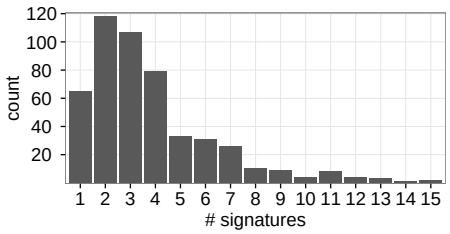


Figure 5.7: Distribution of relations per type signatures.

Global Statistics	
annotated pairs of m-ids	19 M
annotated pairs of m-ids related in KG	1.4 M
relations with phrases	2739
distinct phrases	2.36 M
Filtered Models Statistics	
relations with at least 200 phrases	500
LMs obtained	1934
filtered phrases	27.7 K

Figure 5.8: Statistics obtained after the filtering process.

the founding of" is suitable for movies about organizations. In order to account for such nuances, we partition the phrases associated with each relation based on generic entity types that can be inferred automatically by typical NER systems[2], and are available in Freebase as `/people/person`, `/organization/organization`, and `/location/location`. A catch-all "misc" type is used for all the other entities. This results in each FB relation having up to 16 different LMs, one for each possible combination of types. Fig. 5.7 shows the distribution of relations by the number of type signatures they have.

5.3.4 Model Statistics

In the end, of the 4819 relations in Freebase, we are able to build models for 2739. For the purposes of the evaluations reported here, we experimented only with those relations for which we could find at least 200 distinct phrases. This corresponds to 500 relations and 1934 different models (on average 3.78 LMs per relation, based on the combination

[2]Person (PER), Location (LOC), Organization (ORG), and Miscellaneous (MISC) as defined by Stanford NER [104].

of NER types). Fig. 5.8 summarizes statistics about the number of language models we derived from ClueWeb09, and those we use in our experimentation.

5.4 Relation Ranking

This section gives details the steps involved in relation ranking.

5.4.1 Gathering Evidence from the Web

Given an entity pair $\langle s_1, o_1 \rangle$, we perform a Web search looking for sentences mentioning both entities in the given order. For the purposes of this paper, we collected sentences from *snippets* returned by Google, saving time and bandwidth. The actual scoring of relations is done on relational phrases, extracted from the snippets, that match those used to build the language models. That is, we process the snippets in the same way described in Sec. 5.3.1. If multiple relational phrases are found in the same snippet our system uses all of them. Also, we attempt to minimize the effects of geo-localization and personalization in Google searches by periodically obtaining a new IP address via a Private Virtual Network service. Of course, our system is not restricted to Google. In fact, a local index of a large Web crawl (e.g., ClueWeb or the Web commons crawl) could be used instead of Google for this step.

We compare two strategies for matching phrases: exact matching (equality) and shallow approximate matching, which works as follows. Given a span of text we find candidate phrases using n-grams at character level (3-grams) and then we score them using a Fuzzy Jaccard Similarity [90] that takes into account fuzzy matchings between the individual words[3]. The approximate match comes with a clear precision and recall trade-off: it introduces noise but results in more phrases being matched. For example, the text "filmed entirely in" in a Web search snippet could match phrases "was filmed in" and "were filmed by", belonging to different LMs. We investigate this trade-off and

[3]We use Jaro-Winkler similarity with a threshold of 0.9.

confirm in the experimental evaluation that approximate matching generally improves the quality of the rankings.

5.4.2 Ranking for Individual Entity Pairs

As mentioned in Sec. 5.1.2, we take an IR approach to rank KG relations based on their relevance for an entity pair. To recap the notation and avoid confusion, each "document" D corresponds to a KG relation and a "query" Q corresponds to relational phrases connecting entities obtained through a Web search. Given an entity pair $\langle s_i, o_i \rangle$, we denote by $a(\langle s_i, o_i \rangle)$ the ranking of all documents for the query resulting from that pair, computed according to a $score(\cdot, \cdot)$ function (explained below).

Query Likelihood Scoring The *query likelihood* retrieval model assumes that the query terms are samples drawn from a LM derived from a document. Formally, given query Q and document D, from which a model θ_D is derived, we rank the documents by decreasing score, defined as:

$$score(Q, D) = P(Q|\theta_D).$$

Many approaches have been used for estimating $P(Q|\theta_D)$. We start with the robust multinomial language model, which assumes that terms are generated independently, and avoid overfitting with interpolation (i.e., Jelinek-Mercer smoothing) [91]. More precisely, let C be a document containing all the phrases and S the set of phrases. Then:

$$score(Q, D) = \prod_{p \in S} P(p|\theta_D)^{c(p,Q)} \tag{5.1}$$

where:

$$P(p|\theta_D) = \lambda\, P(p|D) + (1 - \lambda)\, P(p|C) \tag{5.2}$$

$$P(p|D) = \frac{c(p, D)}{|D|} \quad \text{and} \quad P(p|C) = \frac{c(p, C)}{|C|} \tag{5.3}$$

Above, $c(p, \cdot)$ denotes the frequency of phrase p in the query Q, document D or corpus C. We call this ranking approach PLM, for *Phrase Language Model* in the experimental evaluation.

We set $\lambda = 0.9$ experimentally.

Disjunctive Gate Scoring The query likelihood approach uses a *conjunctive gate* to combine evidence from multiple phrases while predicting the likelihood of a relation: a model that cannot generate most phrases in the query is unlikely to rank high. In our setting, this is often an overkill. For example, one can be reasonably certain that the relation `/film/film/featured_film_locations` holds for a pair of entities connected by the phrase "was filmed in". An implicit assumption here is that the frequency of the phrases used to build the LMs and the queries is a good proxy for how reliable they are. While this seems reasonable at Web scale, one could take the trustworthiness of the sources [8] into account, e.g., via re-ranking or by a priori filtering.

To allow for more permissive predictions we calculate the score of each relation interpolating its prior and its posterior probability conditioned by every single phrase in the query. We aggregate the posterior of each phrase using a "noisy-OR" gate [105]:

$$score(Q, D) = \beta\, P(D|p_1, ..., p_Q) + (1 - \beta)\, P(D) \tag{5.4}$$

where:

$$P(D|p_1, ..., p_Q) = 1 - \prod_{p \in Q} (1 - P(D|p)) \tag{5.5}$$

$$P(D|p) = \frac{c(p, D)}{\sum_{l \in L} c(p, D_l)} \tag{5.6}$$

A "noisy-OR" gate combines evidence differently from the standard PLM: a relation scores high if the query contains *any* of the high frequency phrases associated with that relation. We call this ranking approach Noisy OR in the experimental evaluation.

As λ for the standard model, β is a coefficient to control the amount of smoothing. We set it experimentally to $\beta = 0.8$.

5.4.3 Ranking for Multiple Pairs

We now move on to the general form of the problem which applies to a set of entity pairs $\langle s_1, o_1 \rangle, \ldots, \langle s_n, o_n \rangle$, e.g., coming from different rows of the same table, and in

which we need to rank KG relations in some aggregate form. We consider two ways of producing an aggregate ranking. The first is a *global* approach where we merge the results of the Web searches for each individual pair into a single query, used to rank all KG relations, while the second is a *local* aggregation approach, where we score KG relations by merging the individual rankings obtained for each pair separately.

Global Aggregation In this aggregation approach we build a query Q' containing all phrases in each Q_i derived from entity pair $\langle s_i, o_i \rangle$, with the appropriate phrase counts, and obtain a single ranking using Eq. 5.1 or Eq. 5.4 to produce the final answer, depending on the scoring model used.

Local Aggregation In this approach we first rank all relations for each of the entity pairs, and combine these rankings to score the relations. Let $a_i = a(\langle s_i, o_i \rangle)$ be the ranking obtained for entity pair $\langle s_i, o_i \rangle$, computed according to Eq. 5.1 or Eq. 5.4 depending on the scoring model:

$$a_1 = a(\langle s_1, o_1 \rangle) = r_1^{a_1}, r_2^{a_1}, \ldots, r_k^{a_1}$$
$$\vdots$$
$$a_n = a(\langle s_n, o_n \rangle) = r_1^{a_n}, r_2^{a_n}, \ldots, r_j^{a_n}$$

The *local* aggregated score of a relation is its mean (inverse) rank across all individual rankings:

$$agg_{score}(r) = \frac{1}{n} \sum_{a_i} \frac{1}{rank(r, a_i)} \tag{5.7}$$

In this way, a relation that ranks high for a large number of pairs will have a higher score and rank high for the set of entity pairs.

5.5 Experiments

We now report on an experimental evaluation of the LM-based relation ranking approach to show it does not suffer from the KG incompleteness problem and, thus, can be a viable alternative to Entity Linking (EL) approaches. To do that, we experiment first

on two corpora of facts involving pairs of in-KG entities, comprising 9 relations from the person domain (Tab. 5.2 shows the relations). The first corpus, called LectorFacts, consists of facts *known to be missing* from DBpedia and Freebase. The second corpus, called KGFacts, comprises the same relations as LectorFacts, but with facts that are present in both DBpedia and Freebase. With these two corpora, we can simulate the scenarios where EL methods should work (KGFacts) and the scenario where they would not (LectorFacts). We thoroughly evaluate our system both with individual pairs of entities (Sec. 5.5.1) and also with sets of pairs (Sec. 5.5.2) as input, under a variety of scenarios. Then, to further illustrate the KG incompleteness issue, we evaluate our approach on pairs of columns from Wikipedia tables mixing in-KG and out-of-KG entities but for which a state-of-the-art EL method, T2K Match [16, 58, 17], fails to identify the correct relations (Sec. 5.5.7).

For the experiments reported here, we trained our method was to predict 500 different Freebase relations (recall Sec. 5.3.4) and used Google for the Web search step.

Statistics about LectorFacts and KGFacts To the best of our knowledge, no previous benchmark for Web table understanding considers the case where KG incompleteness prevents an EL approach to predict a good relation even when all entities are in the KG. Therefore, we rely on the only corpus of facts *known to be missing* from mainstream and publicly available KGs [18][4], and call it LectorFacts here. We restrict our evaluation to the 9 non-ontological relations in the original corpus[5], and use 50 facts (i.e., entity-pairs) from each relation. For the sake of comparison, we created a similar benchmark, KGFacts, by randomly picking 50 entity pairs from each of the relations in LectorFacts, among those pairs that appear in both DBpedia and Freebase. In a sense, these two benchmarks complement each other: facts from KGFacts concern prominent pairs of entities and generate more hits on a Web search. As shown in Tab. 5.2, on average, we are able to obtain twice as many hits (the "sent." column in

[4] Available at: http://downloads.dbpedia.org/2016-04/ext/lector_facts/

[5] `people/person/nationality`, `people/person/religion` and `people/person/ethnicity` are ontological relations and thus ignored.

the table) on that corpus compared to LectorFacts. The two columns under "phrase" in the table show the (average) number of relational phrases (i.e., the ones used to build the LMs) that *match* the sentences returned by the Web search. These phrases are used (as queries) to predict the relations in our approach. We experiment with exact and approximate matching of sentences and relational phrases. For clarity, all results reported use exact matching, except those in Sec. 5.5.3.

Metrics Given a set $I = \{\langle s_1, o_1\rangle, \ldots, \langle s_n, o_n\rangle\}$ of subject-object entity pairs we have only one correct answer (the pairs are labeled with a single relation). In order to evaluate the ranking produced by the system we use the reciprocal rank [106] that corresponds to the multiplicative inverse of the rank of the correct relation. More precisely, let $a(I) = r_1, \ldots, r_k$ be the response of a prediction for the input pairs I in which relations are given in decreasing order of relevance, and let $truth(I)$ be the ground truth relation for I, the $reciprocal_{rank}(a, I)$ is:

$$reciprocal_{rank}(a, I) = \sum_{i=1}^{k} \frac{\mathbb{1}(truth(I) = a[i])}{i} \tag{5.8}$$

where $\mathbb{1}(\cdot)$ is the indicator function (returns 1 if the condition appearing as its argument holds, 0 otherwise) and $a[i]$ is the relation in position i in the ranking. Note that the metric implicitly takes recall into account. Indeed if the system does not predict any ranking or the correct relation is not present the reciprocal rank is 0. Finally, we use the Mean Reciprocal Ranking (MRR) [106] to evaluate the results of multiple input sets $\mathcal{I} = I_1, ..., I_n$:

$$MRR(\mathcal{I}) = \frac{1}{|\mathcal{I}|} \sum_{I_j \in \mathcal{I}} reciprocal_{rank}(a, I_j) \tag{5.9}$$

5.5.1 MRR on Individual Entity Pairs

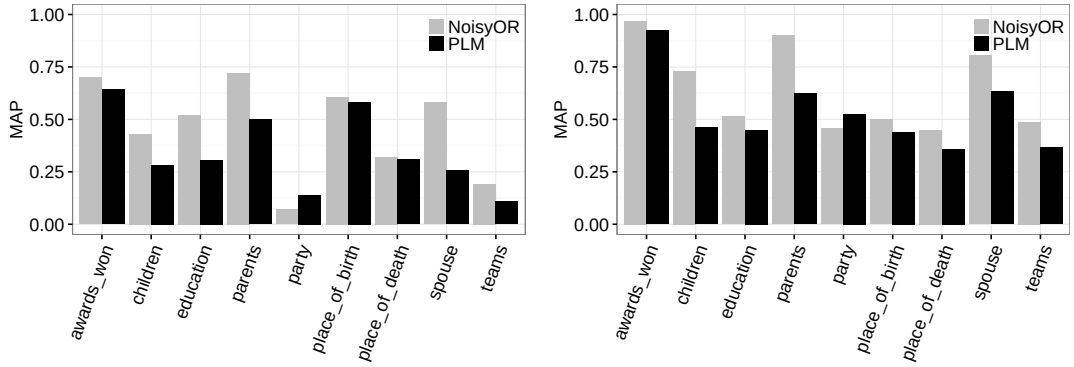
Fig. 5.9 shows the $MRR(\mathcal{I})$, per relation, obtained on predicting the relation for each of the 50 pairs individually. As expected, relation prediction on KGFacts is easier than on LectorFacts regardless of the ranking model. Quantitatively speaking,

Relation	LectorFacts			KGFacts		
	sent.	phrase		sent.	phrase	
		ex.	ap.		ex.	ap.
people/person/parents	39.5	2.8	4.9	93.5	8.6	16.6
people/person/education	22.3	1.3	2.2	113.3	8.6	16.8
sports/pro_athlete/teams	61.4	3.5	6.3	131.4	17.3	29.9
people/person/place_of_death	55.1	1.5	3.3	126.3	10.9	21.5
government/politician/party	37.7	1.6	2.5	102.8	11.0	21.9
people/person/place_of_birth	54.2	2.5	4.4	119.9	9.7	18.5
award/award_winner/awards_won	58.1	2.7	6.1	94.2	8.2	15.0
people/person/spouse	49.0	2.7	5.5	84.2	8.5	15.4
people/person/children	43.2	2.0	4.1	82.0	7.1	14.2
Mean	46.7	2.3	4.4	105.3	10.0	18.9

Table 5.2: Mean number of sentences retrieved from the Web search and corresponding number of matching phrases obtained with the exact and the approximate method. Each relation consists of 50 entity pairs.

Noisy OR was 20% more effective than PLM on the KGFacts corpus (MRR of 0.64 and 0.53, respectively), and 31% more effective on LectorFacts (MRR of 0.46 and 0.35, respectively). This can be explained by the larger number of relational phrases that can be obtained with pairs of entities in KGFacts (recall Tab. 5.2). Looking at the MRR results across relations, one can see that some relations are harder to predict than others. This is explained by the ambiguity of the phrases in some language models, as shown by the snippets exampled in Table 5.3. For example, "is the daughter of" is almost exclusively used in `people/person/parents` (5.3(c)). Similarly, descriptive phrases like "won the" and "was awarded the" are strongly associated with `award/award_winner/awards_won` (5.3(g)). Other relations are expressed through generic phrases that can only be interpreted in context, such as "leader of the", "led the" or "left the" which appear in the models of `.../person/employment`, `.../politician/party` or `.../pro_athlete/teams`.

It should be noted that the difficulties of dealing with ambiguity are more pronounced because the system evaluated here is configured to predict 500 different relations from all Freebase domains. Much better results are to be expected if one learns models for domain-specific relations.



(a) Ranking relations on LectorFacts (b) Ranking relations on KGFacts

Figure 5.9: Single-pairs experiment on both the ground truths and the models. All the results are provided matching the phrases exactly.

5.5.2 MRR on Multiple Pairs

Fig. 5.10 shows the MRR for each combination of: corpus (LectorFacts and KGFacts), scoring model (Noisy OR and PLM), and method for combining evidence (local aggregation and global query), when considering multiple entity pairs. We vary the size of the input set from 3 to 20 pairs, and each plot is the average of 10 different samples. We cap the size to 20 pairs as previous work has reported that this is the average number of rows found in real Web Tables [58]. Several general observations are possible from the figure: (1) using multiple pairs for relation ranking has a significant positive impact on MRR; (2) aggregating pairwise rankings is more robust, regardless of scoring model; and (3) Noisy OR generally outperforms the standard language model PLM. Next, we discuss these findings in more detail.

More Entity Pairs is Better Unsurprisingly, the more entity pairs we use, the higher the MRR. To quantify this, the highest MRR achieved with individual pairs was 0.64, obtained with the Noisy OR on KGFacts. Testing the same model on the same corpus with local aggregation, we achieve MRR of 0.84 using 3 pairs and 0.98 using 20 pairs, corresponding to improvements of 31% and 53%, respectively. Looking at LectorFacts with Noisy OR and local aggregation, even more pronounced gains are

(a) sports/pro_athlete/teams

phrase	$P(p\|D)$
playing for	0.0483
playing at	0.0240
gave	0.0152
led the	0.0120
joined	0.0115
put	0.0092
has joined	0.0062
scored for	0.0059
...	...

(b) people/person/place_of_birth

phrase	$P(p\|D)$
was born in	0.4103
a native of	0.0347
grew up in	0.0178
was born at	0.0158
who was born in	0.0126
left	0.0071
s hometown of	0.0067
is from	0.0063
...	...

(c) people/person/parents

phrase	$P(p\|D)$
son of	0.1335
the son of	0.1257
s father	0.0931
daughter of	0.0667
s mother	0.0589
is the daughter of	0.0392
was the son of	0.0361
was the firstborn of	0.0141
...	...

(d) people/person/education

phrase	$P(p\|D)$
graduated from	0.0692
attended	0.0685
attended the	0.0336
graduated from the	0.0298
is a graduate of	0.0216
studied at the	0.0186
was educated at	0.0184
entered	0.0161
...	...

(e) people/person/children

phrase	$P(p\|D)$
s son	0.1726
begat	0.1249
s daughter	0.0738
blessed	0.0437
the father of	0.0308
father of	0.0197
became the father of	0.0191
the mother of	0.0167
...	...

(f) people/person/place_of_death

phrase	$P(p\|D)$
died in	0.0784
was born in	0.0509
moved to	0.0318
returned to	0.0316
bishop of	0.0112
died at	0.0107
lived in	0.0105
was assassinated in	0.0103
...	...

(g) award/award_winner/awards_won

phrase	$P(p\|D)$
won the	0.1555
was awarded the	0.1244
received the	0.0795
who won the	0.0319
winner of the	0.0235
won	0.0205
shared the	0.0160
was honored with the	0.0147
...	...

(h) government/politician/party

phrase	$P(p\|D)$
leader of the	0.0321
the leader of the	0.0158
won the	0.0140
said the	0.0097
has written a letter to	0.0094
joined the	0.0087
head of the	0.0081
s speech to the	0.0073
...	...

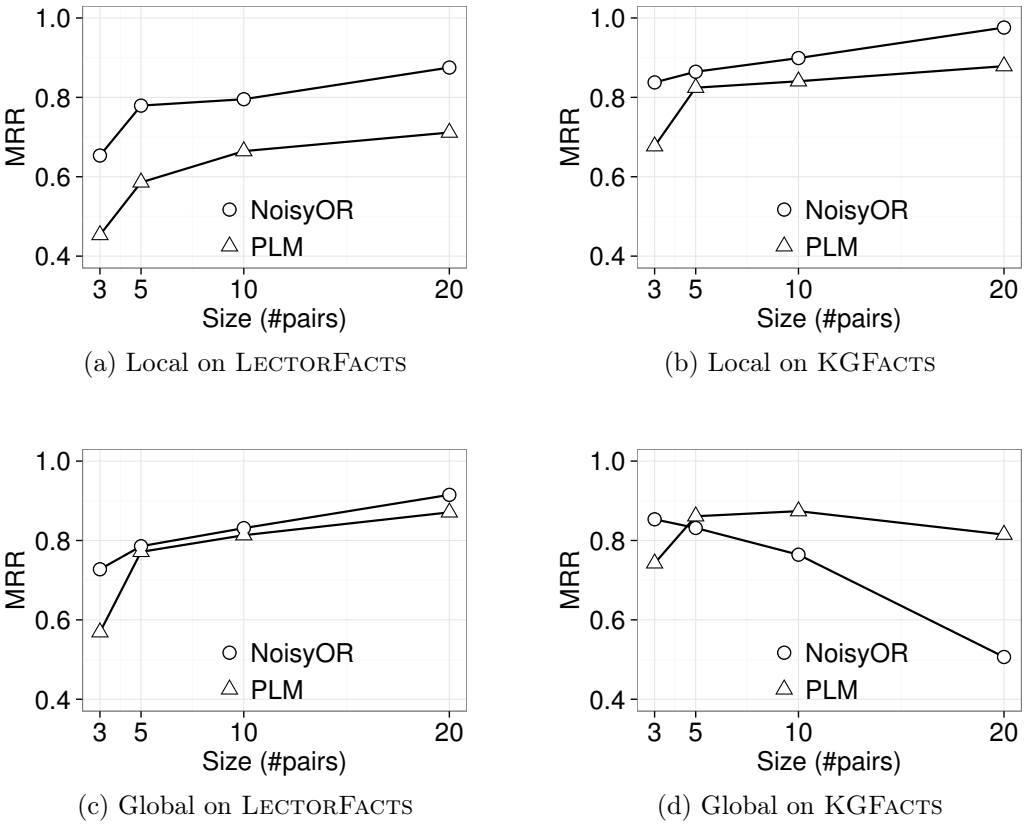
(i) people/person/spouse

phrase	$P(p\|D)$
married	0.1017
s wife	0.0838
s husband	0.0344
wife of	0.0271
met	0.0172
s marriage to	0.0162
the wife of	0.0147
is married to	0.0141
...	...

Table 5.3: Snippets of LMs for the relations of interest in the evaluation.

evident: 43% with 3 pairs and 89% with 20 pairs. The highest MRR are obtained with 20 entity pairs and the Noisy OR model: 0.98 on KGFacts with local aggregation, and 0.91 on LectorFacts with global queries. We conjecture on this discrepancy below. These results underscore the high potential of our method for Web table understanding, where predictions are made on multiple entity pairs, and not just one.

To see how multiple pairs help relation scoring, suppose the input is a table with soccer players and the teams they played for. If we are given just the pair ⟨`"Luis Enrique"`,`"FC Barcelona"`⟩, we will not be able to discern whether `.../pro_athlete/teams`



(a) Local on LectorFacts (b) Local on KGFacts

(c) Global on LectorFacts (d) Global on KGFacts

Figure 5.10: MRR on sets of entity pairs.

or `soccer/.../manager` should rank higher as both relations apply to that pair. However, if the input also contains another pair like ⟨`"Neymar Jr."`,`"Paris St Germain"`⟩, we will be much more likely to predict `.../pro_athlete/teams`. In other words, the ambiguity of the input reduces with more pairs, as expected.

Local Aggregation is More Robust When it comes to the local aggregation of multiple pairwise predictions versus a single prediction using a global query formed by grouping all sentences from the Web searches, there seems to be a dependence on the actual number of phrases that are used: with a small number of phrases (e.g., as in LectorFacts), it is better to use the global approach while with many phrases it is much better to use the local aggregation. Using the AUC of the plots in Fig. 5.10 as a proxy, we can quantify this argument: for LectorFacts, the global approach is

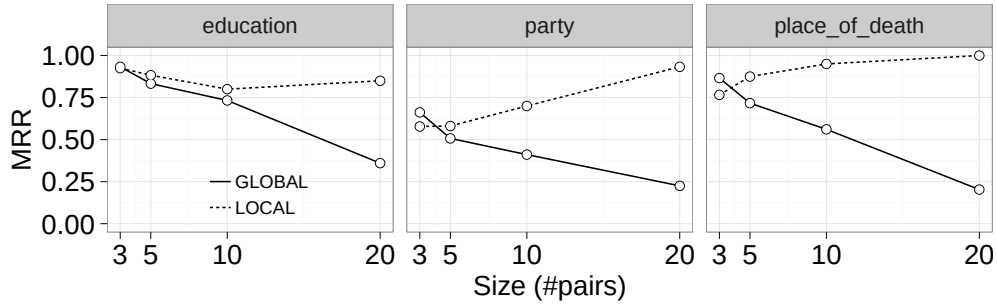
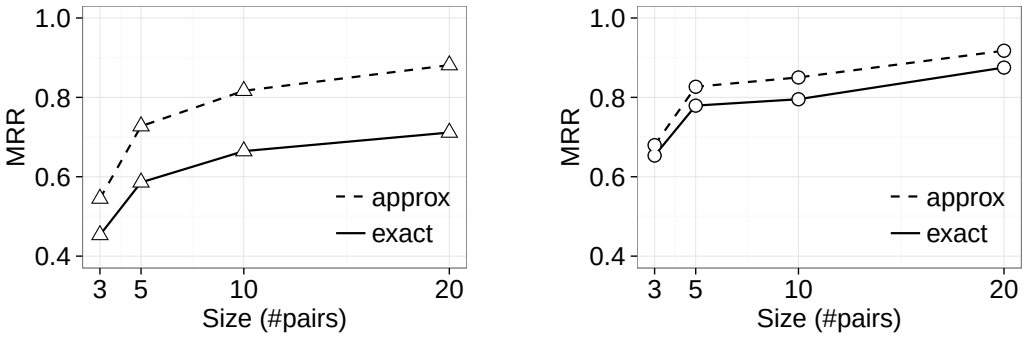


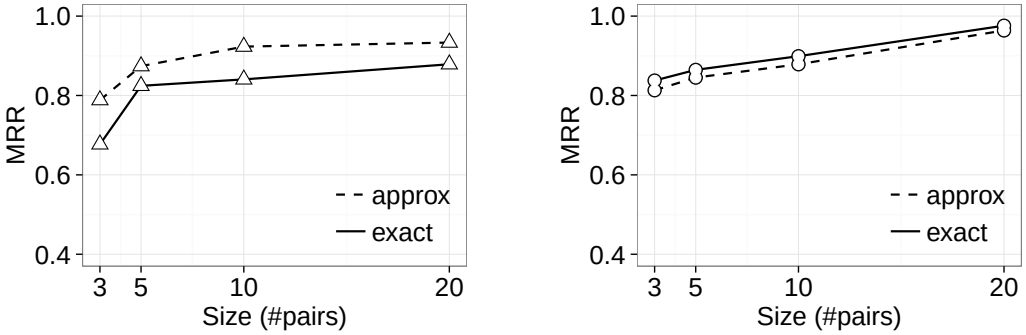
Figure 5.11: Detailed trend of NOISY OR on KGFACTS for three specific relations considering local aggregation (dotted lines) and global (solid lines).

superior by 4% (Fig. 5.10(a) and (c)) while for KGFACTS the local aggregation is better by 7% (Fig. 5.10(b) and (d)). As a matter of fact, the global aggregation method gets worse on KGFACTS as more entity pairs are used. In Fig. 5.11 we report the specific results obtained from a local (dashed line) and global (solid line) aggregation for three different relations. The trend is similar for all the three cases: with more pairs there are more (and more diverse) sentences, which leads to the global query to lose focus. For example, looking at some entity pairs in relation `people/person/education` we can find phrases such as "who became president of", "is the co founder of" or "played football at" that are associated (sometimes very strongly) with many relations. The problem is more evident with the NOISY OR model as it does not take phrase frequency into account and thus does not have any way of weighing importance of the phrases in the query. To verify this hypothesis, we pruned all but the 5 most frequent phrases in the queries used in Fig. 5.10(d), and scored these pruned queries with NOISY OR, resulting in a 5% improvement.

NOISY OR is Better A comparison of the AUC of the plots in Fig. 5.10 reveals that scoring with NOISY OR is generally superior to that with PLM. In quantitative terms, the relative improvements can be as high as 24% (see Fig. 5.10(a)) and considering all the configurations NOISY OR obtains a relative gains of 20% over PLM. In conclusion, a NOISY OR scoring model is very robust if applied with a local aggregation approach independently from the popularity of the entities involved.



(a) PLM on LectorFacts (b) Noisy OR on LectorFacts



(c) PLM on KGFacts (d) Noisy OR on KGFacts

Figure 5.12: Approximate (dashed lines) versus exact (solid lines) matching on multiple entity pairs with local aggregation.

5.5.3 Matching Phrases Approximately

All results shown so far were obtained by *exactly* matching sentences from the Web search against relational phrases in the LMs (to form the queries used for relation ranking). Because exact matching leads to low hit counts (sometimes, even empty queries), we experimented with alternative ways of approximately matching sentences to relational phrases, settling for the scheme described in Sec. 5.4.1. Fig. 5.12 shows that approximate matching (dashed line) generally improves on exact matching (solid line). For clarity, we show the results with local aggregation only. In quantitative terms, comparing the AUC of the plots, we observe the biggest improvement (19%) for the PLM scoring. This improvement can be attributed to the fact that approximate matching doubles the number of hits as shown in Tab. 5.2 (see the *phrase* columns).

It is worth noting that approximate matching closes the gap in MRR between KGFacts and LectorFacts considerably. Comparing the AUC between plots, the best setting for KGFacts (Noisy OR with local aggregation and exact phrase matching) is only 6% superior to the best setting for LectorFacts (Noisy OR with local aggregation and approximate phrase matching).

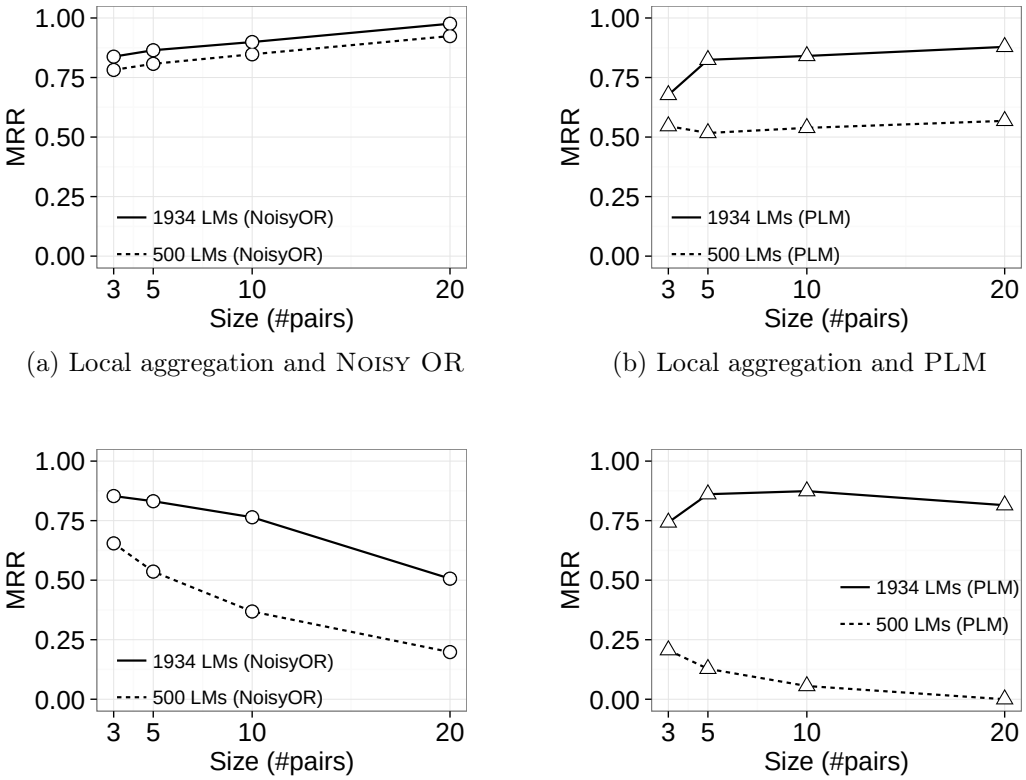
5.5.4 Specializing LMs

We report an experiment to show the advantages derived from creating multiple language models for each KG relation. Indeed, as described in in Sec. 5.3.3, we specialize the LMs on possible different pairs of NER types creating multiple LMs to represent the same relation. In particular, we created 1934 LMs to represent 500 KG relations (Fig. 5.7). Fig. 5.13 compares the results that we have obtained on KGFacts with the configurations described so far (1934 LMs) with the results that we would have obtained using a single language model for each relation (500 LMs). Specializing the LMs improves the system in all its configurations. In particular, the difference is not high when the query is very focused, as in the local aggregation approach (Fig. 5.13(a)), especially if we are using Noisy OR, a model that is driven by a few significant phrases. The main advantages arise when the query misses its focus as shown in Fig. 5.13(d) with a global aggregation and PLM. In this case, when we increase the number of pairs we create ambiguous queries, so that having multiple LMs that describe the same relations helps the system in avoid wrong predictions.

5.5.5 Pruning LMs

While our concern in this work is effectiveness, we report on a brief experiment on the natural trade-off between the LM sizes, inference time and accuracy. Tab. 5.4 shows the MRR and the inference time of the two models on samples of 20 pairs from LectorFacts for different values of the minimum *support* (F in the table) required for a phrase to be used in any language model[6]. Unsurprisingly, increasing the minimum

[6]For clarity, all other results reported here are on LMs with F =1.



(a) Local aggregation and Noisy OR

(b) Local aggregation and PLM

(c) Global aggregation and Noisy OR

(d) Global aggregation and PLM

Figure 5.13: Accuracy obtained by specializing the LMs on different combinations of type (solid lines) versus using a single LM for each relation (dashed lines).

support leads to fewer and smaller models, and, consequently: a drop in MRR, and a drop in inference time. For the Noisy OR scoring model, however, the trade-off is quite favorable: with F =100 there is no drop in MRR while the inference time is cut in half. This is easily explained by the power-law distribution of phrases in the ClueWeb09 corpus (Sec. 5.3.2) and the fact Noisy OR relies on phrases that are *strongly* associated with a relation, which inevitably ignores phrases with low support.

5.5.6 Towards Practical Tools

The observations in the experiments above indicate that the LM-based relation ranking can perform well enough to be useful in practice. Moreover, they indicate that a configuration of Noisy OR with local aggregation and approximate sentence matching

F	PLM		Noisy OR	
	MRR	time (ms)	MRR	time (ms)
1	0.71	230.0 ±11.1	0.88	78.5 ±11.0
10	0.70	180.5 ± 3.5	0.87	78.1 ± 7.1
100	0.55	178.1 ± 4.6	0.88	44.1 ± 2.0
1000	0.14	167.4 ± 3.5	0.85	39.1 ± 1.3

Table 5.4: MRR and inference time versus minimum phrase *support* (F).

to be very robust. One possible tuning would be to use the global query approach if the number of sentences returned by the Web search is below an application specific threshold.

5.5.7 Towards Web Table Understanding

We now report on a preliminary experiment to show that relation prediction with LMs can be a viable alternative to Entity Linking (EL) for Web Table Understanding. While EL methods fail on tables with entity pairs from LectorFacts, even though they concern in-KG entities, the issue is not artificially introduced by that benchmark. To show this, we collected actual Wikipedia tables containing in-KG and out-of-KG entities and facts from different domains and for which even the state-of-the-art EL method T2K Match [16, 58, 17] fails. Since an independent benchmark for this task is lacking, we manually extracted 80 pairs of columns from 65 different Wikipedia tables, for which T2K Match was unable to assign a relation. These pairs cover a range of topics and relations beyond LectorFacts, including geography, sports, actors and directors of movies and TV shows, among others. The average number of entity pairs in this test was 31. The average number of sentences retrieved by the Web search is 39. We tested our method with these 80 pairs and manually evaluated the results to determine if the predicted relations were appropriate. [7]

[7] All data used for this experiment, including the top-3 relations produced by each of our methods, can be found at https://ln.sync.com/dl/b7e5a9f40#rfyp47tn-irz5dvhn-bwmn58kr-cetrhxmj

EPG	EPL	ENG	ENL	APG	APL	ANG	ANL
35	38	43	49	32	41	28	52

Table 5.5: Relations correctly predicted (out of 80) on Web Table Understanding for different combinations of phrase matching (A-approximate, E-exact), model (N-Noisy OR, P-PLM) and aggregation approach (L-local, G-global).

We evaluate the results using precision@1 (1 if the first relation in the output ranking is correct, 0 otherwise) of each pair. Tab. 5.5 shows the number of entity pairs for which the top ranked relation was judged appropriate, for different configurations. The best results (52 relations out of 80) are obtained with approximate matching (A), using the Noisy OR score model (N) and locally aggregating the multiple pairs (L), which is also the best configuration on LectorFacts. Developing a proper benchmark for this task is non-trivial and important future work for this area.

5.6 Re-ranking with Entity Types

From the experimental evaluation it is clear that handling an high number of relations is one of the main challenges for our approach. A possible improvement would be to introduce the types of the entities in the prediction, which can help in differentiate possible overlap between LMs of different relations because of ambiguity in natural language. For example, the relational phrase "joined" applies to a person being drafted by a sports team or being hired by a company. In such cases, knowing the *types* of the input entities can go a long way in deciding which relations are more likely to apply, through strict filtering or a less drastic re-ranking step.

We report here some preliminary results that we obtain introducing a further re-ranking step. Essentially, after the prediction we *re-rank* the ranking of relations considering the similarity between the types of the input entities ($\langle s_i, o_i \rangle$) and the expected types for the relations. Indeed, relations in Freebase are defined between a pairs of expected fine-grained types which express the entities that can participate: for example the relation `sports/athlete/team` is used to link an athlete (`sports/athlete`) to his

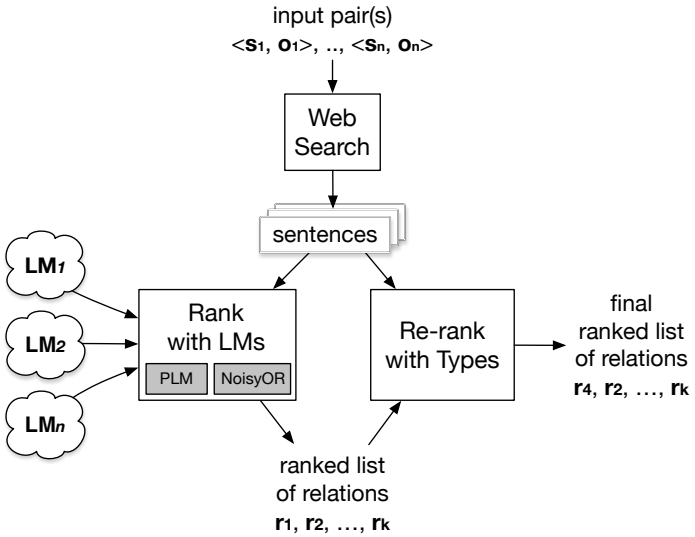


Figure 5.14: Introducing a re-ranking step.

team (`sports/team`).

5.6.1 Re-ranking Based on NER Types

Understanding the fine-grained types of a list of given entities is still an open problem [107, 108] and for this reason we provide an exploratory experiment that is based on using traditional NER types (person, location, organization and miscellaneous) that can be obtained from state-of-the-art NER tools such as Stanford NER [104]. Since some of the entity types can be expressed with multiple NER types (e.g. `/award/award_winner` most of the times a person, but sometimes an organization) we embed every Freebase type into a four-features vector, by associating the NER types of all the annotated pairs of entities from ClueWeb09 (see Sec. 5.3) with their specific fine-grained Freebase types. Let $\vec{r}_s$ and $\vec{r}_o$ denote the embeddings of the expected types for the subject and the object entities for relation r.

At prediction time, we also apply Stanford NER to the sentences returned by the Web search for each entity pair in the input (Fig. 5.14). Aggregating the relative frequencies for each of the base types across all sentences, we obtain two feature vectors:

$\vec{q}_s$, for the subject entities, and $\vec{q}_o$, for the object entities in the input entity pairs. Re-ranking is then done based on the cosine similarity of the respective vectors. In particular, let $a = r_1, \ldots, r_k$ be a ranking of the KG relations obtained according to any combination of the approaches discussed earlier (local or global, PLM or NOISY OR). The final ranking of each relation is produced as:

$$rank'(r) = \frac{1}{rank(a, r)} \cdot \cos(\vec{r}_s, \vec{q}_s) \cdot \cos(\vec{r}_o, \vec{q}_o). \tag{5.10}$$

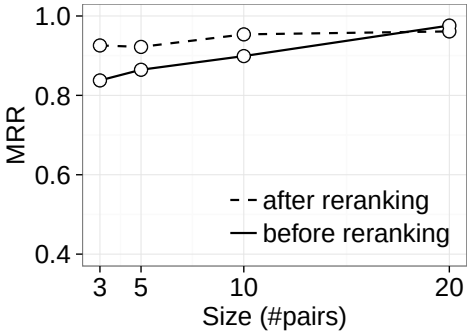
5.6.2 Preliminary Results

The plots in Fig. 5.10 show the impact of using re-ranking: the solid lines correspond to the MRR before re-ranking (they are the same showed previously) while the dashed lines represent the MRR after the re-ranking. A visual inspection shows that MRR improves after re-ranking (since the solid lines are always above the dashed lines).

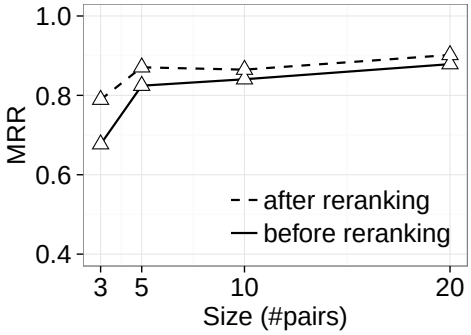
As discussed before, the highest MRR achieved for predicting relations using individual pairs was 0.64, obtained for the NOISY OR on KGFACTS (Fig. 5.9). When we query the system using 3 entity pairs (of the same corpus) we achieve MRR of 0.84 with the same model using a local aggregation (Fig. 5.10b or Fig. 5.15a) and 0.93 introducing a further re-ranking based on NER type (Fig. 5.15a), which represent improvements of 31% and 45%, respectively. Comparing the AUC for each plot in the figure and found that the re-ranking step can lead to a relative improvement up to 20%, as in the plot in Fig. 5.15f. On average across all tested settings, re-ranking improved MRR by 9%.

5.7 Conclusion

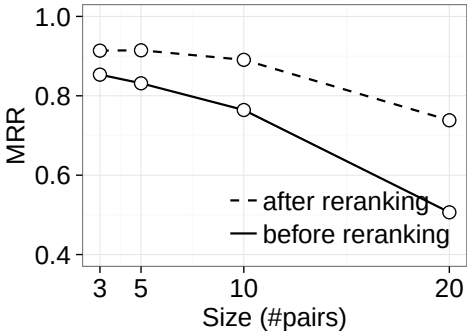
In this chapter we described a possible solution to the problem of predicting relations from a Knowledge Graph that hold over relational data found on the Web, such as tables, structured lists, CSV/TSV files, among others. The methods described here fill a gap left by text-based Information Extraction tools and overcome the main obstacle of table understanding methods relying on linking entities to objects in the graph.



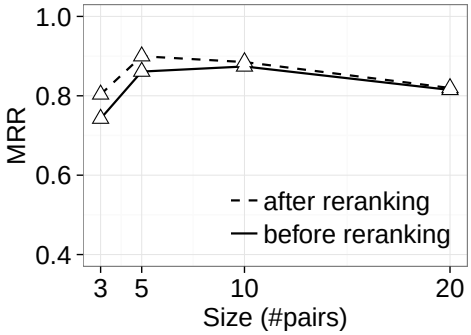
(a) KGFacts, local agg. and Noisy OR



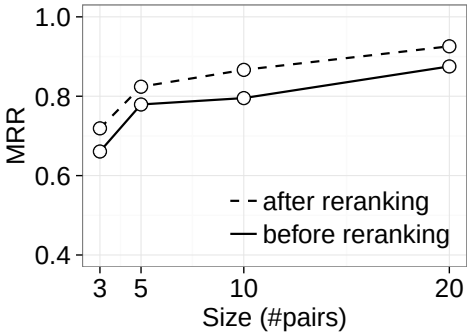
(b) KGFacts, local agg. and PLM



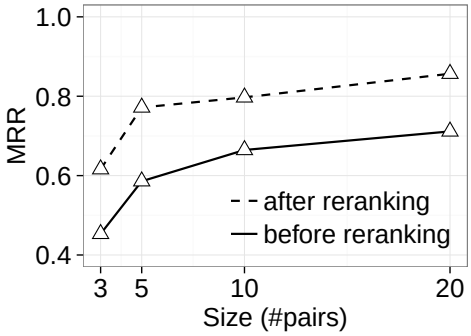
(c) KGFacts, global agg. and Noisy OR



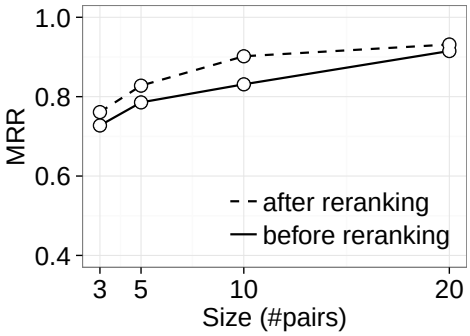
(d) KGFacts, global agg. and PLM



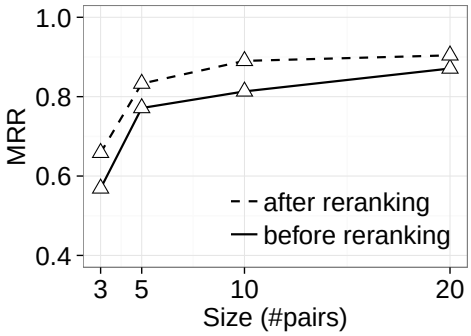
(e) LectorFacts, local agg. and Noisy OR



(f) LectorFacts, local agg. and PLM



(g) LectorFacts, global agg. and Noisy OR



(h) LectorFacts, global agg. and PLM

Figure 5.15: Comparison between accuracy obtained from all the models and ground-truth before re-ranking (solid lines) and after re-ranking (dashed lines).

Our approach borrows from Information Retrieval and Information Extraction work and uses generative language models to represent the relations in a KG with relational phrases, extracted from a Web-scale corpus of independently annotated text using the state-of-the-art in Open Information Extraction to filter out noise. Thus, relation prediction is done by ranking the respective models based on how likely they are to generate the phrases on the Web that connect the entities given as input. We evaluate different approaches for model ranking and for aggregating evidence to predict relations for a set of entity pairs. We also include some preliminary results to show how understanding the entity types can improve the overall approach and further evaluations with other standard ranking approaches are left for future work.

Finally, the experimental evaluation is designed to stress the issues caused by KG incompleteness, a limiting factor of the most related previous work. The encouraging results indicate the method overcome the KG incompleteness issue and that its accuracy is high enough to suggest it can be successfully applied on tasks such as KG construction and augmentation.

Chapter 6

Conclusion and Future Work

Knowledge on the Web is increasingly organized in so-called Knowledge Graphs (KGs), structured repositories that represent real world entities together with their semantic relationships in a graphical form. They cover many millions of entities with billions of facts and thousands of predicates. However, despite this impressive size, none of such KGs can be considered complete. In fact, while most of the existing KGs such as DBpedia, YAGO and Wikidata are semi-automatically extracted from the structured parts of Wikipedia, a majority of the published information still is inaccessible to these tools because encoded in the form of natural-language text only.

Many information extraction efforts are increasingly focusing on natural language as their primary source of extraction. The main goal is to turn textual information into a graphical form that can be queried or used to augment the content of existing KGs. To do so, the main challenge is to align the natural language to the schema of the KG. This thesis focuses exactly on this topic by investigating possible approaches to associate natural language text with relations of an existing KG, and use them to augment the KG with missing information.

We described Lector, a very effective method to augment state-of-the-art KGs with millions of accurate facts extracted from the text of Wikipedia articles. The approach relies on facts that are already present in a KG to select the appropriate typed textual

patterns used to express the corresponding relations in natural language. Our evaluation reveals that Lector produces over 1,8 M facts that are new to Freebase, YAGO, and DBpedia. Our results show that Lector can augment these KGs by as much as 10%, which is significant given that all such KGs are mature, resulting from 10 or more years of work now.

We described Stup, an approach that can be used to predict relations from a KG that hold over relational data found on the Web, such as tables, structured lists, among others. The approach uses generative language models to represent the relations in a KG with relational phrases, extracted from a Web-scale corpus of independently annotated text using the state-of-the-art in Open Information Extraction to filter out noise. Thus, relation prediction is done by ranking the respective models based on how likely they are to generate the phrases on the Web that connect the entities given as input. The experimental evaluation designed to stress the issues caused by KG incompleteness, a limiting factor of the most related previous work and the encouraging results indicate the method overcome the limits of existing approaches.

In both the projects we provide extensive evaluations and the high accuracy obtained suggests that they can be successfully applied on tasks such as KG construction and augmentation. After the success in the 2017 DBpedia TextExt Challenge, Lector will be integrated as part of the DBpedia core dataset. For the future, we have several directions on the general framework. For example, investigating more sophisticated NLP solutions for processing the text or generalize the patterns, always taking into account the multilingual perspective of the extraction tool. For Stup, an immediate line of future work consists in integrating the approach it into a self-contained Web Tables Understanding framework, through the process described by state-of-the-art tools. In terms of improving the methods, a primary target would be to extend the set of features used for relation re-ranking, possibly exploring word embedding models. Another interesting direction would extending our models to handle property relations that associate entities with literals and more complex relations.

However, an evidence resulting from both the projects is that there are many cases in which the natural language can not be aligned properly to the canonical schema of an existing KG, as many predicates are completely missing, or even too complex to be modelled. Since a very strict schema is not completely necessary in many KG applications, more open ended solutions in knowledge extraction are ready to be investigated. Thus, a natural evolution of this work will be to consider the possibility to exploit a larger variety of non canonicalized relations in capturing information from the same unstructured sources.

Bibliography

[1] Filipe de Sá Mesquita, Jordan Schmidek, and Denilson Barbosa. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 447–457, 2013. URL http://aclweb.org/anthology/D/D13/D13-1043.pdf.

[2] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250, 2008. ISBN 978-1-60558-102-6. doi: 10.1145/1376616.1376746.

[3] Denny Vrandečić. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, pages 1063–1064, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1230-1. doi: 10.1145/2187980.2188242. URL http://doi.acm.org/10.1145/2187980.2188242.

[4] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semant.*, 6(3):203–217, September 2008. ISSN 1570-8268. doi: 10.1016/j.websem.2008.06.001. URL http://dx.doi.org/10.1016/j.websem.2008.06.001.

[5] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point

for the web of data. *Web Semant.*, 7(3):154–165, September 2009. ISSN 1570-8268. doi: 10.1016/j.websem.2009.07.002. URL http://dx.doi.org/10.1016/j.websem.2009.07.002.

[6] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 101–110, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. doi: 10.1145/1718487.1718501. URL http://doi.acm.org/10.1145/1718487.1718501.

[7] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 227–236, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0493-1. doi: 10.1145/1935826.1935869. URL http://doi.acm.org/10.1145/1935826.1935869.

[8] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601–610. ACM, 2014.

[9] Bonan Min, Ralph Grishman, Li Wan, Chang Wang 0001, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In Lucy Vanderwende, Hal DaumÃČÂĺ III, and Katrin Kirchhoff, editors, *HLT-NAACL*, pages 777–782. The Association for Computational Linguistics, 2013.

[10] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 515–526. ACM, 2014. ISBN 978-1-4503-2744-2.

[11] Denis Krompaß. Exploiting prior knowledge and latent variable representations for the statistical modeling and probabilistic querying of large knowledge graphs. unpublished, November 2015. URL http://nbn-resolving.de/urn:nbn:de:bvb:19-190190.

[12] Gerhard Weikum. What computers should know, shouldn't know, and shouldn't believe. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 1559–1560, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4914-7. doi: 10.1145/3041021.3051120. URL https://doi.org/10.1145/3041021.3051120.

[13] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: Exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1), 2008. ISSN 2150-8097.

[14] Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538, 2011. ISSN 2150-8097.

[15] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3(1-2): 1338–1347, 2010. ISSN 2150-8097.

[16] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. Matching html tables to dbpedia. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, WIMS '15, pages 10:1–10:6, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3293-4. doi: 10.1145/2797115.2797118. URL http://doi.acm.org/10.1145/2797115.2797118.

[17] Dominique Ritze and Christian Bizer. Matching web tables to dbpedia - A feature utility study. In *Proceedings of the 20th International Conference on Extending*

Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017., pages 210–221, 2017.

[18] Matteo Cannaviccio, Denilson Barbosa, and Paolo Merialdo. Accurate fact harvesting from natural language text in wikipedia with lector. In *Proceedings of the 19th International Workshop on Web and Databases*, WebDB '16, pages 9:1–9:6, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4310-7. doi: 10.1145/2932194.2932203. URL http://doi.acm.org/10.1145/2932194.2932203.

[19] Matteo Cannaviccio, Denilson Barbosa, and Paolo Merialdo. Lector: Rdf triples extraction from wikipedia text. DBpedia TextExt Challenge, co-located with SEMANTICS, 2017.

[20] Matteo Cannaviccio, Denilson Barbosa, and Paolo Merialdo. Towards annotating relational data on the web with language models. *In WWW 2018: The 2018 Web Conference, April 23âĂŞ27, 2018, Lyon, France*, 2018.

[21] Fabian Suchanek and Gerhard Weikum. Knowledge harvesting in the big-data era. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 933–938, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2037-5. doi: 10.1145/2463676.2463724. URL http://doi.acm.org/10.1145/2463676.2463724.

[22] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 1419–1428, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4143-1. doi: 10.1145/2872427.2874809. URL https://doi.org/10.1145/2872427.2874809.

[23] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219745. URL http://doi.acm.org/10.1145/219717.219745.

[24] Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

[25] Gjergji Kasneci, Maya Ramanath, Fabian Suchanek, and Gerhard Weikum. The yago-naga approach to knowledge discovery. *SIGMOD Rec.*, 37(4):41–47, March 2009. ISSN 0163-5808. doi: 10.1145/1519103.1519110. URL http://doi.acm.org/10.1145/1519103.1519110.

[26] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1003–1011. Association for Computational Linguistics, 2009.

[27] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016. doi: 10.1109/JPROC.2015.2483592. URL https://doi.org/10.1109/JPROC.2015.2483592.

[28] Ni Lao, Tom Mitchell, and William W. Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 529–539, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145494.

[29] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: A self-organizing framework for information extraction. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 631–640, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526794. URL http://doi.acm.org/10.1145/1526709.1526794.

[30] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[31] Bongwon Suh, Gregorio Convertino, Ed H. Chi, and Peter Pirolli. The singularity is not near: Slowing growth of wikipedia. In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*, WikiSym '09, pages 8:1–8:10, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-730-1. doi: 10.1145/1641309.1641322. URL http://doi.acm.org/10.1145/1641309.1641322.

[32] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017. doi: 10.3233/SW-160218. URL https://doi.org/10.3233/SW-160218.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[34] Sergey Brin. Extracting patterns and relations from the world wide web. In *Selected Papers from the International Workshop on The World Wide Web and Databases*, WebDB '98, pages 172–183, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-65890-4. URL http://dl.acm.org/citation.cfm?id=646543.696220.

[35] 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. ISBN 1-58113-

231-X. doi: 10.1145/336597.336644. URL http://doi.acm.org/10.1145/336597.336644.

[36] Yuan Fang and Kevin Chen-Chuan Chang. Searching patterns for relation extraction over the web: Rediscovering the pattern-relation duality. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 825–834, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0493-1. doi: 10.1145/1935826.1935933. URL http://doi.acm.org/10.1145/1935826.1935933.

[37] Peter D. Turney. Expressing implicit semantic relations without supervision. *CoRR*, abs/cs/0607120, 2006. URL http://arxiv.org/abs/cs/0607120.

[38] Patrick Pantel and Marco Pennacchiotti. Automatically harvesting and ontologizing semantic relations. In *Proceedings of the 2008 Conference on Ontology Learning and Population: Bridging the Gap Between Text and Knowledge*, pages 171–195, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press. ISBN 978-1-58603-818-2. URL http://dl.acm.org/citation.cfm?id=1563823.1563837.

[39] 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. *Artif. Intell.*, 165(1): 91–134, June 2005. ISSN 0004-3702. doi: 10.1016/j.artint.2005.03.001. URL http://dx.doi.org/10.1016/j.artint.2005.03.001.

[40] Daniel Gerber and Axel-Cyrille Ngonga Ngomo. Extracting multilingual natural-language patterns for rdf predicates. In *Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management*, EKAW'12, pages 87–96, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-33875-5. doi: 10.1007/978-3-642-33876-2_10. URL http://dx.doi.org/10.1007/978-3-642-33876-2_10.

[41] Patrick Arnold and Erhard Rahm. Extracting semantic concept relations from wikipedia. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)*, WIMS '14, 2014.

[42] Mark Craven and Johan Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86. AAAI Press, 1999. ISBN 1-57735-083-9. URL http://dl.acm.org/citation.cfm?id=645634.663209.

[43] Sebastian Riedel, Limin Yao, and Andrew McCallum. *Modeling Relations and Their Mentions without Labeled Text*, pages 148–163. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-15939-8. doi: 10.1007/978-3-642-15939-8_10. URL https://doi.org/10.1007/978-3-642-15939-8_10.

[44] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 541–550, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. URL http://dl.acm.org/citation.cfm?id=2002472.2002541.

[45] 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. URL http://dl.acm.org/citation.cfm?id=2390948.2391003.

[46] Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual*

Meeting of the Association for Computational Linguistics: Long Papers - Volume 1, ACL '12, pages 721–729, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390524.2390626.

[47] Ander Intxaurrondo GonzÃČÂąlez de Langarika, Mihai Surdeanu, Oier LÃČÂşpez de Lacalle Lekuona, and Eneko Agirre Bengoa. Removing noisy mentions for distant supervision, 2013-09.

[48] Benjamin Roth, Tassilo Barth, Michael Wiegand, and Dietrich Klakow. A survey of noise reduction methods for distant supervision. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 73–78, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2411-3. doi: 10.1145/2509558.2509571. URL http://doi.acm.org/10.1145/2509558.2509571.

[49] Thomas Palomares, Youssef Ahres, Juhana Kangaspunta, and Christopher Ré. Wikipedia knowledge graph with deepdive. In *Wiki@ICWSM*, 2016.

[50] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

[51] 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. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145596.

[52] Yuval Merhav, Filipe de Sá Mesquita, Denilson Barbosa, Wai Gen Yee, and Ophir Frieder. Extracting information networks from the blogosphere. *TWEB*, 6(3): 11:1–11:33, 2012. doi: 10.1145/2344416.2344418. URL http://doi.acm.org/10.1145/2344416.2344418.

[53] 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, pages 1135–1145, 2012.

[54] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, 2012.

[55] Filipe de Sá Mesquita, Jordan Schmidek, and Denilson Barbosa. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 447–457, 2013. URL http://aclweb.org/anthology/D/D13/D13-1043.pdf.

[56] Matthew Hurst. Layout and language: Challenges for table understanding on the web. In *Proceedings of the International Workshop on Web Document Analysis*, pages 27–30, 2001.

[57] Michael J. Cafarella and Eugene Wu. Uncovering the relational web. In *In under review*, 2008.

[58] Dominique Ritze, Oliver Lehmberg, Yaser Oulabi, and Christian Bizer. Profiling the potential of web tables for augmenting cross-domain knowledge bases. In *Proceedings of the 25th International Conference on World Wide Web*, pages 251–261, 2016.

[59] Yalin Wang and Jianying Hu. A machine learning based approach for table detection on the web. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 242–250, New York, NY, USA, 2002. ACM. ISBN 1-58113-449-5. doi: 10.1145/511446.511478. URL http://doi.acm.org/10.1145/511446.511478.

[60] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. A large public corpus of web tables containing time and context metadata. In Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao, editors, *Proceedings of the 25th International Conference on World Wide Web, (WWW 2016), Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 75–76. ACM, 2016. doi: 10.1145/2872518.2889386. URL http://doi.acm.org/10.1145/2872518.2889386.

[61] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, IDEA '13, pages 18–26, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2329-1. doi: 10.1145/2501511.2501516. URL http://doi.acm.org/10.1145/2501511.2501516.

[62] Emir Munoz, Aidan Hogan, and Alessandra Mileo. Triplifying wikipedia's tables. In *LD4IE'13 Workshop Proceedings, ISWC. CEUR*, 2013.

[63] Hannes Dohrn and Dirk Riehle. Design and implementation of the sweble wikitext parser: Unlocking the structured data of wikipedia. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, WikiSym '11, pages 72–81, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0909-7. doi: 10.1145/2038558.2038571. URL http://doi.acm.org/10.1145/2038558.2038571.

[64] Eric Crestan and Patrick Pantel. Web-scale table census and classification. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 545–554, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0493-1. doi: 10.1145/1935826.1935904. URL http://doi.acm.org/10.1145/1935826.1935904.

[65] Ziqi Zhang. Towards efficient and effective semantic table interpretation. In *Proceedings of the 13th International Semantic Web Conference - Part I*, ISWC '14, pages 487–502, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

ISBN 978-3-319-11963-2. doi: 10.1007/978-3-319-11964-9_31. URL http://dx.doi.org/10.1007/978-3-319-11964-9_31.

[66] Disheng Qiu, Luciano Barbosa, Xin Luna Dong, Yanyan Shen, and Divesh Srivastava. Dexter: Large-scale discovery and extraction of product specifications on the web. *Proc. VLDB Endow.*, 8(13):2194–2205, September 2015. ISSN 2150-8097. doi: 10.14778/2831360.2831372. URL https://doi.org/10.14778/2831360.2831372.

[67] Hazem Elmeleegy, Jayant Madhavan, and Alon Halevy. Harvesting relational tables from lists on the web. *Proc. VLDB Endow.*, 2(1):1078–1089, August 2009. ISSN 2150-8097. doi: 10.14778/1687627.1687749. URL http://dx.doi.org/10.14778/1687627.1687749.

[68] Marco D. Adelfio and Hanan Samet. Schema extraction for tabular data on the web. *Proc. VLDB Endow.*, 6(6):421–432, April 2013. ISSN 2150-8097. doi: 10.14778/2536336.2536343. URL http://dx.doi.org/10.14778/2536336.2536343.

[69] Zhe Chen and Michael Cafarella. Automatic web spreadsheet data extraction. In *Proceedings of the 3rd International Workshop on Semantic Search Over the Web*, SS@ '13, pages 1:1–1:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2483-0. doi: 10.1145/2509908.2509909. URL http://doi.acm.org/10.1145/2509908.2509909.

[70] Zhe Chen and Michael Cafarella. Integrating spreadsheet data via accurate and low-effort extraction. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1126–1135, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623617. URL http://doi.acm.org/10.1145/2623330.2623617.

[71] Mirko Bronzi, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Extraction and integration of partially overlapping web sources. *Proc. VLDB Endow.*, 6(10):805–816, August 2013. ISSN 2150-8097. doi: 10.14778/2536206.2536209. URL http://dx.doi.org/10.14778/2536206.2536209.

[72] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 109–118, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-804-4. URL http://dl.acm.org/citation.cfm?id=645927.672370.

[73] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. doi: 10.3115/992133.992154. URL https://doi.org/10.3115/992133.992154.

[74] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *Transactions on Knowledge & Data Engineering*, 27(2):443–460, 2015. doi: 10.1109/TKDE.2014.2327028. URL http://www.computer.org/csdl/trans/tk/2015/02/06823700-abs.html.

[75] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. AIDA: an online tool for accurate disambiguation of named entities in text and tables. *PVLDB*, 4(12):1450–1453, 2011.

[76] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, pages 1–8, 2011. doi: 10.1145/2063518.2063519. URL http://doi.acm.org/10.1145/2063518.2063519.

[77] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, 2013.

[78] Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. J-REED: joint relation extraction and entity disambiguation. In *Proceedings of the 2017 ACM on*

Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017, pages 2227–2230, 2017. doi: 10.1145/3132847.3133090. URL http://doi.acm.org/10.1145/3132847.3133090.

[79] Luciano Del Corro and Rainer Gemulla. Clausie: clause-based open information extraction. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 355–366, 2013. URL http://dl.acm.org/citation.cfm?id=2488420.

[80] Alexander A. Morgan, Lynette Hirschman, Marc Colosimo, Alexander S. Yeh, and Jeff B. Colombe. Gene name identification and normalization using a model organism database. *Journal of Biomedical Informatics*, 37(6):396 – 410, 2004. ISSN 1532-0464. doi: https://doi.org/10.1016/j.jbi.2004.08.010. URL http://www.sciencedirect.com/science/article/pii/S1532046404000875. Named Entity Recognition in Biomedicine.

[81] Nicolas Heist and Heiko Paulheim. Language-agnostic relation extraction from wikipedia abstracts. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, pages 383–399, 2017. doi: 10.1007/978-3-319-68288-4_23. URL https://doi.org/10.1007/978-3-319-68288-4_23.

[82] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: exploring the power of tables on the web. *PVLDB*, 1(1): 538–549, 2008.

[83] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *PVLDB*, 3(1):1338–1347, 2010.

[84] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. Using linked data to mine rdf from wikipedia's tables. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 533–542. ACM, 2014.

[85] Yoones A. Sekhavat, Francesco Di Paolo, Denilson Barbosa, and Paolo Merialdo. Knowledge base augmentation using tabular data. In *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014.*, 2014. URL http://ceur-ws.org/Vol-1184/ldow2014_paper_02.pdf.

[86] Yue Wang and Yeye He. Synthesizing mapping relationships using table corpus. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1117–1132. ACM, 2017.

[87] Yusra Ibrahim, Mirek Riedewald, and Gerhard Weikum. Making sense of entities and quantities in web tables. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1703–1712, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4073-1. doi: 10.1145/2983323.2983772. URL http://doi.acm.org/10.1145/2983323.2983772.

[88] Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 777–782, 2013.

[89] Chi Wang, Kaushik Chakrabarti, Tao Cheng, and Surajit Chaudhuri. Targeted disambiguation of ad-hoc, homogeneous sets of named entities. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 719–728, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5. doi: 10.1145/2187836.2187934. URL http://doi.acm.org/10.1145/2187836.2187934.

[90] Jiannan Wang, Guoliang Li, and Jianhua Feng. Extending string similarity join to tolerant fuzzy token matching. *ACM Trans. Database Syst.*, 39(1):7:1–7:45, January 2014. ISSN 0362-5915. doi: 10.1145/2535628. URL http://doi.acm.org/10.1145/2535628.

[91] ChengXiang Zhai. *Statistical Language Models for Information Retrieval*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2008. doi: 10.2200/S00158ED1V01Y200811HLT001. URL http://dx.doi.org/10.2200/S00158ED1V01Y200811HLT001.

[92] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, 1998. doi: 10.1145/290941.291008. URL http://doi.acm.org/10.1145/290941.291008.

[93] John D. Lafferty and ChengXiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 111–119, 2001. doi: 10.1145/383952.383970. URL http://doi.acm.org/10.1145/383952.383970.

[94] Krisztian Balog, Marc Bron, and Maarten De Rijke. Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.*, 29(4): 22:1–22:31, 2011.

[95] Marc Bron, Krisztian Balog, and Maarten de Rijke. Example based entity search in the web of data. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, ECIR'13, pages 392–403. Springer-Verlag, 2013. ISBN 978-3-642-36972-8.

[96] Mandar Joshi, Uma Sawant, and Soumen Chakrabarti. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1104–1114, 2014.

[97] Jeffrey Pound, Ihab F. Ilyas, and Grant E. Weddell. QUICK: expressive and flexible search over knowledge bases and text collections. *PVLDB*, 3(2):1573–1576, 2010.

[98] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. Language-model-based ranking for queries on rdf-graphs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 977–986, 2009. doi: 10.1145/1645953.1646078. URL http://doi.acm.org/10.1145/1645953.1646078.

[99] Mohamed Yahya, Denilson Barbosa, Klaus Berberich, Qiuyue Wang, and Gerhard Weikum. Relationship queries on extended knowledge graphs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 605–614. ACM, 2016. ISBN 978-1-4503-3716-8.

[100] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 379–390, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390948.2390995.

[101] Danushka Tarupathi Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Relational duality: Unsupervised extraction of semantic relations between entities on the web. In *Proceedings of the 19th International Conference on World Wide Web*, WWW 2010, pages 151–160. ACM, 2010. ISBN 978-1-60558-799-8.

[102] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, 2013.

[103] Doug Downey, Stefan Schoenmackers, and Oren Etzioni. Sparse information extraction: Unsupervised language models to the rescue. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007.

[104] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL'05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219885. URL http://dx.doi.org/10.3115/1219840.1219885.

[105] Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989.

[106] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. ISBN 978-0-521-86571-5.

[107] Dong Deng, Yu Jiang, Guoliang Li, Jian Li, and Cong Yu. Scalable column concept determination for web tables using large knowledge bases. *PVLDB*, 6 (13):1606–1617, 2013. URL http://www.vldb.org/pvldb/vol6/p1606-li.pdf.

[108] Kentaro Inui, Sebastian Riedel, Pontus Stenetorp, and Sonse Shimaoka. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1271–1280, 2017. URL https://aclanthology.info/papers/E17-1119/e17-1119.

[109] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, and Vassilis Christophides. Matching web tables with knowledge base entities: From entity

lookups to entity embeddings. In *International Semantic Web Conference*, pages 260–277. Springer, 2017.

[110] William A. Gale, Kenneth W. Church, and David Yarowsky. One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 233–237, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics. ISBN 1-55860-272-0. doi: 10.3115/1075527.1075579. URL https://doi.org/10.3115/1075527.1075579.

[111] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 327–336. ACM, 1998. ISBN 0-89791-962-9.

[112] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2302–2310, 2015.

[113] Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 365–374, 2014.

[114] Xiaonan Li, Chengkai Li, and Cong Yu. Entity-relationship queries over wikipedia. *ACM Trans. Inf. Syst.*, 3(4):70, 2012.

[115] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706, 2007.

[116] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 28–36, 2003.

[117] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[118] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 2006.

[119] Li Ding, Dominic DiFranzo, Alvaro Graves, James Michaelis, Xian Li, Deborah L. McGuinness, and Jim Hendler. Data-gov wiki: Towards linking government data. In *Linked Data Meets Artificial Intelligence, Papers from the 2010 AAAI Spring Symposium, Technical Report SS-10-07, Stanford, California, USA, March 22-24, 2010*, 2010.

[120] Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff, editors. *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings*, 2013. The Association for Computational Linguistics. ISBN 978-1-937284-47-3.

[121] Yalin Wang and Jianying Hu. Detecting tables in html documents. In *Proceedings of the 5th International Workshop on Document Analysis Systems V*, DAS '02, pages 249–260, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44068-2. URL http://dl.acm.org/citation.cfm?id=647798.736657.

[122] Shuo Zhang and Krisztian Balog. Entitables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Confer-*

ence on Research and Development in Information Retrieval, SIGIR '17, pages 255–264, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5022-8. doi: 10.1145/3077136.3080796. URL http://doi.acm.org/10.1145/3077136.3080796.

[123] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. Finding related tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 817–828, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1247-9. doi: 10.1145/2213836.2213962. URL http://doi.acm.org/10.1145/2213836.2213962.

[124] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 0-934613-73-7.

[125] José Manuél Gómez-Pérez, Jeff Z. Pan, Guido Vetere, and Honghan Wu. Enterprise knowledge graph: An introduction. In *Exploiting Linked Data and Knowledge Graphs in Large Organisations*, pages 1–14, 2017. doi: 10.1007/978-3-319-45654-6_1.

[126] Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[127] Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Automatic expansion of dbpedia exploiting wikipedia cross-language information. In Philipp Cimiano, ÃŕÂ£ÅŞscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph, editors, *ESWC*, volume 7882 of *Lecture Notes in Computer Science*, pages 397–411. Springer, 2013. ISBN 978-3-642-38288-8.

[128] Yangqiu Song and Dan Roth. Machine learning with world knowledge: The position and survey. *CoRR*, abs/1705.02908, 2017. URL http://arxiv.org/abs/1705.02908.

[129] Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. Partially supervised classification of text documents. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 387–394, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7. URL http://dl.acm.org/citation.cfm?id=645531.656022.

[130] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3):103–134, May 2000. ISSN 0885-6125. doi: 10.1023/A:1007692713085. URL https://doi.org/10.1023/A:1007692713085.

[131] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 587–592, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1630659.1630746.

[132] FranÃŕÂ£ÅŞois Denis, RÃŕÂ£ÅŞmi Gilleron, and Marc Tommasi. Text classification from positive and unlabeled examples, 2002.

[133] 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, 2007.

[134] 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, 2010.

[135] Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 427–434, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219893. URL https://doi.org/10.3115/1219840.1219893.

[136] Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 523–534. ACL, 2012. URL http://dl.acm.org/citation.cfm?id=2390948.2391009.

[137] Chris Quirk and Hoifung Poon. Distant supervision for relation extraction beyond the sentence boundary. *CoRR*, abs/1609.04873, 2016. URL http://arxiv.org/abs/1609.04873.

[138] Gang Wang, Yong Yu, and Haiping Zhu. *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007. Proceedings*, chapter PORE: Positive-Only Relation Extraction from Wikipedia Text, pages 580–594. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-76298-0. doi: 10.1007/978-3-540-76298-0_42. URL http://dx.doi.org/10.1007/978-3-540-76298-0_42.

[139] Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. Extending the coverage of dbpedia properties using distant supervision over wikipedia. In *Proceedings of the 2013th International Conference on NLP & DBpedia - Volume 1064*, NLP-DBPEDIA'13, pages 20–31, Aachen, Germany, Germany, 2013. CEUR-WS.org. URL http://dl.acm.org/citation.cfm?id=2874479.2874482.

[140] Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. Multilingual relation extraction using compositional universal schema. *CoRR*, abs/1511.06396, 2015. URL http://arxiv.org/abs/1511.06396.

[141] Denis Savenkov and Eugene Agichtein. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 235–244, New York, NY, USA, 2016.

ACM. ISBN 978-1-4503-4069-4. doi: 10.1145/2911451.2911536. URL http://doi.acm.org/10.1145/2911451.2911536.

[142] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2011. ISBN 1107015359, 9781107015357.

[143] Eduard Hovy, Roberto Navigli, and Simone Paolo Ponzetto. Collaboratively built semi-structured content and artificial intelligence: The story so far. *Artif. Intell.*, 194:2–27, January 2013. ISSN 0004-3702.

[144] Partha Pratim Talukdar, Derry Tanti Wijaya, and Tom M. Mitchell. Acquiring temporal constraints between relations. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, CIKM '12, pages 992–1001. ACM, 2012. ISBN 978-1-4503-1156-4.

[145] Zhe Chen and Michael J. Cafarella. Automatic web spreadsheet data extraction. In *SSW '13*, SS@ '13. ACM, 2013. doi: 10.1145/2509908.2509909. URL http://doi.acm.org/10.1145/2509908.2509909.

[146] Varish Mulwad, Tim Finin, and Anupam Joshi. Semantic message passing for generating linked data from tables. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, pages 363–378, 2013.

[147] Marco D. Adelfio and Hanan Samet. Schema extraction for tabular data on the web. *PVLDB*, 6(6):421–432, 2013. ISSN 2150-8097.

[148] Guodong Zhou, Min Zhang, Dong-Hong Ji, and Qiaoming Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 728–736, 2007.