Knowledge Graph Exploration for Natural Language Understanding in Web Information Retrieval

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

vorgelegt von
Michael Schuhmacher

Mannheim, 2016

Dekan: Professor Dr. Heinz Jürgen Müller, Universität Mannheim
Referent: Professor Dr. Simone Paolo Ponzetto, Universität Mannheim
Korreferent: Professor Dr. Heiner Stuckenschmidt, Universität Mannheim
Tag der mündlichen Prüfung: 11. November 2016

Abstract

In this thesis, we study methods to leverage information from fully-structured knowledge bases (KBs), in particular the encyclopedic knowledge graph (KG) DBpedia, for different text-related tasks from the area of information retrieval (IR) and natural language processing (NLP). The key idea is to apply entity linking (EL) methods that identify mentions of KB entities in text, and then exploit the structured information within KGs. Developing entity-centric methods for text understanding using KG exploration is the focus of this work.

We aim to show that structured background knowledge is a means for improving performance in different IR and NLP tasks that traditionally only make use of the unstructured text input itself. Thereby, the KB entities mentioned in text act as connection between the unstructured text and the structured KG. We focus in particular on how to best leverage the knowledge as contained in such fully-structured (RDF) KGs like DBpedia with their labeled edges/predicates – which is in contrast to previous work on Wikipedia-based approaches we build upon, which typically relies on unlabeled graphs only. The contribution of this thesis can be structured along its three parts:

In Part I, we apply EL and semantify short text snippets with KB entities. While only retrieving types and categories from DBpedia for each entity, we are able to leverage this information to create semantically coherent clusters of text snippets. This pipeline of connecting text to background knowledge via the mentioned entities will be reused in all following chapters.

In Part II, we focus on semantic similarity and extend the idea of semantifying text with entities by proposing in Chapter 5 a model that represents whole documents by their entities. In this model, comparing documents semantically with each other is viewed as the task of comparing the semantic relatedness of the respective entities, which we address in Chapter 4. We propose an unsupervised graph weighting schema and show that weighting the DBpedia KG leads to better results on an existing entity ranking dataset. The exploration of weighted KG paths turns out to be also useful when trying to disambiguate the entities from an open information extraction (OIE) system in Chapter 6. With this weighting schema, the integration of KG information for computing semantic document similarity in Chapter 5 becomes the task of comparing the two KG subgraphs with each other, which we address by an approximate subgraph matching. Based on a well-established evaluation dataset for semantic document similarity, we show that our unsupervised method achieves competitive performance similar to other state-of-the-art methods. Our results from this part indicate that KGs can contain helpful background knowledge, in particular when exploring KG paths, but that selecting the relevant parts of the graph is an important yet difficult challenge.

In Part III, we shift to the task of relevance ranking and first study in Chapter 7 how to best retrieve KB entities for a given keyword query. Combining again text with KB information, we extract entities from the top-k retrieved, query-specific documents and then link the documents to two different KBs, namely Wikipedia and DBpedia. In a learning-to-rank setting, we study extensively which features from the text, the Wikipedia KB, and the DBpedia KG can be helpful for ranking entities with respect to the query. Experimental results on two datasets, which build

upon existing TREC document retrieval collections, indicate that the document-based mention frequency of an entity and the Wikipedia-based query-to-entity similarity are both important features for ranking. The KG paths in contrast play only a minor role in this setting, even when integrated with a semantic kernel extension. In Chapter 8, we further extend the integration of query-specific text documents and KG information, by extracting not only entities, but also relations from text. In this exploratory study based on a self-created relevance dataset, we find that not all extracted relations are relevant with respect to the query, but that they often contain information not contained within the DBpedia KG. The main insight from the research presented in this part is that in a query-specific setting, established IR methods for document retrieval provide an important source of information even for entity-centric tasks, and that a close integration of relevant text document and background knowledge is promising.

Finally, in the concluding chapter we argue that future research should further address the integration of KG information with entities and relations extracted from (specific) text documents, as their potential seems to be not fully explored yet. The same holds also true for a better KG exploration, which has gained some scientific interest in recent years. It seems to us that both aspects will remain interesting problems in the next years, also because of the growing importance of KGs for web search and knowledge modeling in industry and academia.

Zusammenfassung

In dieser Arbeit wird die Nutzung von strukturierten Wissensbasen, insbesondere des enzyklopädische DBpedia Knowledge Graphs, für verschiedene Problemstellungen aus dem Bereich des Information Retrieval (IR) und des Natural Language Processing (NLP) untersucht. Im Zentrum steht dabei die Idee, Textdokumente mithilfe existierender Entity Linking Methoden zu den Entitäten dieser Wissensgraphen zu verlinken, und somit die Erschliessung des externen Hintergrundwissens aus diesen Wissensbasen zu ermöglichen. Der wesentliche Beitrag dieser Arbeit liegt in der Entwicklung von Methoden zur Erschließung von Wissensgraphen für im Kontext verschiedener Aufgabenstellungen des IR und NLP.

Im ersten Teil der Arbeit (Teil I) wird ein Verfahren zum Search Results Clustering entwickelt, in welchem die Dokumentenergebnisse von mehrdeutigen Suchanfragen in semantisch kohärente Gruppen von Dokumenten zusammengefasst werde. Hierfür werden Textfragmente der Suchergebnisse semantifiziert, indem im Text genannte Entitäten zu deren entsprechenden Repräsentationen in der DBpedia Wissensbasen verknüpft werden. Von dieser Wissensbases kann im Anschluss Hintergrundwissen bezogen werden, um das Clustering zu verbessern.

Im darauffolgenden Teil II wird dieser Ansatz der Verknüpfung von Text und Wissensbasis weiter vertieft, indem die Bestimmung der semantische Ähnlichkeit von Textdokumenten (Semantic Document Similarity) mithilfe einer Projektion derselben in den Wissensgraphen ermöglicht wird. Das Problem der semantisch Ähnlichkeit wird somit auf den paarweisen Vergleich zweier Teilgraphen reduziert, welcher dann durch eine Approximation der Graph Edit Distance (GED) zwischen diesen gelöst wird. Der hierfür notwendige semantische Vergleich von einzelnen Entitäten (Semantic Relatedness) wird dabei durch die Berechnung von kürzesten Pfaden innerhalb des Wissensgraphen ermöglicht. Aufgrund der hohen Anzahl an Kanten und Pfade innerhalb des Graphen wird ein informationstheoretisches Gewichtungsschema vorgeschlagen, welches Pfade nach deren Informationsgehalt gewichtet. Das Gesamtverfahren wird mit etablierten Referenzdatensätzen evaluiert und gegen alternative Methoden verglichen.

Im Teil III findet eine Erweiterung des Problemhorizontes statt, indem das Problem der Anfragerelevanz (query relevance) betrachtet wird, d.h. die Frage, wie Suchanfragen mithilfe von Entitäten anstelle von Textdokumenten beantwortet werden können. Hierzu wird die in Teil I vorgestellte Methode der Semantifizierung von Suchergebnissen genutzt, um relevante Entitäten aus den Dokumenten zu extrahieren. Eine sinnvolle Sortierung der Entitäten wird im Anschluss durch eine Kombination verschiedener Signale, basierend auf den Informationen der relevante Dokumente, aber auch aufgrund des Hintergrundwissens von semi- und vollstrukturierten Wissensbasen, erreicht. Im finalen Kapitel wird in einem explorativen Ansatz die Relevanz von Relationen aus Textdokumenten untersucht und die Frage der Kombination von text- und wissensbasenbasierten Informationsdarstellungen aufgeworfen.

Im abschließenden Ausblick auf zukünftige Forschungsfragen wird insbesondere auf das Verhältnis von textbasierten und wissensbasisbasierten Informationen, und deren Rolle für die Weiterentwicklung in Richtung einer echten semantischen Informationssuche diskutiert.

Contents

List of Publications

The work presented in this thesis has been published before in the proceedings of different conferences, this may also include figures, tables, and algorithms. For all publications the author of this thesis was the key contributor of the work presented in both the publications and this thesis.

- Michael Schuhmacher and Simone Paolo Ponzetto: Exploiting DBpedia for web search results clustering. *In Proceedings of AKBC'13, pages 91–96* (Schuhmacher and Ponzetto, 2013): Chapter 3
- Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-based Graph Document Modeling. *In Proceedings of WSDM'14, pages 543–552* (Schuhmacher and Ponzetto, 2014a): Chapter 4 and 5
- Michael Schuhmacher and Simone Paolo Ponzetto: Ranking Entities in a Large Semantic Network. *In Proceedings of ESWC'14 Satellite Events, pages 254–258* (Schuhmacher and Ponzetto, 2014b): Chapter 4
- Arnab Dutta and Michael Schuhmacher: Entity Linking for Open Information Extraction. *In Proceedings of NLDB'14, pages 75–80* (Dutta and Schuhmacher, 2014): Chapter 6
- Michael Schuhmacher, Laura Dietz, and Simone Paolo Ponzetto: Ranking entities for web queries through text and knowledge. *In Proceedings of CIKM'15, pages 1461–1470* (Schuhmacher et al., 2015): Chapter 7
- Michael Schuhmacher, Benjamin Roth, Simone Paolo Ponzetto, and Laura Dietz. Finding Relevant Relations in Relevant Documents. *In Proceedings of ECIR'16, pages 654–660* (Schuhmacher et al., 2016): Chapter 8

List of Figures

List of Tables

List of Acronyms

AI artificial intelligence

BoW bag-of-words

ESA explicit semantic analysis

EL entity linking

GED graph edit distance

idf inverse document frequency

IC information content

IR information retrieval

IRI internationalized resource identifier

INEX Initiative for the Evaluation of XML Retrieval

KB knowledge base

KBP knowledge base population

KG knowledge graph

LSA latent semantic analysis

LOD linked open data

LP50 50 documents dataset from Lee et al. (2005)

LTR learning-to-rank

MRF Markov random field

NLP natural language processing

OIE open information extraction

POS part-of-speech

PMI pointwise mutual information

RDF resource description framework

STS semantic textual similarity

SVM support vector machine

tf-idf term frequency–inverse document frequency

VSM vector space model

Chapter 1

Introduction

Assisting humans when working with natural language text (documents) is the focus of two well-established areas of computer science, namely information retrieval (IR) and natural language processing (NLP). While IR comes traditionally from the task of finding relevant text document for a user, NLP is focused on the more fine-grained text processing, often at the sentence level. Tasks are here for example to identify grammatical structures or to compare the meaning, i.e. the semantic, of words or sentences. Both research areas have in common that they build upon the data at hand, i.e. e.g. the document to be retrieved, but often also include additional information, called background knowledge, that is not explicitly contained within the data at hand. This approach of combining different knowledge sources is used for example to overcome challenges like the vocabulary (mismatch) problem: A user is looking for a document mentioning "a coastal fish", but when applying a purely syntactic word matching, it is hard for a computational method to retrieve a document which mentions the "lumpfish" – even though this is a well-known coastal fish and the document should be retrieved. Such encyclopedic knowledge, that a lumbfish is a coastal fish, is usually formalized within knowledge bases (KBs).

In this thesis, we will explore in which ways IR and NLP tasks can benefit from the usage of general, encyclopedic KBs like DBpedia. In the following, we will first outlay our motivation and the overall context of this thesis, before presenting the thesis structure in Section 1.2.

1.1 Motivation

In recent years, significant efforts have been made to build such wide-coverage KBs like DBpedia, Yago, or Freebase (Bizer et al., 2009; Suchanek et al., 2008; Bollacker et al., 2008), which are often (partially) derived from Wikipedia and which have become popularly known as KGs.[1] Those KGs contain factual knowledge about real world entities and their relations and attributes in a fully machine-readable format. They contain, e.g. the fact that the aforementioned lumpfish is a coastal fish, and that it is a fish. Complementary to the trend of creating machine-readable

[1] A term that goes actually back to Google's commercial "Knowledge Graph" and which is used to emphasize the fact that the contained entities are connected by binary predicates, thus creating a graph structure. We will go into more details below in Section 2.1.

KBs from (the manually created) Wikipedia, much research efforts have also concentrated on the automatic acquisition of machine-readable knowledge from textual data such as the Web (Banko et al., 2007; Carlson et al., 2010): Those KBs can contain information from arbitrary websites (or other sources) that we would usually not find in Wikipedia – simply because even the large English Wikipedia cannot cover all of the world's knowledge.

As a result of the availability of these knowledge resources, recent years have seen a renaissance of knowledge-rich approaches for many task in IR, in particular in Web Search, and NLP (Hovy et al., 2013). One prominent IR example are the entity boxes shown by commercial search engines: Those boxes, usually shown on the right-hand side of the screen, provide a structured view on (e.g. person) entities. The information shown there, like a person's birth date or links to the person's children, are extracted from Wikipedia, which acts as an entity repository as well as the source of such information here. Another example from NLP is the task of finding the different senses of a word, e.g. that bank can refer to a financial institute but also to a river bank. For this word sense disambiguation (WSD) task Wikipedia provides the sense repository as well as the textual features to distinguish between the different senses in a given context (cf. Navigli, 2009).

This trend indicates to us that semantic information and knowledge-intensive approaches are key components for state-of-the-art methods that build heavily on KBs. However, for many high-end applications, Wikipedia – even though being only a semi-structured resources with many information contained only in the natural language text – remains the knowledge resources of choice for many tasks (cf. e.g. Ponzetto and Strube, 2007; Nastase and Strube, 2012), while the usage of fully structured KGs lacks behind. This seems to be a curiosity, because full-fledged KGs with their labeled relations do actually contain more precises information compared to the simple HTML hyperlinks in Wikipedia: For example, from Wikipedia we can know that a person's Wikipedia page links to another person's Wikipedia page, so there is some kind of relationship connecting both entities – but form DBpedia we would known that those two persons are actually connected by the parentOf relation. To us, it appears to be valid question why such structured KGs are not widely used (yet).

We aim to address this particular question, and study how wide-coverage KGs – we use DBpedia in our experiments – can be utilized for IR and NLP tasks, because those structured KGs offer interesting properties: They contain (i) disambiguated representations of real word entities, (ii) predicates describing the relationships between entities, and (iii) concepts acting as topological information like type (hierarchies) or categories. Given we are interested in IR and NLP tasks and applications where those KGs can be useful, the usual setting as depicted in Figure 1.1 in this thesis – which is one of the recurring elements in most chapters – is to (i) take natural language text, (ii) apply entity linking (and optionally relation extraction as in Chapter 8) to get (iii) KB entities (and optionally relations), and finally (iv) leverage (certain parts of) the KG for the task at hand, e.g. text clustering (Chapter 3). In the end, our motivation is always to figure out what one can do with the KG information obtained from such a pipeline, i.e. how and for what are the KG information useful. Note that the inclusion of (text-based) relations is an extension only used in the last part of this thesis (Chapter 8), where not only entity linking, but also relation

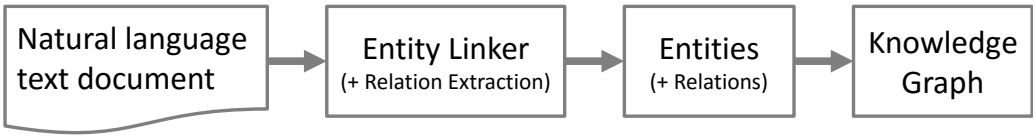

Figure 1.1: Typical workflow: (i) take natural language text, (ii) apply entity linking (and optionally relation extraction) to get (iii) KB entities (and optionally relations), and finally (iv) leverage (certain parts of) the KG.

extraction of subject-predicate-object facts will be applied.

1.2 Structure

The structure of the thesis follows our successive efforts to exploit more and more information from the KG, starting from simple entity types, going to entity relations and graph exploration, and ending with textual relations. While the first half of the thesis focuses thereby on natural language understanding tasks, the second part is designed around IR problems in the context of KB entities. The chapters are designed as follows:

- Starting in Chapter 3, we study how clustering of short text can benefit from entity linking the text and incorporating the DBpedia type and category information (but without any graph exploration).
- The natural extension follows in Chapter 4, where we explore arbitrary relations (KG paths) between entities, and not just types and categories, to estimate their semantic relatedness – which is computed as the cheapest path in a weighted version of the KG.
- In Chapter 5, we extend this method to compute semantic relatedness of documents by representing them as subgraphs of the weighted KG and matching them via graph edit distance.
- The method for entity relatedness is reused in Chapter 6 with a specific linking task for open information extraction, where NELL triples get partially linked to KB entities.
- After having focused on natural language understanding tasks, in Chapter 7 we turn to a more IR-oriented setting and work on query-driven KB entity ranking. Again, KG entities and their relations are utilized, amongst many other features, to compare entities and queries and in the end to find query-relevant entities.
- The previous approach is extended in the last chapter (Chap. 8) where we aim at finding query-relevant fact, i.e. entities and relations. This opens a new research direction as, in contrast the previous chapters, not only entities but also relations are extracted from the input text documents.

A graphical representation of the dependencies between the different chapters is given by the topic map in Figure 1.2. It shows in particular how the idea of the KG path exploration from Chapter 4 and the approach to semantify text with KB entities from Chapter 3 is reused in other chapters throughout this thesis. Chapter 8 is only indirectly (dotted line) influence by Chapter 4, as the KG relations are only compared against (but not integrated with) the relations extracted from text.

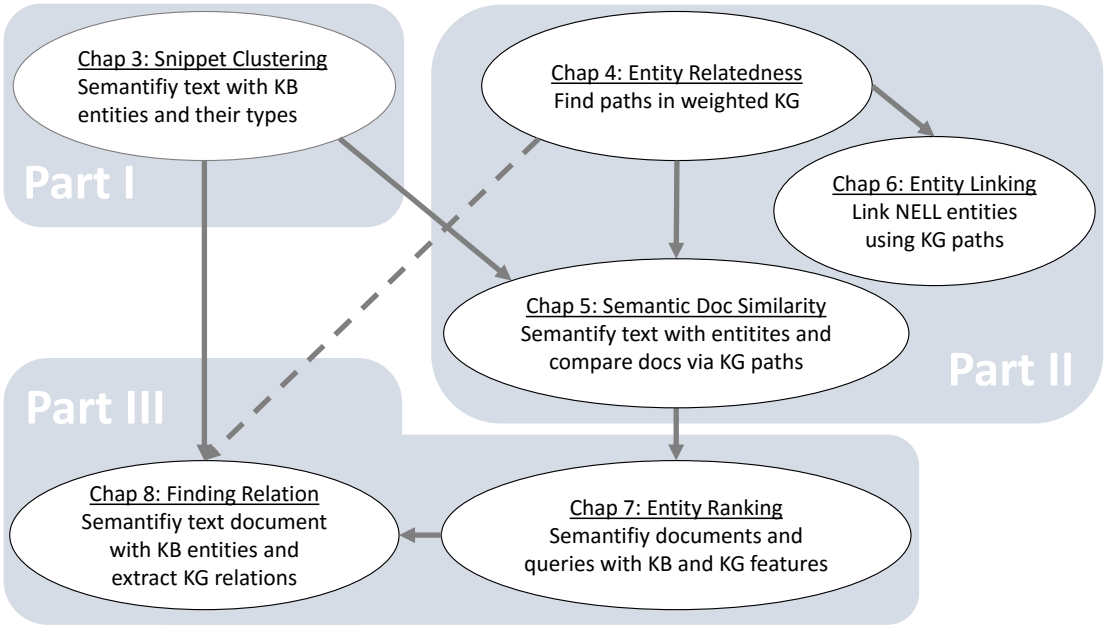

Figure 1.2: Topic map showing dependencies between chapters: Arrows indicate that methods/ideas from this chapter are being reused by the other chapter. The dotted line indicates that Chapter 8 only compares its relations from text against the KG relations from Chapter 4.

Before staring with the actual content chapters, we first briefly introduce in the following Chapter 2 some essential fundamentals, which the average computer scientist might not know, but which are needed for understanding the remainder of this thesis.

1.3 Contributions

The contribution of this thesis is a broad and diverse, experimental study on the potential of structured KGs, like DBpedia, for different NLP and IR tasks. The study provides the reader with an understanding of for what applications KGs might be useful, and what their strength and weaknesses are compared to statistical, non-symbolic approaches.

Most work in the area of KB exploration makes only use of semi-structured knowledge resources, very often Wikipedia (Hovy et al., 2013). In contrast, our contribution is to shed light on

the interplay between text (documents), semi-structured KBs (Wikipedia), and fully-structured KGs (DBpedia). In the end and on a higher level, we thus also give justification for building such wide-coverage KGs and advocate to thinking about other applications that might benefit from using these knowledge resources.

More specifically, this thesis promotes the idea to combine text and structured KGs via the usage of entity linking (cf. Figure 1.1), thus making use of the symbolic, knowledge-rich approaches in combination with statistical approaches.

Our experiments indicate that KBs can indeed help to bridge the vocabulary gap (Furnas et al., 1987) and introduce access to helpful background knowledge, but it depends on the specific task to decided what is needed: When computing semantic document similarity in Chapter 5, we find that entities are not enough to represent a document, but KG paths improve performance (cf. Section 5.3). In contrast, when ranking entities by relevance w.r.t. a given query in Chapter 7, the KG paths by themselves are not that helpful, but the occurrence frequency of Wikipedia KB entities in the retrieved documents is (cf. Section 7.3). In summary, we provide the reader of this thesis with a differentiated view on the usage of (semi-)structured KBs in IR and NLP.

In terms of methods developed, the key contributions of this thesis are as follows:

- A KG weighting and exploration method for computing semantic relatedness of entities (Chap. 4: Schuhmacher and Ponzetto, 2014a,b)
- A KG-based document model that computes semantic document similarity by subgraph matching (Chap. 5: Schuhmacher and Ponzetto, 2014a)
- A query-specific entity ranking combining information from retrieved documents with background KB and KG (Chap. 7: Schuhmacher et al., 2015)

Chapter 2

Fundamentals

This chapter will introduce terms, methods, and datasets recurrently used throughout the remainder of this thesis and with which the average reader (computer scientist) will not be familiar with. Topics relevant only for one chapter are discussed within the chapter, e.g. Learning-to-rank in Section 7.2.3.

2.1 Entities and Knowledge Bases

Throughout the previous introduction chapter, we have used the terms knowledge base (KB), knowledge graph (KG), and entity without paying much attention to their definitions. When now trying to provide proper definitions, we are faced with a variety of competing and/or overlapping definitions and understandings. We are going to present here only a narrow selection of ideas and opt to focus in the end on the practical, i.e. technical reality as established by the given data available, i.e. the RDF KBs.

2.1.1 Entities

From the point of view of the DBpedia KB, entities are the atomic units, like e.g. the city `db:Mannheim`, that the KB makes statements about: `db:MVV_Energie dbo:locationCity db:Mannheim`. We will come back to this technical interpretation, but look first at the general case, as the question what an "entity" is has been discussed for a long time in the area of knowledge representation and artificial intelligence (AI).

In AI, the need for a proper knowledge representation, and thus an understanding of what an "entity" is, was mainly driven by the insight that for matching human performance in tasks such as natural language understanding, but also for building expert systems (also known as knowledge-based systems), the accumulation and use of large amounts of problem-specific knowledge is essential (cf. Russell and Norvig, 1995, p. 258).

A definition of an entity is based on a decision on what kind of things should be valid entities in such KBs. And, not surprisingly, different researchers have found very different answers to

this question. Being philosophers, linguists, or computer scientists, they limited themselves to different entity types, e.g. “physical object, numbers, sets, times, possible worlds, propositions, events” (Hobbs, 1985, p. 61). The arguments, why one thing should be an entity and the other not, are obviously complicated and full of contradicting points of view.

We opt to not involve ourselves deeper into the discussion what an entity can or should be, but instead use a pragmatical approach and thus follow Hobbs (1985) “ontological promiscuity”: For us, just like for most creators of real-world KBs, an entity is everything there is that we can make a statement about – a definition which obviously holds true for anything we find in a given KB: Giving an example, when the RDF KB DBpedia contains the subject-predicate-object triple `db:Bob_Dylan rdf:type dbo:MusicalArtist` (cf. Figure 2.2), then, by definition, the subject Bob_Dylan becomes an entity here, simply because we make a statement about a subject being of type MusicalArtist. Note that this rather pragmatical entity definition, that is supported by RDF and Hobbs (1985), is actually not new, but was advocated already before in Philosophy, amongst others, famously by Quine (1948) who stated: “To be assumed as an entity is [...] to be reckoned as the value of a variable".

2.1.2 Knowledge Bases

The structured knowledge bases (KBs) we consider here, foremost DBpedia, but also Freebase, Yago, or Wikidata, can all be described as resource description framework (RDF) graphs. RDF (Wood et al., 2014) is a well-established framework for representing information in the Web, and is used by the above mentioned KBs – they are all (partially) derived from the Web encyclopedia Wikipedia and are made available via web applications and services.

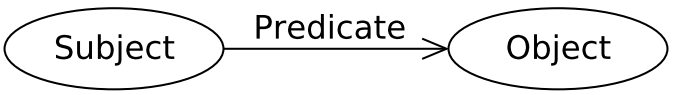

Figure 2.1: An RDF graph consisting of two nodes (subject and object) and a predicate connecting them (taken from Wood et al., 2014).

As defined by the W3C Recommendation: RDF 1.1 Concepts and Abstract Syntax (Wood et al., 2014), an RDF document or KB is a graph, consisting of a set of RDF triples. Each triple consists of a subject, a predicate, and an object, as visualized in the directed-arc diagram in Figure 2.1. An example of an RDF KB, represented as graph, is depicted in Figure 2.2 and shows a graph containing statements about person entities (Bob Dylan and Johnny Cash). We will use this example again later in Section 2.1.3 and also in Chapter 5, Figure 4.1.

In contrast to these fully structured KBs, we denote Wikipedia as being an only semi-structured KB: While it also knows disambiguated entities, the vast majority of the KB information is contained in the unstructured data, i.e. the natural language text of the articles describing each entity. In the following, whenever we want to emphasize that a KB is fully structured and representable as an RDF graph, we often call this KB a knowledge graph (KG) – a term that

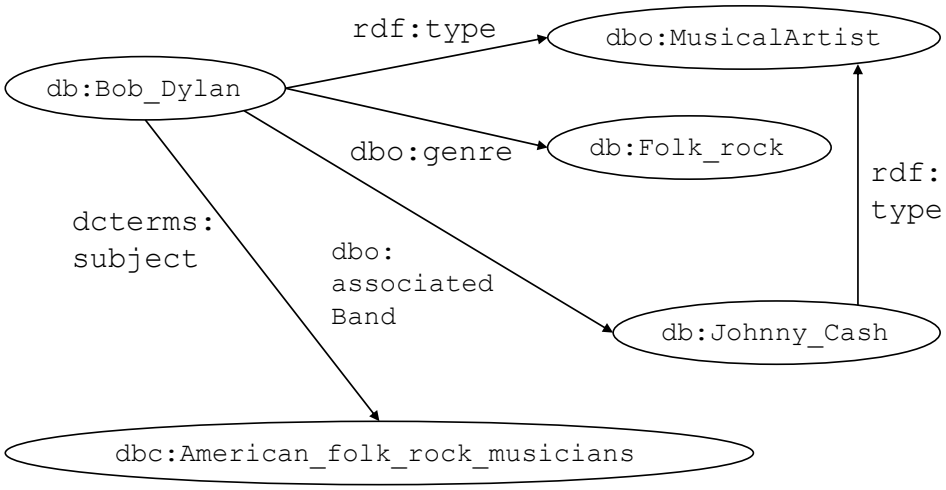

Figure 2.2: Example of an RDF graph (data taken from DBpedia). Nodes are subject or object entities. A directed edge represents a predicate that points from subject to object.

become popularly known due to Google's "Knowledge Graph", which is (one of) the KGs that serves (to a not publicly known extent) the entity search results of the Google web search.[1]

In an RDF graph, the subject and the predicate have to be identified by a unique internationalized resource identifier (IRI), while the object may contain an IRI or a literal. Because literals are by definition not unique but just string values, and can thus not be compared, we often exclude all literals in the following.

2.1.3 Wikipedia and DBpedia

Because we make use of Wikipedia and DBpedia as resources throughout all our experiments in one way or another, we describe both resources in more detail.

Wikipedia

Wikipedia is a collaboratively-written, community-built online encyclopedia and considered to be the largest and most popular general reference work on the Internet.[2] Each article contains a name, a natural language text (the actual article), a URL – commonly used as unique identifier when treating article pages as KB entities – and often, but not always, a property summarizing table on the right hand side, referred to as infobox, see Figure 2.3. Wikipedia also has a rich set of hyperlinks connecting articles with each other (cf. Kamps and Koolen, 2009) – however, in

[1] "Introducing the Knowledge Graph: things, not strings" from `http://googleblog.blogspot.de/2012/05/introducing-knowledge-graph-things-not.html`

[2] Cf. `https://en.wikipedia.org/wiki/Wikipedia`.

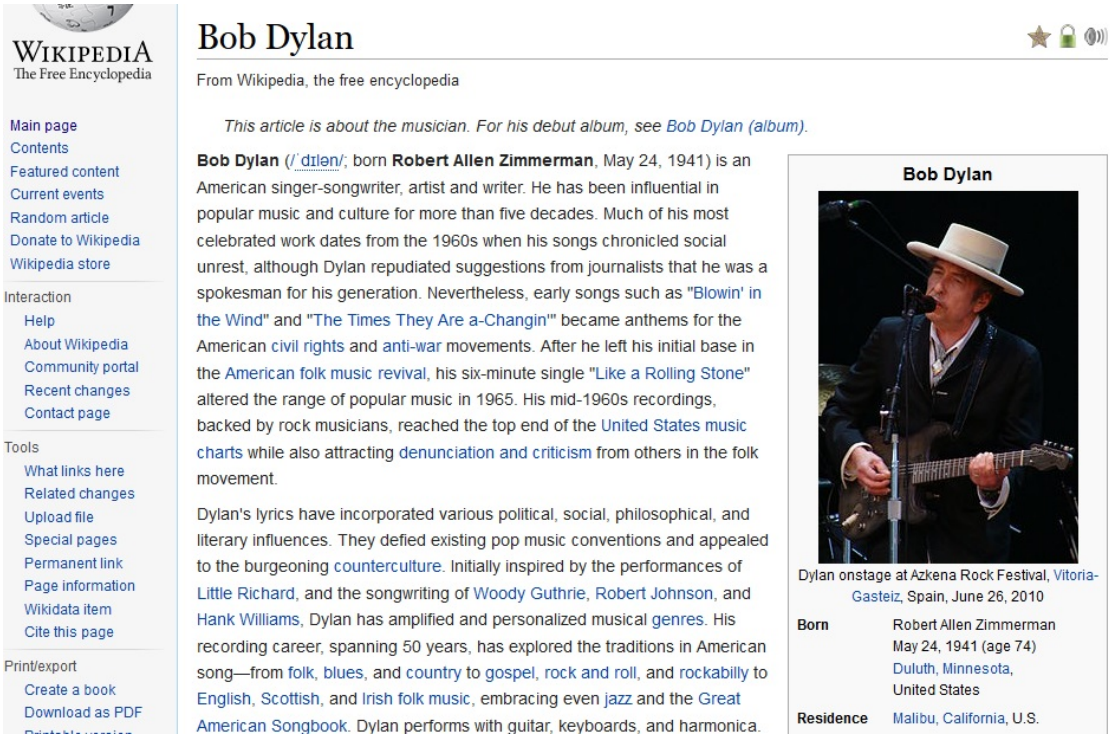

Figure 2.3: Screenshot of the Wikipedia article page for Bob Dylan, showing the entity (page) name, the article text, and the infobox on the right hand side summarizing some properties of Dylan (Wikipedia, 2016).

contrast to KGs like DBpedia, these are plain Web hyperlinks and do not contain any semantic meaning.

One of the most important functions of Wikipedia in the context of this thesis is to act as entity repository: When we want to annotate text with unique and unambiguous names for real word entities, such as persons or locations, we face the task of defining (and maintaining) such a repository of names (technically URIs in RDF). While this being a challenge in itself, when defining our own naming schema, i.e our own ontology, interoperability with other applications and datasets becomes highly challenging as each entity has to be correctly mapped to the third-party resource. The common solution to this problem is to not define your own entity repository, but instead, if possible, to reuse the Wikipedia article titles as entity identifiers, just like DBpedia does when deriving `db:Bob_Dylan` from the Wikipedia article "Bob Dylan". Wikipedia's high coverage, the additional entity information available from the article text, and the (untyped) hyperlinks connecting article pages have made it the first choice for many applications (Hovy et al., 2013).

While available in many languages, we use only the English Wikipedia, which contains around 5 million articles like "Bob Dylan", "Mannheim", "True North (novel)", "German Type U 151

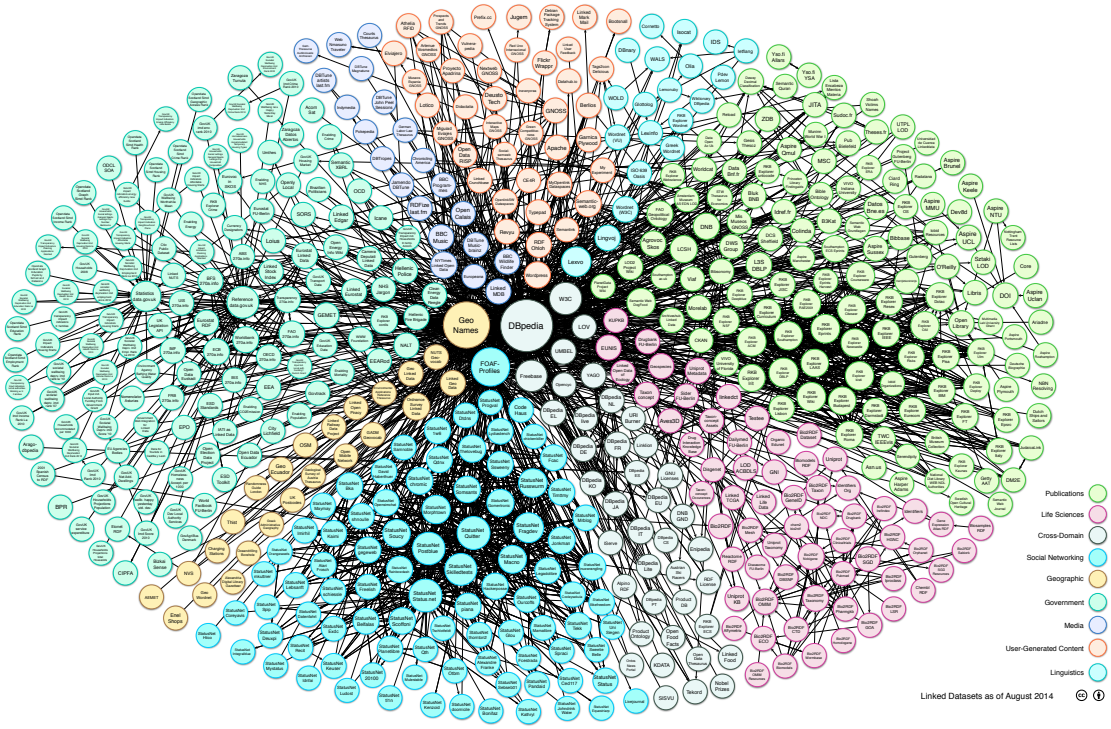

Figure 2.4: The Linking Open Data cloud diagram 2014[3] showing DBpedia in the center of the cloud.

submarine", or "Bipartite graph" – but also articles containing primarily lists like "Diving at the 2015 World Aquatics Championships – Women's 3 metre springboard" or articles with just one sentence like "KKBJ (AM)". As argued above in Section 2.1.1, we will consider in the following anything that has a Wikipedia article page to be an entity, even abstract entities such as `Bipartite_graph`.

DBpedia

In the following chapters, we frequently make use of DBpedia (Bizer et al., 2009) as a knowledge graph (KG), since it is a wide-coverage, encyclopedic KB with many (sometimes more than 1,000) fine-grained explicit semantic relations between millions of entities, organized as an RDF graph. See Figure 2.2 above for a subset of the DBpedia graph.

DBpedia is a community effort that extracts information from Wikipedia and makes this information available on the Web in various machine-readable formats.[4] It is also well-known for being the central entity repository authority within the linked open data (LOD) world, see Fig-

[3] By Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. Available from `http://lod-cloud.net`

[4] `http://www.dbpedia.org`

ure 2.4, because it reuses the Wikipedia article titles as entity identifiers (as explained above in the Section on Wikipedia).

The key idea behind DBpedia is to parse the Wikipedia infoboxes, which are the property-summarizing tables found on many Wikipedia pages, in order to automatically acquire properties and relations about a large number of entities. As the type of the infobox triggers which extraction template will be used, it also determines the type of the entity:

```
db:Bob_Dylan rdf:type dbo:MusicalArtist .
```

A type statement like `R rdf:type C` thereby denotes that the entity (the subject) is an instance of the class C. `dbo:MusicalArtist` is consequently a class (more precisely an instance of `rdfs:Class`). Besides the DBpedia Ontology types (which have the XML namespace `dbo:`) taken from the Wikipedia templates, DBpedia also contains type statements from other sources, most importantly the Yago types which are provided from the YAGO ontology (Hoffart et al., 2013) – a resource similar to DBpedia that was extracted from Wikipedia and WordNet.

The DBpedia extraction templates contain also extraction rules for predicates connecting entities with each other, like e.g.

```
db:Bob_Dylan dbo:genre db:Folk_rock .
```

Infobox entries which have no matching extraction rule are extracted nevertheless, but inserted into the `dbprop:` namespace. While increasing coverage, they can also often be hard to interpret, e.g. because the row in the infobox did not contain meaningful text (e.g. `dbprop:p`). For that reason, in the follow chapters we usually do not include the `dbprop:` properties. If not denoted otherwise, we will shorten the common namespace prefixes[5] of DBpedia using `db:` for `dbpedia:` resources, `dbc:` for the subset of resources denoting Wikipedia categories (`dbpedia:Category:`), and `dbo:` for `dbpedia-owl:` properties from the DBpedia ontology.

Two predicates in DBpedia are of particular interest: The first one is `dcterms:subject`, which reflects the extracted Wikipedia categories and makes them available as a taxonomy (Ponzetto and Strube, 2007) by connecting the categories via `skos:broader` predicates, e.g.:

```
db:Bob_Dylan dcterms:subject dbc:American_folk_rock_musicians .
dbc:American_folk_rock_musicians skos:broader
        dbc:American_rock_musicians .
```

The second noteworthy predicate is the `rdf:type`, which assigns one ore more entity types from the DBpedia OWL Ontology (`dbo:`) and/or the YAGO types (`yago:`):[6]

```
db:Bob_Dylan rdf:type dbo:Person .
db:Bob_Dylan rdf:type yago:MusiciansFromNewYorkCity .
```

[5] Full namespace URIs and their common prefixes can be looked up at `http://prefix.cc`.

[6] Note that DBpedia is increasing the type coverage. While the DBpedia and YAGO types are still predominant, the latest version DBpedia 1015-10 also contains e.g. types from Wikidata (`wikidata:Q215627`), Umbel (`umbel-rc:MusicalPerformer`), DUL (`dul:NaturalPerson`), and Schema.org (`schema:Person`) for the entity `db:Bob_Dylan`.

The type statements are not only interesting because of their high informativeness, but also because they link entities to a subsumption hierarchy, in which RDFS entailment can be applied on subclass relations like e.g. `dbo:Person rdfs:subClassOf dbo:Agent`. Consequently, following the RDFS semantics, every entity of type person is also of type agent (Hayes and Patel-Schneider, 2014). In the DBpedia datasets available for download or via the public SPARQL endpoint,[7] the type statements are materialized, i.e. it contains explicitly triple statements for all inferable super types.

A final remark on DBpedia: When in the following chapters DBpedia gets used, conceptually it will act as a placeholder for any kind of RDF KG, like e.g. YAGO (Hoffart et al., 2013), Freebase (Bollacker et al., 2008), Wikidata (Erxleben et al., 2014; Vrandečić and Krötzsch, 2014), or any other (even non-RDF) KB, as long as this KB has disambiguated entities and explicit semantic relations. To ensure this flexibility, the graph-based methods proposed below are agnostic w.r.t. the actual KG modeling, also because of many different options when modeling a KB – as we can see from the DBpedia-specific information described above.

2.2 Entity Linking

The task of entity linking (EL) is to annotate a given natural language text (document, sentence, fragment) with the KB entities mentioned in this text. EL systems thus provide us with the means to connect text to KB entities, as illustrated in Figure 2.5. In contrast to other systems, e.g. for named entity recognition (NER), the type of EL system we consider here has the sole purpose of annotating a given text (document/sentence/fragment) with Wikipedia entities.

We understand EL as the task of finding mentions in text, and link them to their (ideally correct) KB entity (we consider only Wikipedia entities throughout this thesis). This understanding of EL is also called end-to-end EL (Guo et al., 2013), wikification (Mihalcea and Csomai, 2007), or Annotate-to-Wikipedia (A2W) (Cornolti et al., 2013) – all thus highlighting that EL systems take text as input and return entities as output. This is in contrast to the definition of EL of e.g. the early TAC KBP Entity Linking task (Ji et al., 2010), which lacks the entity mention detection problem, as the entity mention (for entities of type person, geo-political entity, or organization) is already given to the EL system as prior knowledge.

The actual EL task in that context is then to find the correct KB entity to link to; or to link it to NIL if the corresponding KB entity does not exist).[8]

[7] SPARQL is an RDF query language which provides, amongst other things, a convenient way to query for subject-predicate-object triples; for details see `https://www.w3.org/TR/sparql11-query/`. The DBpedia endpoint is available at `http://dbpedia.org/sparql`.

[8] In the recent 2016 TAC KBP, there exists the Entity Discovery and Linking (EDL) task, which contains both steps, to find the named entity mention in text and to link them to the KB (cf. Committee, 2016).

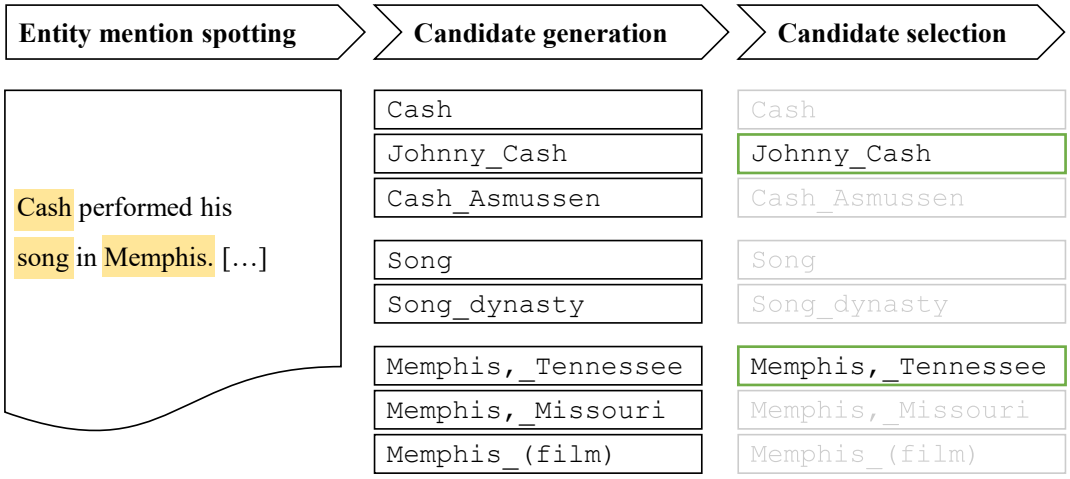

Figure 2.5: Generalized workflow of an entity linking (EL) system: The example shows how an EL system would annotate a given sentence with the (correct) entities `Johnny_Cash` and `Memphis,_Tennessee`, and how it would (deliberately) not link the entity mention "song" to any entity.

2.2.1 Common Methods

Typical EL systems for Wikipedia entities have three components/steps as shown in Figure 2.5: (i) entity mention spotting, (ii) candidate generation, and (iii) candidate ranking/selection (cf. Hachey et al., 2013; Olieman et al., 2014).[9]

In the *mention spotting* phase, the surface form mentions of an entity within the text document are identified, e.g. "Cash". This can be done e.g. by running a named entity recognizer, or simply by a string search for any known entity surface form as obtained from a lexicon created before. While both approaches are well-established, the first one has the advantage of being able to identify also entities that are not contained in the KB, thus allowing a NIL link. In contrast, the latter one is usually faster and more precises, which comes however at the cost of a decrease in coverage, as only known surfaces forms can be identified. Some EL systems perform also deeper linguistic analysis at this steps, e.g. coreference resolution.

Next, in the *candidate generation* phase, for each possible mention, entity candidates are generated. This step very often involves mining the Wikipedia hyperlink anchors and their frequencies in order to compute the likelihood of an entity being referred to from a certain surface form (Mihalcea and Csomai, 2007).

Last, in the *candidate selection* phase, the most likely (ideally correct) entity has to be selected

[9] See Ji and Grishman (2011) for a survey focusing on the disambiguation and ranking step in the context of the TAC KBP EL task.

from the candidates. To this end, supervised classification and the exploitation of the relationships between the entity candidates are common approaches (cf. Ji and Grishman, 2011). Many EL systems try to find the correct disambiguation of the surface form by making use of Wikipedia information. The basic assumption is that entities that occur together in one document (or within a certain text window size), are semantically related, and that this relatedness should be higher between the correct entities in contrast to the incorrect entity candidates. This idea is actually also pursued by us, when exploiting entity relatedness in DBpedia as a means for entity linking in Chapter 6. The most well-known approach for this problem is probably the measure by Milne and Witten (2008a) which considers the degree of overlap of the incoming Wikipedia hyperlinks as a proxy for their relatedness. Many EL can also make the decision to not link a found mention to any entity, either because all candidates seem to be too unlikely as to be linked, as illustrated in Figure 2.5 by the "song" mention, or because no candidates were found at all.

2.2.2 Common Systems

There exists a variety of commercial and academic EL systems. Because the linking step has an important influence on the final outcome of the subsequent pipeline following the EL, we often perform our experiments with two different systems, DBpedia Spotlight and TagMe, as a means to understand the influence of the EL step within the context of our task-specific findings. We opt for those two systems, because (a) DBpedia Spotlight is closely tied to the DBpedia project and offers the option to select/prune the candidate entity set based on entity type information, and (b) because TagMe was evaluated to be the most accurate and fastest EL system on different benchmarking datasets as reported in the evaluation by Cornolti et al. (2013). Note that our intention is not to provide an evaluation of the entity linking performance itself; for a dedicated EL evaluation of different systems see e.g. Cornolti et al. (2013), who also provide a good overview about some established evaluation datasets, and Usbeck et al. (2015) for a more general evaluation setting.

DBpedia Spotlight

DBpedia Spotlight was developed by Mendes et al. (2011) as a system to find and disambiguate natural language mentions of DBpedia resources, thus making it a standard EL system for Wikipedia entities. While we do not make use of it, DBpedia Spotlight provides the rather distinct feature of limiting the set of entities the input text can be annotated with by allowing the user to specify SPARQL queries over the DBpedia dataset. This allows to select e.g. only entities of a certain type (like politician) or with certain other properties (like nationality). The actual entity linking system consists of three stages, that correspond only partially to the prototypical steps described above, as the mention spotting is not generic but already limited to existing entity surface forms:

1. *Mention spotting* is based on a simple lexicon of surface forms referring to a DBpedia entity. The lexicion is harvested from Intra-Wikipedia link anchors (see above), entity labels, and redirects and disambiguation pages using the LingPipe Exact Dictionary-Based Chunker (cf. also DBpedia Lexicalization dataset Mendes et al., 2012).

2. *Candidate generation* in a strict sense does not exist in Spotlight, as the mention spotting is already based only on existing entity surface forms.

3. *Candidate selection*, i.e. the disambiguation is based on selecting the most likely candidates, computed within a vector space model (VSM) that represents each entity by the words found within all Wikipedia paragraphs mentioning that entity. Instead of standard term frequency–inverse document frequency (*tf-idf*), words are weighted by a schema called inverse candidate frequency (ICF), which captures according to Mendes et al. the discriminates power of a term with respect to the possible entity candidates. The candidate entities are then ranked by cosine similarity with the mention surrounding context.

While the authors report improvements in precision over the most frequent sense baseline by a large margin, the comparative evaluation by Cornolti et al. (2013) finds DBpedia Spotlight to be clearly outperformed by other systems, e.g. TagMe which we present next. Spotlight is available as stand-alone system and, as of 2013, as a public webservice.

TagMe

The TagMe system by Ferragina and Scaiella (2012) was created for the annotation of short text, and is thus appropriate for our usecases. It was evaluated by Cornolti et al. (2013) to be the best EL system on most of the general text datasets tested.[10] It is, in contrast to DBpedia Spotlight, a "classical" text-to-Wikipedia entity linking system and not tailed to DBpedia – which makes however no difference given the way we use both systems. TagMe operates similarly to DBpedia Spotlight and in the following steps:

1. *Mention spotting* and *candidate generation* is like in Spotlight based on a surface form lexicon harvested from Wikipedia.

2. *Candidate selection* in TagMe is different to Spotlight, as it builds upon the Wikipedia hyperlink structure via an adaption of the entity relatedness measure from Milne and Witten (2008a). Finally, a voting schema combines the relatedness among all candidate entities with the input text document, employing different heuristics for reducing computational complexity (instead of comparing all mention-candidate-pairs which would lead to a quadratic complexity in the mention input size) and ensuring annotation performance.

TagMe is available as stand-alone system and, as of 2013, as a public webservice.

[10] Cornolti et al. (2013) evaluated five well-known and freely available academic EL system, namely AIDA (Hoffart et al., 2011), Illinois Wikifier (Ratinov and Roth, 2009), TagMe (Ferragina and Scaiella, 2012), DBpedia Spotlight (Mendes et al., 2011), and Wikipedia Miner (Milne and Witten, 2008b).

Part I

Using Knowledge Base Entities

Chapter 3

Text Clustering using KB Types

In this chapter, we make a first step towards knowledge base exploration and study the usefulness of enriching text with Wikipedia entities and thus introduce additional entity type and category information for the task of clustering short text. The idea is to cluster those web search results snippets together which are semantically highly similar and thus would provide rather similar information to the user. Being a first step into the exploitation of KBs, in this chapter we make use only of the entities themselves and their type and category information, but not of any other KB relations that would e.g. connect entities directly. Considering those more complex KG relations will be studied in the subsequent chapters.

The work presented in this chapter has been published before as: *Michael Schuhmacher and Simone Paolo Ponzetto: Exploiting DBpedia for web search results clustering. In Proceedings of AKBC'13, pages 91–96* (Schuhmacher and Ponzetto, 2013).

The research questions (RQ) we aim to study here are centered around the overall question how KBs can be beneficial for textual tasks:

- RQ1: To what extent can an entity linking system be used to semantify short text and thus provide access to additional background knowledge (i.e. entity type and category information)?

- RQ2: Can clustering of short text benefit from the additional background knowledge?

We address this question in the context of the SemEval-2013 evaluation challenge on search results snippet clustering (Navigli and Vannella, 2013), as it provides us with an, even though indirect, but standardized evaluation setting (data and metrics). Our results show that clustering compact, topically semantified representations of snippets is indeed able to yield competitive performance on this task, thus indicating the viability of a knowledge-rich approach based on entity disambiguation techniques for complex, high-end Web applications.

3.1 Introduction

As introduced above, we look at the problem of clustering short texts from the Web, here search result snippets, to see whether this IR task can benefit from text semantification.

The specific task we follow here was defined by Navigli and Vannella (2013) in the context of the SemEval-2013 evaluation campaign. They assume a keyword web search setting with an ambiguous query, like for example "Apache". For such a query, a retrieval system would return documents about the HTTP Server Apache as well as about the helicopter Apache – at least, maybe even more. To mitigate the information a such a mixed results list of different document with different word interpretations/senses would pose to the user, Navigli and Vannella propose – like others before, see (Carpineto et al., 2009) for a survey – to cluster the search results, i.e. the returned documents which are here represented by their snippets. Each cluster should thereby cover one distinct interpretation of the ambiguous query, for example one cluster contains all Apache the server snippets/documents, and the other cluster all Apache the helicopter snippets/documents. The evaluation task differentiates between two types of systems for performing this snippet clustering: Word sense induction (WSI) systems, which have to cluster the snippets into semantically-related groups according automatically, and word sense disambiguation (WSD) systems, which had to label each given snippets with the appropriate senses, as taken from an external sense inventory, e.g. Wikipedia, thus implicitly determining a clustering of the snippets.

In our system, we first semantified the text snippets and then retrieve additional background knowledge from DBpedia as features for a standard clustering algorithm. The semantification is achieved by obtained the Wikipedia entities mentioned in the text from a state-of-the-art entity linking system, namely here DBpedia Spotlight and TagMe. Our approach uses DBpedia entities, which are actually the same as Wikipedia entities, identified in text as seeds to collect topical concept labels for the snippets. These are then used as features to cluster the snippets on the basis of their topical similarity, using the Wikipedia categories and the DBpedia types. Note, while we use DBpedia here as reference KG, our method could actually be used with any other wide-coverage knowledge resource and entity linker, e.g., YAGO (Suchanek et al., 2008; Hoffart et al., 2013) and AIDA (Hoffart et al., 2011).

We evaluate our approach within the experimental framework provided by the SemEval-2013 task (Navigli and Vannella, 2013) and use their evalution data and metrics (including the provided implementations).

3.2 Method

We present an approach to search results clustering based on the entities and their attributes as provided by Wikipedia (categories) and DBpedia (types).[1] Our method takes as input a collec-

[1] Technically, we retrieve both features from DBpedia, which contains the Wikipedia categories and provides them via the `dcterm:subject` predicate.

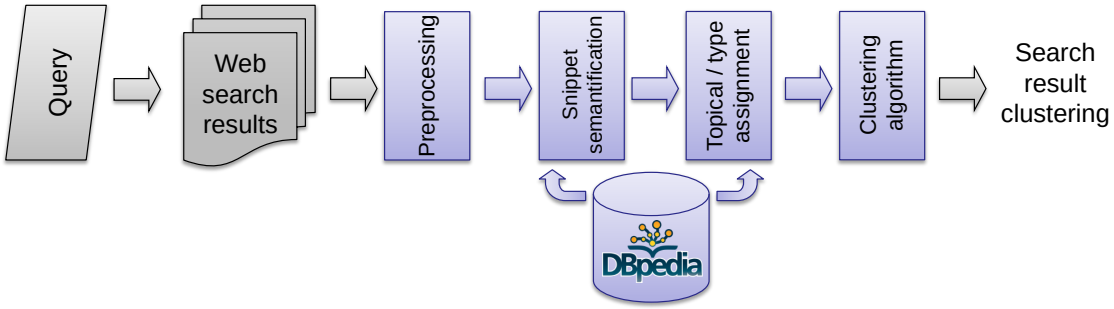

Figure 3.1: The workflow that annotates search results snippets with entities (and KB information) and yields cluster snippets.

tion of Web search snippets, and groups them together into topically coherent sets in order to provide the best clustering as output. The rationale is here that semantically similar snippets describe website which have the same interpretation of an ambiguous query. For instance, given a query such as "Apache", our dataset contains, among others, the following snippets, as returned by the Google search engine (Navigli and Vannella, 2013):

(1) "The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows ..."

(2) "The Boeing AH-64 Apache is a four-blade, twin-engine attack helicopter with a tailwheel-type landing gear arrangement, and a tandem cockpit for a two-man ..."

Each snippet identifies a separate meaning of "Apache" - namely, the software foundation and the helicopter, in our case. Accordingly, the task is to assign these snippets to different clusters, where each cluster contains snippets conveying the same meaning. We summarize the workflow of our approach in Figure 3.1. Key to our proposal is

(a) a semantified representation of the search result snippets as a bag of the most relevant topical concepts (i.e., types) associated with them,

(b) obtained on the basis of the structure of an underlying ontological resource, i.e., DBpedia.

We now turn to describe each component of our system in turn.

Data pre-processing

We first pre-process the snippets' text using a standard pipeline of NLP components, including stopword removal and WordNet-based lemmatization, as provided by the NLTK toolkit (Bird et al., 2009). Next, we filter out words having a comparably low discriminative power. To this end, we first compute for each word in the snippet a *tf-idf* score using the content of the webpages associated with each snippet. Words in the snippet with a *tf-idf* score below an experimentally determined threshold (as obtained by testing on a development dataset, see Section 3.3) are excluded from further processing. We perform *tf-idf*-based filtering mainly for two reasons, namely to

(a) provide the entity linker with a cleaner, highly discriminant context for disambiguation, and to

(b) remove common words which could otherwise be annotated with broad, domain-unspecific concepts.

Frequency statistics are computed directly from the snippets' documents in order to capture domain-specific usages of words (e.g., "Windows" being used as a proper name in snippet (1)). As output of this pre-processing step, we end up with snippets containing between 10 and 25 words on average per topic.[2] Given this small size, the corresponding snippets' word vectors are very sparse, and can hardly be used for any similarity computation (which is the basis for snippet clustering). In the next step, we thus acquire background knowledge from DBpedia capturing the snippets' topics, in order to overcome this sparsity problem.

Snippet semantification

We semantify the snippets by identifying the DBpedia entities contained. To this end, words and phrases are annotated with DBpedia/Wikipedia entities by an EL system, in our experiments we use DBpedia Spotlight and TagMe (cf. Section 2.2). The output of the entity linker consists of a set of disambiguated entities associated with corresponding words and phrases found in the snippet. For instance, for the two example snippets show above on page 21, the EL system obtains entities like

(1) `Apache_HTTP_Server`, `HTTP_Server`, `Unix` and `Microsoft_Windows` for the Apache web server

(2) `Boeing_AH-64_Apache`, `Attack_helicopter`, and `Undercarriage` for the Apache helicopter

We can see here two different effects of the EL. First the EL system was able, at least in this example, to correctly disambiguates between the two senses of Apache, helicopter and web server, even though the context around the entity mentions was altered in our data pre-processing, as described before. If this step would always return such flawless results, our task would already be solved as we found two different senses for the ambiguous query. However, this works of course not for all snippets and often the EL cannot distingush between the different senses – if the query word is contained in the snippet at all.

The second effect of the EL, and this is the more important because more robust one, is that the EL finds entities in the snippets that describe the conceptual context of the sense. In our example, we find `Microsoft_Windows`, a software, in the snippet referring to the Apache web server, which is also a software. And, in contrast, for the helicopter referring snippet, the EL returns the entity `Attack_helicopter`. Those two entities are obviously good features with a high discriminative power for the subsequent clustering algorithm.

[2] Note that we discuss how the EL works on this snippets in the next paragraph.

Acquiring KB information

The EL step extracts and disambiguates words and phrases by annotating them with unambiguous senses, i.e. KB entities. As stated above, these entities could, in principle, already be used directly as a representation for the snippets. However, questions remain on whether the resulting vectors would be too sparse (as indicated by results on the held-out data observed during prototyping). An alternative would also be to build a bag of words from the text contained within the Wikipedia articles associated with each identified DBpedia concept. However, this surface-level representation would still suffer from the same problems of the simple bag-of-words model, such as not being able, for instance, to capture synonymity – e.g., Wikipedia pages mentioning helicopter and chopper both providing evidence that the snippet belongs to the cluster corresponding to the `Boeing_AH-64_Apache` meaning of "Apache". Therefore, we incorporate structured knowledge encoded in DBpedia by retrieving additional entity attributes (via the public SPARQL endpoint, with DBpedia Version 3.8).

We query for all DBpedia and YAGO types denoted by the `rdf:type` predicate and all Wikipedia categories denoted by the `dcterms:subject` predicate, which have been previously found to provide useful information for topic labeling (Hulpus et al., 2013). As a result, we are able to assign type (from the YAGO and DBpedia namespace) and topical (from the Wikipedia categories) labels to all snippets. In our case, for instance, snippet (1) is assigned features such as `dbo:Software` and `dbc:Web_server_software`, whereas snippet (2) is labeled with concepts `db:Attack_helicopter` and `dbc:Military_helicopters`, among others. The final snippets' vectors contain only these types and categories, i.e., we leave out the words initially extracted from the snippets. The set of types and categories is thus a document representation by conceptual features, comparable to the Explicit Semantic Analysis approach Gabrilovich and Markovitch (2007), but created by making use of the explicit semantic relations provided by DBpedia.

Clustering

We finally cluster the snippets using their entity vectors, as obtained in the previous step. To this end, there exists a wide variety of clustering algorithms. In this work, we opt for affinity propagation clustering (Frey and Dueck, 2007), since it neither requires an a a-priori fixed number of clusters (like, for instance, *k*-means), nor it needs a similarity cutoff threshold (in contrast to hierarchical clustering). As standard practice, we manually tune all algorithm-specific parameters such as, for instance, the clustering damping factor, on our held-out data (see Section 3.3).

3.3 Evaluation

We evaluate our approach to Web search result clustering on a benchmarking dataset for this task, namely the data from the SemEval-2013 task on "Evaluating Word Sense Induction & Disambiguation within an End-User Application" (Navigli and Vannella, 2013).

System	RI	ARI	JI	F_1	# cl.	ACS
DWS-MANNHEIM-ESA	60.08	7.51	12.49	70.38	12.60	5.97
DWS-MANNHEIM-TAGME	60.49	8.72	13.09	**71.31**	11.98	6.04
DWS-MANNHEIM-SPOTLIGHT	61.53	9.15	15.68	70.94	11.32	7.12
DULUTH.SYS1.PK2	52.18	5.74	31.79	56.83	2.53	26.45
DULUTH.SYS7.PK2	52.04	6.78	31.03	58.78	3.01	25.15
DULUTH.SYS9.PK2	54.63	2.59	22.24	57.02	3.32	19.84
HDP-CLUSTERS-LEMMA	**65.22**	21.31	33.02	68.30	6.63	11.07
HDP-CLUSTERS-NOLEMMA	64.86	**21.49**	33.75	68.03	6.54	11.68
SATTY-APPROACH1	59.55	7.19	15.05	67.09	9.90	6.46
UKP-WSI-WACKY-LLR	50.02	2.53	**33.94**	58.26	3.64	32.34
UKP-WSI-WP-LLR2	51.09	3.77	31.77	58.64	4.17	21.87
UKP-WSI-WP-PMI	50.50	3.64	29.32	60.48	5.86	30.30
RAKESH	58.76	8.11	30.52	39.49	9.07	2.94
SINGLETONS	60.09	0.00	0.00	100.00	–	–
ALL-IN-ONE	39.90	0.00	39.90	54.42	–	–

Table 3.1: Evaluation results on text snippet cluster quality.

Experimental setting

The benchmark consists of 100 ambiguous queries (randomly sampled from the AOL search logs) for which there exists a finite set of possible meanings given by a corresponding Wikipedia disambiguation page. Each query comes with 64 search results, as returned by Google's Web search, which are then annotated with any of the meanings provided in the disambiguation page (plus an additional OTHER class used for snippets for which no sense is appropriate). For system development and parameter tuning, we use the Ambient (AMBIguous ENTries) dataset as held-out data. [3] Ambient was designed for evaluating subtopic IR and contains 44 ambiguous queries, the different senses were generated from Wikipedia disambiguation pages.

We report three different system configurations. First, our base system using DBpedia Spotlight (DWS-MANNHEIM-SPOTLIGHT), and second a version using TagMe as an alternative state-of-the-art entity linking system (DWS-MANNHEIM-TAGME) The third system (DWS-MANNHEIM-ESA) combines the affinity propagation clustering, which is also used by the two previous configurations, with the semantified snippets obtained from Wikipedia-Based Explicit Semantic Analysis (Gabrilovich and Markovitch, 2007) instead of entities. We use the Java-based ESA implementation ResearchESA by Philipp Sorg[4] with its standard configuration on the English Wikipedia and a fixed 1,000 vector dimensions cut-off.

[3] Available from `http://credo.fub.it/ambient`

[4] AIFB, KIT Karlsruhe, Germany. Code available from `https://code.google.com/archive/p/research-esa/`

System	K			
	5	10	20	40
DWS-MANNHEIM-ESA	37.65	54.74	69.77	86.05
DWS-MANNHEIM-TAGME	38.15	56.38	72.53	85.66
DWS-MANNHEIM-SPOTLIGHT	40.30	54.89	71.22	85.28
HDP-CLUSTERS-NOLEMMA	50.80	63.21	79.26	92.48
HDP-CLUSTERS-LEMMA	48.13	65.51	78.86	91.68
UKP-WSI-WACKY-LLR	41.19	55.41	68.61	83.90
UKP-WSI-WP-LLR2	41.07	53.76	68.87	85.87
UKP-WSI-WP-PMI	40.45	56.25	68.70	84.92
SATTY-APPROACH1	38.97	48.90	62.72	82.14
DULUTH.SYS7.PK2	38.88	53.79	70.38	86.23
DULUTH.SYS9.PK2	37.15	49.90	68.91	83.65
DULUTH.SYS1.PK2	37.11	53.29	71.24	88.48
RAKESH	46.48	62.36	78.66	90.72

Table 3.2: S-Recall@K for text snippet clustering.

Results

We report our results in Table 3.1, where we evaluate the quality of the clusters output by our method, as defined in the SemEval task using standard clustering measures from the literature – namely, Rand Index (RI), Adjusted Rand Index (ARI), Jaccard Index (JI) and F_1 measure (F_1). In addition, we report the average number of clusters (# cl.) and average cluster size (ACS) for our system, as well as those which participated to the SemEval task. Finally, we present in Table 3.3 and 3.2 our results in the clustering diversity sub-task evaluation – quantified as S-*recall*@K and S-*precision*@r. All performance figures were computed using the SemEval task's official scorer (cf. Navigli and Vannella (2013) for details).

Overall, we generally observe a favorable performance trend, as our system ranks among the best performing ones for this task. In the clustering quality evaluation, in fact, we are able to rank third out of 10 systems in the results for RI and ARI – i.e., right after HDP, the best approach for this task, consisting of a Word Sense Induction system based on Hierarchical Dirichlet Process Lau et al. (2013) – and achieve the best F_1 measure overall. Moreover, together with HDP, we are the only system performing above the baseline for RI[5]. Finally, we consistently beat by a large-margin on 3 out of 4 measures RAKESH, the only other knowledge-rich system that participated in the SemEval competition.

When looking at the properties of the clusters themselves (# cl. and ACS) we observe that our approach produces many medium-small sized clusters. We expect this to indicate that, in a Web search result diversification evaluation setting, our system shows a precision-oriented behavior.

[5]As typically the case, baseline methods are notably a difficult competitor for unsupervised and knowledge-rich sense disambiguation and induction systems.

System	r			
	50	60	70	80
DWS-MANNHEIM-ESA	42.07	31.90	27.23	21.80
DWS-MANNHEIM-TAGME	47.31	34.51	27.55	22.02
DWS-MANNHEIM-SPOTLIGHT	44.20	31.46	27.30	23.40
HDP-CLUSTERS-LEMMA	48.85	42.93	35.19	27.62
HDP-CLUSTERS-NOLEMMA	48.18	43.88	34.85	29.30
UKP-WSI-WP-PMI	42.83	33.40	26.63	22.92
UKP-WSI-WACKY-LLR	42.47	31.73	25.39	22.71
UKP-WSI-WP-LLR2	42.06	32.04	26.57	22.41
DULUTH.SYS1.PK2	40.08	31.31	26.73	24.51
DULUTH.SYS7.PK2	39.11	30.42	26.54	23.43
DULUTH.SYS9.PK2	35.90	29.72	25.26	21.26
SATTY-APPROACH1	34.94	26.88	23.55	20.40
RAKESH	48.00	39.04	32.72	27.92

Table 3.3: S-Precision@r for text snippet clustering.

This analysis is supported by the figures in Table 3.3 and 3.2, where we observe that our system generally ranks in the middle in terms of S-Recall@K, whereas it achieves a middle-high performance on S-Precision@r. The results, thus, seem to indicate that using type-level information from semantified snippets helps us focus on more precise meanings of the query terms.

The comparison of different variants of our system shows that using entity taggers consistently improves over ESA-based snippet semantification, thus indicating that a topically semantified representation of snippets can compete with a conceptual vector space model within a high-end task. Spotlight generally outperforms TagMe, and achieves the best performance in the cluster quality evaluation on all measures except F_1. The clustering diversity evaluation shows that Spotlight achieves a higher recall (for a lower precision) when compared with TagMe, which is in-line with previous findings from Cornolti et al. (2013) obtained from an intrinsic evaluation of entity disambiguation on Web text.

3.4 Related Work

Over the last years many researchers focused on the problem of Web search result clustering – see Carpineto et al. (2009) for a survey. A significant amount of work has been devoted to identify features which are useful for discriminating the search results' topics, including latent concept models (Osiński and Weiss, 2005), mining query-logs (Wang and Zhai, 2007), as well as using spectral geometry Liu et al. (2008) and graph-clustering algorithms applied to word co-occurrence graphs (Navigli and Di Marco, 2013).

The work probably closest to ours is that of Scaiella et al. (2012), who use graph-based representations of snippets for Web search results clustering. Their method also links the snippet

to Wikipedia entities (using TagMe). For comparing the entities with each other, however, they make use of the relatedness measure by Milne and Witten (2008a), similarly to Shen et al. (2012). In contrast, we use the explicit semantic information from Wikipedia (categories) and DBpedia (entity types) – but then "only" consider them as binary features in the subsequent clustering step, while Milne and Witten (2008a) represent the entity relatedness by weighted edges in a graph.

Navigli and Di Marco (2013) cast the problem of snippet clustering as a word sense disambiguation (WSD) problem, i.e. the task of identifying the different meanings of an ambitious term given a word sense inventory. However, because it seems unrealistic to have a sense inventory available that covers any possible sense returned by a real-world web search engine, Navigli and Di Marco propose actually to use Word Sense Induction (WSI), which is the automatic discovery of word senses, here query interpretations, from raw text, here search results snippets. This approach has thus the advantage of not be limited to the known query senses. Our method is in contrast somehow a hybrid approach between WSD and WSI: On the one hand side, we rely on DBpedia, thus a KB of fixed and limited coverage, when identifying the entities in the snippets. On the other hand side, as we only annotated entities found within the snippets, our KB does not actually need to contain all meanings, i.e. all senses of a query. For example, even if the Apache Helicopter would not be an entity in DBpedia, the snippets text would nevertheless mention Helicopter, Boing, or United States Army – all entities contained by DBpedia. Thus, we make use of background knowledge where available, but are not limited in case a specific query sense is not explicitly available from the KB.

While also aiming at snippet clustering, Jong and Lee (2008) focused in addition on finding meaningful labels for the clusters, interestingly using the DMOZ websites directory. Such methods are also referred to as description-centric approaches (cf. Carpineto et al., 2009) as they are, instead of data-centric approach like ours and those mentioned above, more focused on producing meaningful descriptions for each cluster of search results, motivated the understanding that clusters without labels provide only limited benefit to the users. The system computes a language model for each DMOZ category, and later applies the model to cluster the snippets and to obtain cluster labels. DMOZ, the Open Directory Project, contains manually created category tags for websites, and thus fulfills in a way a similar purpose here as the manually created DBpedia categories and types do for our method: They act as an external source of knowledge to overcome the vocabulary ambiguity issues of purely lexical clustering methods.

3.5 Conclusion

In this chapter, we presented our first step towards leveraging KBs information for text understanding, here for clustering short text fragments (Web search result snippets) according to their different word senses. Our experiments indicate that the pipeline of (i) use entity linking (EL) to extract entities from the text snippets, and then (ii) utilize DBpedia as wide-coverage knowledge resources for obtaining additional clustering features is a viable approach, as we obtain additional and sense-discriminating entities (RQ1). We furthermore conclude, that this approach

goes beyond a simply bag-of-words (BoW) clustering model and that, thanks to the additional KB information, can improve the performance of the clustering of the snippets according to their different senses (RQ2).

The obvious limitation of our approach is that it does not exploit any relational KG information that would connect entities with each other, e.g. the fact that Johnny Cash and Bob Dylan are related (`db:Johnny_Cash dbo:associatedBand db:Bob_Dylan`, cf. Figure 4.1). While this is, for the specific setting in this chapter's clustering task for word senses, not really a limitation, in general it seems very desirable to take into account any kind of information available from the KG. This also holds true in particular for the information not available via our simple approach, e.g. the Wikipedia category hierarchy (), which we did not explore in this setting. This hierarchy, modeled in DBpedia via the `skos:broader` predicate and extracted from Wikipedia, cf. Section 2.1.3, contains e.g. that for the category `dbc:United_States_military_helicopters` a broader category is `dbc:United_States_military_aircraft`.

For that reason, the next chapter will go into exploiting the full KG, and not just type and category information. Thereby, we will traverse the KG as a graph and make us of arbitrary KG relations – an approach that has the advantage of being agnostic towards the semantics of the specific KG vocabulary.

Part II

Using the Knowledge Graph for Understanding

Chapter 4

Entity Relatedness using the Knowledge Graph

In the previous Chapter 3, we focused on the usage of entity links and made only rather limited use of the KB information, i.e. DBpedia types and categories, for clustering. A natural extension of this approach is to exploit the KG in order to obtain information about the relationships between entities in general.

We pursue this idea in the following Chapter 5 and develop a method that models documents as subgraphs of KG entities and predicates in order compute the semantic similarity between document pairs. A prerequisite for being able to compare entity graphs with each other, is to be able to compare single entities first. For this reason, in this chapter, we will first develop a method to compute entity relatedness and study different predicate weighting schemata as an unsupervised method to compute a semantic relatedness measure for KB entities. The idea, to compute entity relatedness by exploiting DBpedia as a weighted KG, will then be used for the document modeling in Chapter 5 and also as a means to improve entity disambiguation for open information extraction in Chapter 6.

The work presented in this chapter has been published before as:

- *Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-based Graph Document Modeling. In Proceedings of WSDM'14, pages 543–552* (Schuhmacher and Ponzetto, 2014a).
- *Michael Schuhmacher and Simone Paolo Ponzetto: Ranking Entities in a Large Semantic Network. In Proceedings of ESWC'14 Satellite Events, pages 254–258* (Schuhmacher and Ponzetto, 2014b)

The research question we want to answer in this chapter originates from the nature of the knowledge graph used. Such knowledge graphs, like DBpedia or Freebase, but in principle any kind of state-of-the-art knowledge graph, have unique entities and – in contrast to resources like Wikipedia or simple hierarchies – labeled edges (RDF predicates). As the example in Figure

2.2 (page 9 in Section 2.1.3) shows, DBpedia contains the information that Bob Dylan and Johnny Cash performed music together (`dbo:associatedBand`). In contrast, from Wikipedia we could only extract that there is an HTML hyperlink connecting both pages, but we would not know what type of relation connects them – all relations look the same.

When comparing entities in unlabeled graphs like Wikipedia with standard graph theoretic measures, e.g. shortest path, all relations are of equal type and thus of equal importance. But for resources with multiple and labeled edges, the question arises how to make best use of the rich semantic contained in edge predicates like `dbo:associatedBand`, `dbo:author`, `dbo:birthPlace`, `dbo:influencedBy`, `rdf:type`, etc. Therefore, our research question for this chapter is as follows:

- RQ: How to compute semantic entity relatedness in a KG with unsupervised methods?

In the remainder of this chapter, we will try to answer this question by developing an unsupervised KB exploration method, before presenting the actual document modeling method that builds upon the findings from this chapter in Chapter 5.

4.1 Introduction

Key to our unsupervised approach for computing entity relatedness is the combination of a fine-grained relation vocabulary, the KB predicates, with information-theoretic measures of concept associativity to produce a weighted knowledge graph that relies on the information and structure encoded within its underlying knowledge graph. We use the DBpedia graph as described in Section 2.1.3, but our method can also be used with any other knowledge graph, e.g. YAGO (Hoffart et al., 2013), provided it has disambiguated entities and explicit semantic relations (predicates).

The remainder of this chapter is structured as follows: In the next section (4.2, Method) we describe only how to construct a subset of the DBpedia graph for a given set of input entities and weight the connecting graph edges in order to be able to compare pairs of entities. This method can be used to compute the relatedness of single entities and gets evaluated in Section 4.3. The next step, the representation of documents as knowledge graphs containing multiple entities and the comparison of those graphs is described in the next Chapter 5 (Document Modeling using the Knowledge Graph), where also the final evaluation of the semantic document similarity computation is then presented (Section 5.3).

4.2 Method

We present in the following a purely graph-based approach. The motivation for exploring graph-based methods originate from the fact that (a) they are general in nature, and can be used with *any* knowledge graph, i.e., a knowledge resource that can be viewed as a graph, regardless of its specific vocabulary; (b) they have been shown to be effective for language understanding tasks when combined with labeled and unlabeled resources (cf. e.g. Navigli and Ponzetto, 2012; Hoffart et al., 2011).

4.2.1 Semantic Graph Construction

Let C_{db} be the full set of DBpedia's entities and C an arbitrary subset of it, given as input – e.g., the set of entities mentioned within a document. In the first phase of our method, we create from the set of input entities a labeled, directed graph $G = (V, E)$ containing i) the entities themselves, ii) their semantic relations, as well as iii) any additional entity that is related to any of the input ones by means of some relation in the graph. That is, $C \subseteq V \subseteq C_{db}$ and $E \subseteq V \times R \times V$, where $r \in R$ is a relation (or predicate) found in DBpedia, e.g., `rdf:type`, `dbo:birthDate` or `dbp:genre`. Additionally, we want to associate a weight w with each edge $(v_i, r, v_j) \in E$, in order to capture the degree of associativity between the source and target nodes – i.e., how strongly related the two corresponding entities are. Note that we do not make any distinction between A-box and T-box statements, since we remain agnostic as to the specific vocabulary used by our underlying resource: Some knowledge graphs might model information as classes (e.g. `rdf:type BayernMunichSoccerPlayer`), while others define a predicate (`playsForTeam BayernMunich`).

To produce our semantic graphs, we start with a set of input entities C and create a labeled directed graph $G = (V, E)$ as follows: a) first, we define the set of nodes V of G to be made up of all input concepts, that is, we set $V := C$; b) next, we connect the nodes in V based on the paths found between them in DBpedia. Nodes in V are expanded into a graph by performing a depth-first search along the DBpedia graph and successively adding all outgoing relations r, thus adding all simple directed paths $v, v_1, \ldots, v_k, v'$ of maximal length L that connect them to G, i.e., $V := V \cup \{v_1, \ldots, v_k\}$, $E := E \cup \{(v, r_1, v_1), \ldots, (v_k, r_k, v')\}$. We filter out any administrative information and data using a list of stop-URIs provided by Hulpus et al. (2013) and extended by us.

As a result, we obtain a sub-graph of DBpedia containing the initial entities, together with all edges and intermediate entities found along all paths of maximal length L that connect them. In this work, we set $L = 2$ following evidence from previous related work (Navigli and Ponzetto, 2012; Hulpus et al., 2013).

Figure 4.1 illustrates an example of a semantic graph generated from the set of entities {`db:Bob_Dylan`, `db:Monterey_Country_Fairgrounds`, `db:Mozambique_(Song)`, `db:Johnny_Cash`}, e.g. as found within the sentence "Dylan played Mozambique at Monterey right before Cash". Starting from these seed entities, we perform a depth-first search to add relevant intermediates nodes and relations to G (here e.g. `foaf:Person` or `db:Folk_music`).

Finally, we obtain a semantically-rich graph: additional nodes and edges provide us with a rich structured context, in which the initial concepts are now connected by a variety of entities and explicit semantic relations.

4.2.2 Weighting KG Relations

The approach described so far simply connects a set of input entities by traversing the given knowledge, which is similar in spirit to graph-based approaches to Word Sense Disambiguation

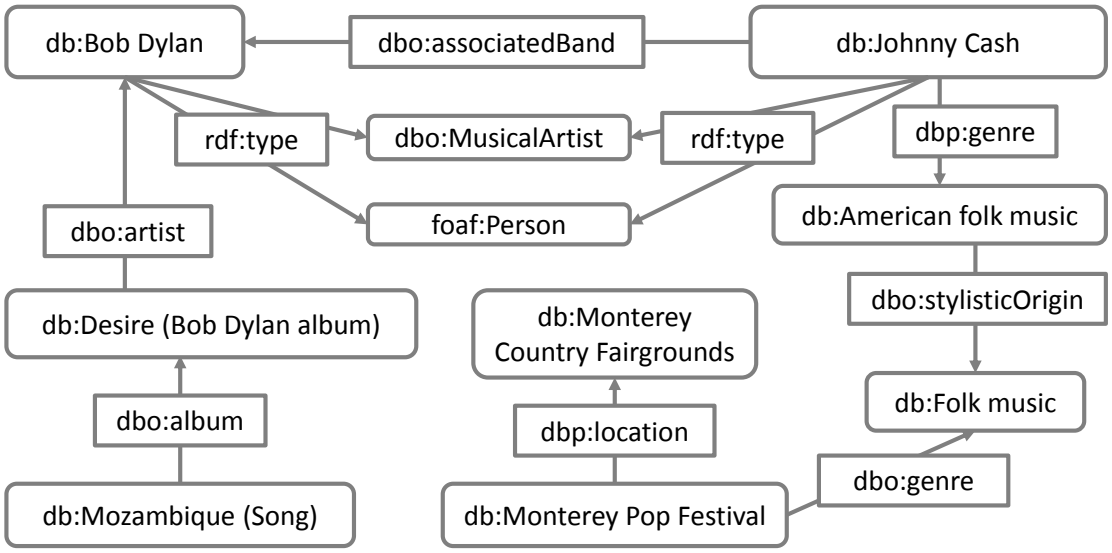

Figure 4.1: Illustrating example showing a part of DBpedia, represented as RDF graph where edges represent predicates that point from subject to object (cf. Section 2.1 with Fig. 2.1 and 2.2).

(WSD) using lexical resources (Navigli and Ponzetto, 2012) like WordNet. However, in contrast to lexical resources and to Wikipedia, our knowledge graph contains many different, fine-grained semantic relations.

But not all relations are equally informative, as we can see from Figure 4.1: There exist multiple paths between the source nodes `db:Bob_Dylan` and `db:Johnny_Cash`, which is often the case due to the typical high density of the DBpedia KG. And connecting paths include both, highly informative relations (e.g., the two entities being linked directly via `dbo:associatedBand`), as well as rather generic links (both entities being of `rdf:type` `foaf:Person`). The latter edge types tend to apply to a very many entities, here for example all persons, and thus carry only low discriminative power – e.g., in order to identify relations useful for entity relatedness. Thus, the question arises what kind of information to take into account when trying to make use of the KB relation information, e.g. when looking at graph paths within the KB graph.

One solution to this problem is to restrict the KB relations used to build semantic graphs to a manually-selected set of relations that capture the application domain well, as proposed e.g. by Hulpus et al. (2013). However, we want to overcome this manual and domain-specific step and opt here instead for an automatic approach based on relation-specific edge weighting. This is because, while a manual approach ensures overall good quality, it does not scale and needs to be tuned for every knowledge base in turn.

For this reason, we opt to enrich the KG by weighting its edges. Weights are meant to capture the degree of associativity between concepts in the graph – i.e., the degree of relevance of an

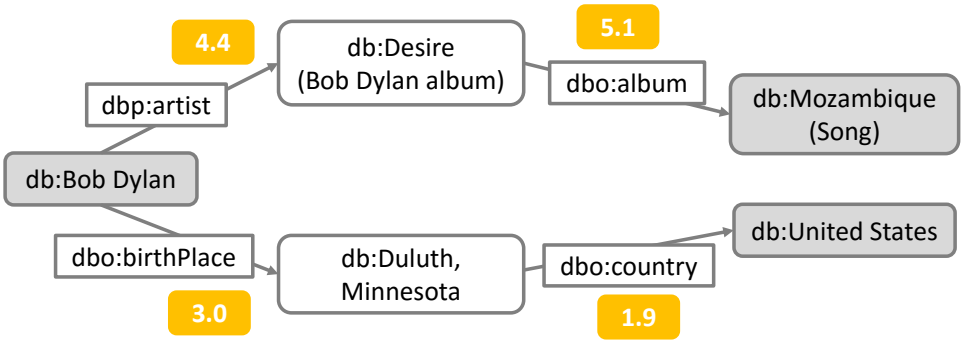

Figure 4.2: Example of two paths between entities with different semantic specificity in the DBpedia knowledge graph. Numbers illustrated the w_{combIC} weights: The more specific the edge, the higher the weight.

edge (i.e., semantic relation) for the entities it connects. The key idea underlying our weighting is to reward, for a given source node, those edges and target nodes that are most specific to it. An example is shown in Figure 4.2, where, starting from the entity Bob Dylan, high scores stand for a high specificity (upper path), while low scores are attached to comparably generic information. We formalize this intuition and propose different edge weight computation formula in the following.

At the core of our edge weighting lies the notion of information content (IC),

$$IC_{X_{Pred}}(\omega_{Pred}) = -\log\left(P(\omega_{Pred})\right), \tag{4.1}$$

where $P(\omega_{Pred})$ is the probability that the random variable X_{Pred} describing the type of edge, i.e. a specific semantic relation, shows the outcome ω_{Pred}. Giving an example, when assuming 1 out of 100 predicates in DBpedia are `rdf:type` statements, then

$$IC_{X_{Pred}}\left(\omega_{\texttt{rdf:type}}\right) = -\log\left(P\left(1/100\right)\right) = 2\,.$$

This measure makes the assumption that specificity is a good proxy for relevance – cf., for instance the `rdf:type` vs. `dbo:associatedBand` predicates. We can compute these IC values for all types of predicates, as we have the full DBpedia graph available and can query for all potential realizations of the random variable X_{Pred}. In the example in Figure 4.1, the edge labeled with `rdf:type` will accordingly get an IC which is comparably lower than, the one for `dbo:associatedBand`. The same effect is illustrated in Figure 4.2, where `dbo:country` has a lower score than `dbo:album` (note that these weights actually already also incorporate information about the triple object, as we will explain next when introducing the different weighting schema.

Based on this IC measure, we propose the following three edge weighting schema.

Joint Information Content (jointIC) While the information content of semantic relations provides us with a way to distinguish general vs. specific connections, it only covers the *a-priori* specificity of an edge, i.e., regardless of the entities it actually connects. However, as shown in Figure 4.1, the same type of edge, e.g. `rdf:type`, can lead to very general concepts with low discriminative power (`foaf:Person`), but also to very informative (because rare) ones, like `dbo:MusicalArtist`, which do, in fact, provide valuable information. We capture this by adding the conditional information content $IC\left(\omega_{Obj}|\omega_{Pred}\right)$ to our weighting scheme, which accounts for the concept the predicate is pointing to, given that the edge has already been observed. Formally, given an edge $e = (Subj, Pred, Obj)$ we compute the information content of the joint probability distribution, $IC\left(\omega_{Pred}, \omega_{Obj}\right)$, which we take as our weighting function:

$$w_{jointIC}\left(e\right) = IC\left(\omega_{Pred}\right) + IC\left(\omega_{Obj}|\omega_{Pred}\right) \tag{4.2}$$

In our example, the `rdf:type` edge leading to `dbo:MusicalArtist` accordingly receives a much higher weight than that pointing to the far more generic `foaf:Person`. Next, we present two alternative weighting functions that actually build upon the idea of jointIC.

Combined Information Content (combIC) Joint Information Content, although taking into account predicate and object specificity at the same time, can nevertheless penalize infrequent objects that occur with infrequent predicates – e.g., `db:American_folk_music` being overall very infrequent, but getting a high probability (and, hence, a low IC) when occurring conditional on `dbo:genre`. We propose to mitigate this problem by computing the joint information content while making an independence assumption between the predicated and the object. The resulting weights are then computed as the sum of the Information Content of the predicate and the object:

$$w_{combIC}\left(e\right) = IC\left(\omega_{Pred}\right) + IC\left(\omega_{Obj}\right) \tag{4.3}$$

Information Content and Pointwise Mutual Information (IC+PMI) An alternative way to compute the strength of association between the predicate and the object is by means of pointwise mutual information (PMI):

$$PMI\left(\omega_{Pred}, \omega_{Obj}\right) = \log \frac{P\left(\omega_{Pred}, \omega_{Obj}\right)}{P\left(\omega_{Pred}\right) P\left(\omega_{Obj}\right)} \,. \tag{4.4}$$

PMI measures the mutual dependence between the two variable outcomes ω_{Pred} and ω_{Obj}, and can thus be seen as a measure of how much deviation from independence there is between the two outcomes, i.e., the specific predicate and object found along a DBpedia graph edge. Our hunch here is to use PMI to find a middle ground between the assumption of full dependence (jointIC) or independence (combIC) between predicates and objects. We additionally combine PMI with the IC of the predicate, in order to bias our weights towards less frequent, and thus more informative, predicates:

$$w_{IC+PMI}\left(e\right) = IC\left(\omega_{Pred}\right) + PMI\left(\omega_{Pred}, \omega_{Obj}\right) \ . \tag{4.5}$$

4.2.3 Path Finding for Entity Relatedness

At this point, we start to make use of the weighted KG subgraph by computing a shortest weighted path between two entities as a proxy for entity relatedness. Entity relatedness can then be used e.g. to rank related entities, as shown in Figure 4.3, a task we will use later in Section 4.3 to evaluate our different weighting schemata.

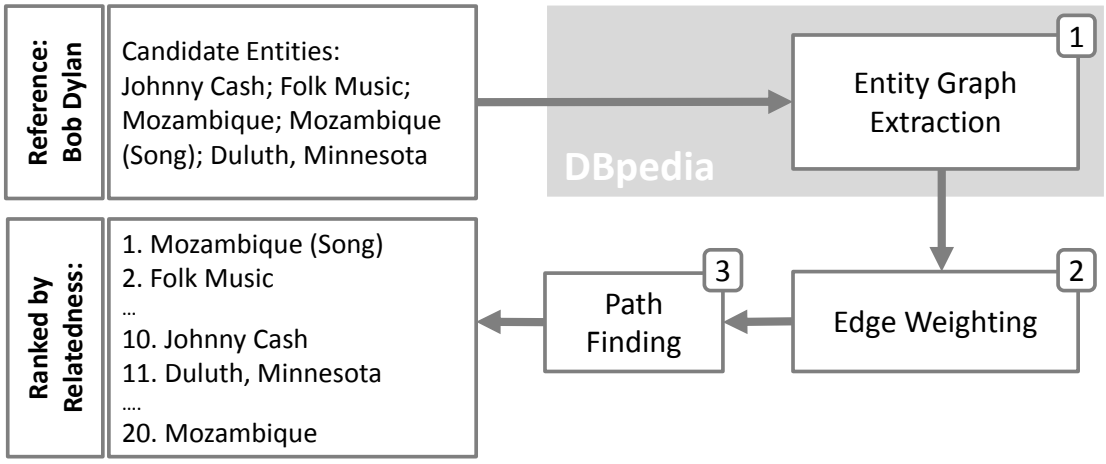

Figure 4.3: Workflow: From entities to weighted KG paths for entity ranking.

We now explain our approach to compute entity relatedness between pairs of entities. Given two entities E_1 and E_2 (both found in DBpedia), we perform the following three steps:

1) we build a semantic graph following the procedure of Section 4.2.1 using E_1 and E_2 as input entities.

2) we weight all graph edges e by edge cost ($cost(e)$), which is defined as

$$cost(e) = w_{max} - w\left(e\right) \ , \tag{4.6}$$

where $w(e)$ is any of the three weighting functions defined in Section 4.2.2, and w_{max} is the globally highest possible weight in the DBpedia graph for the selected weighting function. We need w_{max} as an upper bound to ensure that $cost(e) \geq 0$, as some graph algorithms cannot handle negative edge weights and we want to stay flexible for future applications, including the graph matching in Chapter 5.

3) we compute the minimum path cost between both entities – which acts as a measure of semantic distance,

$$distance(E_1, E_2) = \min_{p \in paths(E_1, E_2)} cost_p(E_1, E_2)\,, \tag{4.7}$$

where the cost of a path is calculated as the sum of the edge costs along the undirected connecting path p:

$$cost_p(E_1, E_2) = \sum_{e \in \{(E_1, r_1, v_1), \ldots, (v_k, r_k, E_2)\}} cost(e)\,. \tag{4.8}$$

As a result of our method, we obtain a measure of semantic relatedness between two arbitrary entities within our knowledge graph base as the inverse of their semantic distance as computed.

We briefly illustrate our method in Figure 4.2 with an example using `db:Bob_Dylan` and `db:Mozambique_(Song)` as input entity pair. When looking at the entity `db:Bob_Dylan` (the musician), we note that it is not directly connected to his song, `db:Mozambique_(Song)`. However, thanks to the fact that DBpedia encodes very specific facts – namely i) that Bob Dylan is the main artist of the album `db:Desire_(Bob_Dylan_album)`, and ii) that `db:Mozambique_(Song)` is a song contained in that very same album – we are able to estimate a high degree of semantic relatedness between the two input entities. Note that our weighting scheme plays a crucial role in estimating the degree of semantic overlap. If we look, for instance, to another entity pair such as the one consisting of `db:Bob_Dylan` and `db:United_States`, we note that in DBpedia these entities are connected by a short, albeit rather uninformative (because unspecific), path consisting of a single intermediate entity (`db:Duluth,_Minnesota`). Our weighting captures this by assigning a low weight to edges denoting general semantic relations such as `dbo:birthPlace` and `dbo:country`. As a result of this, we are able to state that `db:Mozambique_(Song)` has a stronger semantic relatedness than `db:United_States`, although both are connected to `db:Bob_Dylan` by a path of equal length.

4.3 Evaluation

In this chapter, we present our own experimental evaluation on an existing benchmarking dataset, see details below. A second, external evaluation of our measure was done later by Hulpuş et al. (2015), who find our work helpful for entity disambiguation. We summarize their findings in Section 4.4 (Related Work).

4.3.1 Experimental setting

In this setting, entity ranking (Hees et al., 2013; Hoffart et al., 2012) is the task of ordering a given set of entities on the basis of their relatedness with respect to a specific reference entity.[1] In

[1] Please note that "entity ranking" is thus understood here differently compared to Chapter 7 (Relevance Ranking of Entities), where the entity ranking task means not to compare entities against each other, but to retrieve KB entities that are relevant for a given non-entity, general keyword query (cf. Section 7.1.4). Both definitions of entity ranking are rather problem specific, and most readers will probably be more familiar with the INEX entity ranking task, in

our case, since we work with DBpedia as KB, we take, e.g., `db:Bob_Dylan` as reference and try to compute, how strongly `db:Johnny_Cash` is related to it, in comparison to `db:Folk_music` or `db:Mozambique_(Song)`, etc. This ranking task has the advantage that it provides a focused, extrinsic evaluation of our different weighting methods: besides, there exists established gold standard datasets against which we can compare our approach.

Ranking entities by relatedness can here be seen as similar in spirit to computing word relatedness (Zhang et al., 2012), except that in our setting we are given as input unambiguous entity references, rather than potentially ambiguous words. Besides, entity relatedness also plays a key role in entity linking (see Section 2.2), since many system rely on estimating the degree of relatedness between candidate entity references of different mentions in text. That is, within a global document-level EL approach, entity mentions can be jointly disambiguated by maximizing their degree of semantic overlap as obtained, for instance, from information stored within the target knowledge base – cf. e.g. the AIDA entity linking system (Hoffart et al., 2012).

4.3.2 KORE Dataset

We use the KORE entity ranking dataset from Hoffart et al. (2012). This dataset consists of 21 different reference entities from four different domains, namely IT companies, Hollywood celebrities, television series, video games, and Chuck Norris (a singleton dataset). For each ranking problem, Hoffart et al. selected a set of 20 candidate entities by extracting hypertext links from the corresponding Wikipedia article. As those entities were found to be related with different degrees to the reference entity, the final relatedness assessments were obtained from human judges using a crowd-sourcing approach. As an example, the entity "Apple Inc." (from the IT Companies category) is paired with, among others, the following other entities:

Reference Entity	Related Entity (Rank out of 20)
Apple Inc.	Steve Jobs (1), Steve Wozniak (2), ...
	NeXT (10), Safari (web browser) (11) ...
	Ford Motor Company (20)

Obviously, different entities have different degrees of relatedness with the concept of "Apple" as a company. "Steve Jobs", for instance, ranks highest, having been a key figure of the company. In the middle range, instead, we find related companies such as "NeXT", another company founded by Steve Jobs (rank 10). Finally, at the end of the ranking we find "Ford Motor Company", which is only marginally related to "Apple", being also an American company but from a completely different industry.

As KG, we use DBpedia (cf. Section 2.1.3) Version 3.8 with the same dataset configuration as available from the public SPARQL endpoint. This includes also materialized type (`rdf:type`) statements generated from the subclass hierarchy (`rdfs:subClassOf`) of the DBpedia classes.

which for a given query an entity/ a list of entities is retrieved from a reference knowledge base – for more details read Section 7.4.1.

Table 4.1: Rank correlation on the KORE (Hoffart et al., 2012) entity ranking dataset for each entity group comparing the different weighting schema (best results are bolded).

	Unwghtd	jointIC	combIC	IC+PMI
Hollywood Celebr.	0.639	0.541	**0.690**	0.661
IT Companies	0.559	0.636	**0.644**	0.583
Television Series	0.529	0.595	**0.643**	0.602
Video Games	0.451	**0.562**	0.532	0.484
Chuck Norris	0.458	0.409	**0.558**	0.506
All 21 Entities	0.541	0.575	**0.624**	0.579

Table 4.2: Rank correlation on the KORE (Hoffart et al., 2012) entity ranking dataset compared against other systems (best results are bolded).

Method	ρ
Unwghtd	0.541
jointIC	0.575
combIC	0.624
IC+PMI	0.579
Hoffart et al. (2012)	0.673
Milne and Witten (2008a)	0.610

We follow the original evaluation setting of Hoffart et al. (2012) and compute Spearman's rank correlation coefficient ρ per reference entity ranking. Overall results are then obtained by averaging over all reference entities in the dataset.

4.3.3 Results

We report the results in Table 4.1, where we compare our different weighting schemes from Section 4.2.2. As baseline we use an unweighted version of the DBpedia graph: this amounts to computing entity relatedness simply as a function of distance in the network. Looking at the overall performance of the three alternative weighting schemes for all 21 ranking tasks, we observe that combIC consistently outperforms the baseline and both jointIC and IC+PMI on three domains out of four. Looking at specific domains, we find that jointIC does not always

improve the baseline, as results for Chuck Norris and Hollywood celebrities are actually getting worse. Nevertheless, on average all 3 weighting methods improve the baseline, with combIC, which shows an average increase of 15.5% (statistically significant for each task at $p \leq .001$ level using a paired t-test), achieving the best results.

When compared with the original results from Hoffart et al. (2012) as shown in Table 4.2, our method achieves a performance slightly lower than their original proposal ($\rho = 0.673$), but outperforms all its approximations (ρ = 0.621 and 0.425). Overall, we take these results as indicator that our edge weight schemata are helpful when computing weighted path length, and

that the combIC weighting is the best choice in this setting.

4.3.4 Error Analysis

For getting a better understanding of the actual rankings created by our method, Table 4.3 shows two selected examples of a low-performing (`db:Apple_Inc.` with correlation 0.495)) and a high-performing (`db:Brad_Pitt` with correlation 0.723) ranking.

We can see that our method identifies the top entities rather well, but then, for the `db:IBM` ranking, it fails and ranks high unrelated entities like `db:New_York_Stock_Exchange`. This is because both entities are connected via multiple rather uncommon predicate paths, including: `db:IBM dbo:tradedAs db:S&P_500` while `db:New_York_Stock_Exchange dbo:exchanges db:S&P_500`. And, in addition, both entities are connected via `dbo:locationCity` to `db:New_York`. Such short paths with very specific predicates and/or objects will consequently get rather low path cost, and those end up at the top of the entity ranking – even though this is not desired here.

4.3.5 Effect of Top-k Paths

Until now, we always considered only the single cheapest path that connected two entities. This approach, however, does not take into account if multiple, distinct paths connect a given entity pair. It seems intuitive that this information would be relevant for computing entity relatedness, because when e.g. two person entities are connected not only by their place of birth, but also by their profession and in addition have both acted in the same movie, these multiple KG relationships should indicate a stronger semantic relatedness between both persons. Note that, given the high density of the KGs like DBpedia, there is potentially a large amount of connecting paths for an arbitrary input entity pair – even for entity pairs that are not related at all.

Consequently, we analyze the impact of considering multiple paths between a pair of entities, and aggregating evidence by averaging their costs to compute the final relatedness score. This approach should penalize entity pairs that are connected by one specific but also many unspecific paths in contrast to entity that are connected by many specific paths. We show the results in Figure 4.4. For all three weighting schemes, the performance of our method monotonically decreases with the number of top-k paths used for computing entity relatedness (evaluated using the same settings as above). The best results are obtained for $k = 1$, i.e. for taking the single cheapest path only, indicating that robust performance on this task relies on finding specific, highly informative paths – and thus meaningful semantic relations – between entities. Again, the best results are obtained using the combIC weighting, which outperforms all other measures for any k.

4.4 Related Work

The recent years have seen a significant amount of work on computing semantic similarity (Zhang et al., 2012). This is arguably because semantic similarity provides a valuable model

Table 4.3: Rank correlation for two single rankings from the KORE entity ranking dataset as examples for a high-performing, Brad_Pitt (correlation 0.723), and a low-performing IBM (correlation 0.490) output of our method. Scr denotes the relatedness score, Rk the according rank, and GS the rank according to the KORE gold standard from Hoffart et al. (2012).

IBM (correlation 0.490)				**Brad_Pitt (correlation 0.723)**			
Entity	**Scr**	**Rk**	**GS**	**Entity**	**Scr**	**Rk**	**GS**
Samuel_J._Palmisano	11.2	1	2	Angelina_Jolie	11.4	1	1
Armonk,_New_York	13.4	2	6	Rusty_Ryan	13.2	2	4
IBM_DB2	16.2	3	4	Plan_B_Entertainment	13.9	3	7
New_York_Stock_Exchange	16.9	4	19	Jennifer_Aniston	14.8	4	2
Hewlett-Packard	18.3	5	12	University_of_Missouri	16.9	5	10
Thomas_Watson,_Jr.	18.5	6	3	Fight_Club	17.2	6.5	3
Linux	24.0	7	11	Seven_(film)	17.2	6.5	5
Smarter_Planet	37.7	8	9	Shawnee,_Oklahoma	18.3	8	8
Rational_Software	38.6	9	5	David_Fincher	24.6	9	11
Nintendo	51.9	10	10	People_(magazine)	39.5	10	16
Xbox_360	53.0	11	8	Tom_Cruise	40.8	11	12
Dehomag	54.8	12	13	Guy_Ritchie	45.9	12.5	13
Herman_Hollerith	55.1	13	1	Robert_Redford	45.9	12.5	14
Human_Rights_Campaign	58.4	14	18	CNN	61.2	14	19
National_Medal_of_Science	64.2	15	17	Golden_Globe_Award	61.5	15	9
Nobel_Prize	64.6	16	20	ONE_Campaign	61.6	16	15
Cell_(microprocessor)	68.4	17	7	Sudan	63.9	17	18
Edwin_Black	72.8	18	15	Nice	68.3	18	17
Six_Sigma	74.5	19	16	Pakistan	68.5	19	20
Service-oriented_architecture	79.3	20	14	Achilles	999.0	20	6

of semantic compatibility that is widely applicable to a variety of tasks, including both preprocessing tasks like Word Sense Disambiguation (Patwardhan et al., 2003) and coreference resolution (Ponzetto and Strube, 2007), but also high-user applications such as information retrieval (Egozi et al., 2011) or multi-document summarization (Nastase, 2008).

Most of the previous work on semantic similarity has concentrated on computing pairwise similarity of words, although recent efforts concentrated on the broader task of text similarity (Bär et al., 2011), as also shown by community efforts such as the shared tasks on Semantic Textual Similarity (Agirre et al., 2013). Overall, the best results in these evaluation campaigns have been obtained by supervised models combining large feature sets (Bär et al., 2012; Šarić et al., 2012), although questions remain on whether this approach can be easily ported to domains for which no labeled data exists. In contrast, in this work we presented an unsupervised model that requires virtually no parameter tuning and exploits the implicit supervision provided by very large amounts of structured knowledge encoded in DBpedia.

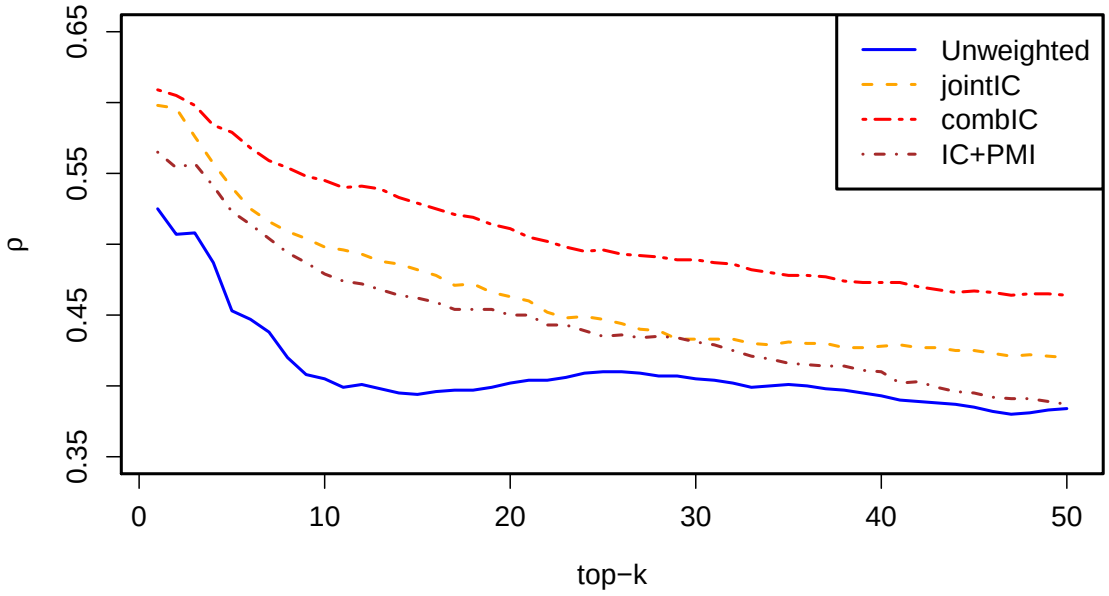

Figure 4.4: Results using top-k average path costs.

4.4.1 Semantic Relatedness of Words

Methods on semantic similarity of words can be broadly categorized into corpus-based and knowledge-based approaches (cf. Hassan and Mihalcea, 2011). The knowledge-based methods extract information from manually created resources, in particular from lexical taxonomies like WordNet (Fellbaum, 1999) – which is the actual strength of those methods: they use a resource specifically created to describe the relationships between words. Wu and Palmer (1994), for example, exploit the WordNet hierarchy to find the least common superconcept of a pair of verbs. The shortcoming is obviously the limited coverage and the high cost of manually creating such task-specific resources. Our approach can also be classified as a knowledge-based method, however, we build upon a general purpose KB that serves many different application and use-case.

Corpus-based method try to overcome human effort of creating knowledge resources, and rely instead on existing text corpora as information resource, thus being easily scalable to large amounts of text. They often represent text as a BoW and compute e.g. the PMI between word as a measure for their relatedness (Church and Hanks, 1990). Another, well-known method is latent semantic analysis (LSA) by Landauer and Dumais (1997), which builds upon a standard term-document matrix, describing term occurrences per document. The key idea of LSA is to find a low-rank approximation, i.e. to performing a singular value decomposition, of this matrix, resulting in a matrix with fewer dimensions then the original term-document matrix. By reducing the number of term dimensions, dimensions representing related words get conflated into (ideally) one dimension representing the abstract/latent word sense that semantically similar (or synonym), but syntactical different, words have in common. LSA is thus also a method to

overcome the vocabulary mismatch problem in IR or text clustering.

The same idea is utilized by explicit semantic analysis (ESA) from Gabrilovich and Markovitch (2007), who represent words as concept vectors. But instead of generating the vector space via dimension reduction from an arbitrary corpus, ESA uses the word-document-matrix (BoW approach) of all Wikipedia articles. A word is then represented by a vector, consisting of the top-k documents, i.e. Wikipedia articles (in the end, Wikipedia entities), in which the given word was observed. Semantic relatedness of two words is then computed as the cosine similarity between the two vectors (which is the standard approach to compare vectors in a vector space model). ESA was proposed for computing semantic document similarity, we compare our own method against ESA below in Section 5.3.

4.4.2 Semantic Relatedness of KB Entities

Semantic relatedness of (KB) entities is an important task for many applications that deal with natural language text, including word sense (or named entity) disambiguation for entity linking (Milne and Witten, 2008b; Pehcevski et al., 2008), general word sense disambiguation (Navigli, 2009), or knowledge base population (Dutta et al., 2015). Semantic relatedness of entities from KBs like Wikipedia is also often discussed within the semantic web community. The work from Passant (2010), for example, aims at computing semantic distances on linked data for the purpose of entity recommendation. But as entity recommendation is the task, it relies (naturally) on disambiguated input, which is, however, a requirement hard to satisfy for most applications working with arbitrary natural language text like we do.

The work from Hoffart et al. (2012) proposes a method to compute semantic relatedness of KB entities and is thus closely related – we actually use their evaluation gold standard in our own experiments, cf. Section 4.3. Their motivation for estimating semantic relatedness originates from the aim to build an entity linking system. As explained in Section 2.2, when having multiple candidate entities for multiple mentions within one document (or sentence, or text fragment), the information on how related the KB entity candidates are can be helpful for finding the correct disambiguation and in entity selection/ranking step.

Instead of relying only on the entity connecting Wikipedia hyperlinks for relatedness computation (cf. Milne and Witten (2008a) and Section 2.2), Hoffart et al. propose a measure based on the overlap of keyphrases (*Keyphrase Overlap Relatedness*; KORE). The keyphrases are obtained from the Wikipedia page of each entity, extracting link anchors of internal and external links, titles of citations, and names of categories, which are then weighted by inverse document frequency (*idf*). The work also makes a contribution regarding the efficient computation using locality-sensitive-hashing (LSH), which we will not discuss here.

Interestingly, the selection of sources for the keyphrases show a (partial) overlap with the KG information we use for computing semantic relatedness, i.e. the usage of Wikipedia categories. But instead of working on the surface forms of the links and categories, we in contrast go into the other direction and leverage the even more structured DBpedia links, instead of just the Wikipedia hyperlinks.

The most relevant related work is from Hulpuş et al. (2015), who built upon our work and explored the benefits of KG path-based semantic relatedness measures for word and entity disambiguation – motivated, like us, by the aim to go beyond Wikipedia-based measures and exploit the semantic information of KGs. In their system for word and entity disambiguation, they also implemented our *combIC* metric (cf. Section 4.2) as one of the KG-based entity relatedness measures. As part of their extensive experimental evaluation, the authors also evaluate the disambiguation step, i.e. the test how well the selection of the correct entity for a given list of candidate entities works for a specific noun-phrase.[2]

When studying the influence of the entity relatedness measure in this setting, they find that "while CombIC achieved much worse performance when evaluated against human assessment of relatedness, it achieved the best disambiguation capability" (Hulpuş et al., 2015, p. 455) and outperformed the other two KG-based methods on all five evaluation datasets. Our *CombIC* measure achieves always the best F_1, due to its comparably higher precision. Because of the rather contradicting performance, Hulpuş et al. (2015) conclude that their findings indicate "that for disambiguation, measures must have additional properties than correlation to human assessment of relatedness."

4.5 Conclusion

In this chapter, we proposed a method to estimate semantic relatedness of entities within the DBpedia knowledge graph. Entity relatedness is thereby computed as the shortest path in a weighted version of the KG, where the weighting and path finding is a purely unsupervised approach. We proposed different information-theoretic measures to weight the semantic relations, and automatically quantify their degree of relevance with respect to the entities they connect. Edges in the semantic graphs are thus weighted so as to capture the degree of associativity between entities, as well as their different levels of specificity.

When evaluating our approach via the task of entity ranking on the KORE dataset, we show that weighting the graph outperforms an unweighted exploration, but also that the specific weighting schema matters; we find combIC to perform best. In comparison with other well-established method (Milne and Witten, 2008a), we can show a gain in performance, but are not as good as the state-of-the-art method by Hoffart et al. (2012) for entity ranking. However, later work by Hulpuş et al. (2015) adapted and evaluated our graph metrics for disambiguation and found that, while the combIC measure "achieved much worse performance when evaluated against human assessment of relatedness, it achieved the best disambiguation capability".

In summary, we have developed a method to compute entity relatedness within a KG with labeled edges, and have thus gone beyond the Wikipedia hyperlink based approach (Milne and Witten, 2008a). Having obtained experimental confirmation that our approach works in principle, we can now make the next step in Chapter 5 and move towards representing whole documents as

[2]This is similar to the evaluation of disambiguating NELL subjects/objects to DBpedia entities presented in Chapter 6.

knowledge graphs and compare them with each other for computing semantic document similarity.

Chapter 5

Document Modeling using the Knowledge Graph

After having developed a method to compute semantic relatedness of single entities, we are now going to build upon that approach and present our method to model documents as knowledge base subgraphs – with the intention to compute the semantic similarity between document pairs in the end. The key idea is to model a document as a set of KB entities and then use the DBpedia KG to compute how semantically similar those KG subgraphs are, using an adaption of graph edit distance. This chapter presents thus the "strongest" integration of text document and KG in the context of this thesis, as the document itself will be represented as a subgraph of the KG.

The work presented in this chapter has been published before as: *Michael Schuhmacher and Simone Paolo Ponzetto. Knowledge-based Graph Document Modeling. In Proceedings of WSDM'14, pages 543–552* (Schuhmacher and Ponzetto, 2014a).

Our research questions for this chapter arise from the problem at hand, namely computing semantic document similarity. Following the thesis' main perspective on how to make use of entity links and the findings from Chapter 4 on how to compute entity relatedness within a KG, the subsequent questions to be address in this chapter are:

- RQ1: How to project text documents onto a knowledge graph (KG)?
- RQ2: How to compare documents that are represented as KGs while integrating the semantic KG information available?

In the remainder of this chapter, we will try to answer these questions before presenting another application of our entity relatedness method in the context of NELL fact disambiguation in the following Chapter 6.

5.1 Introduction

Being able to compare document is a key capability for different document processing tasks, including document retrieval and document clustering, and requires a defined understanding on what a document is, i.e. how to represent a document. Naturally, traditional approaches on document modeling draw upon document representations that rely solely on morpho-syntactic information by means of "flat" meaning representations. Probably most well-known and widely-used and -adapted are vector space models (for an overview see e.g. Turney and Pantel, 2010). However, more recent research moved towards a "deeper" representation of meaning of document content, which includes conceptual (Gabrilovich and Markovitch, 2007) and grounded (Bruni et al., 2012) vector spaces models, indicating the usefulness of semantic information for improving document comparison, also for more high-end tasks in IR and NLP (Hovy et al., 2013).

With this work, we take the next step in knowledge-rich document models and incorporated explicitly external knowledge by not integrating knowledge into the document, but in contrast take the document and represent it within the space the external knowledge base. While Gabrilovich and Markovitch (2007) introduced the well-received idea of representing documents as vectors of Wikipedia articles, the exploration of wide-coverage knowledge bases, such as YAGO (Hoffart et al., 2013) or DBpedia (Bizer et al., 2009) which are fully structured resources in contrast to Wikipedia, for such tasks has not been studied extensively yet.

Because we aim for an experimental evaluation of our document modeling method, we choose to test its capability to compute semantic document similarity, which is essentially the intention when defining a document model for real world applications like document retrieval. We rely on the notion of semantic document similarity as used by Lee et al. (2005), who also provide a ground truth dataset for evaluating computational methods to compute semantic document similarity which has been widely adopted (Gabrilovich and Markovitch, 2007; Hassan and Mihalcea, 2011, and others).

5.2 Method

In the previous chapter, we have already proposed a method to compare single KB nodes using weighted KB relations. In this chapter, we now turn to representing whole documents as KG subgraphs, and compare those subgraphs in order to determine the semantic similarity of the original documents.

We provide an overview of our approach in Figure 5.1, starting from the output of an entity disambiguator, which is used to identify a set of concepts from the input texts (1). Next, connecting paths between entities are collected, in order to identify the sub-graph of DBpedia covered by each document (2). Nodes in the semantic graph consist of concepts capturing the main topics of the documents: in addition, edges in the graph are weighted to identify the semantic relations that are most relevant for these concepts (3). Finally, we view computing semantic similarity as a matching problem between the concepts of different documents, and apply a Graph Edit

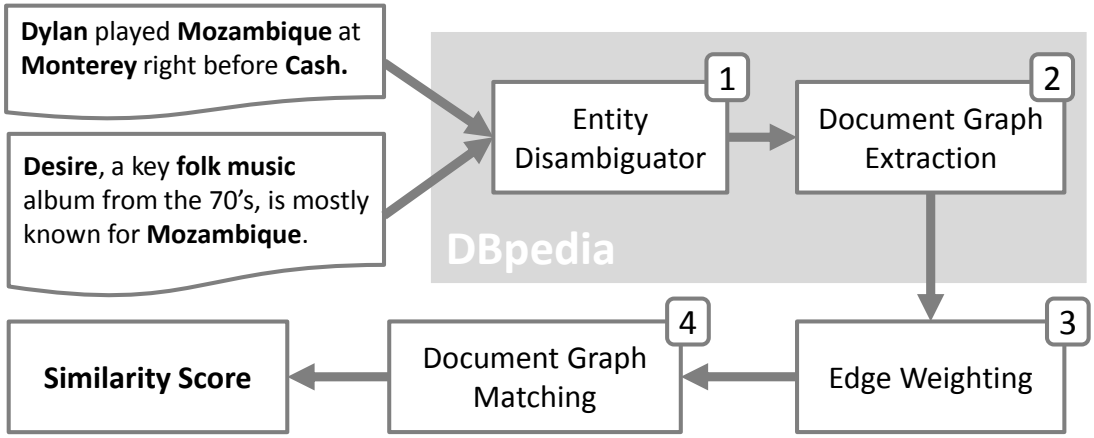

Figure 5.1: Workflow: From document pairs via entity linking, KG construction, and edge weighting to graph matching for computing semantic document similarity.

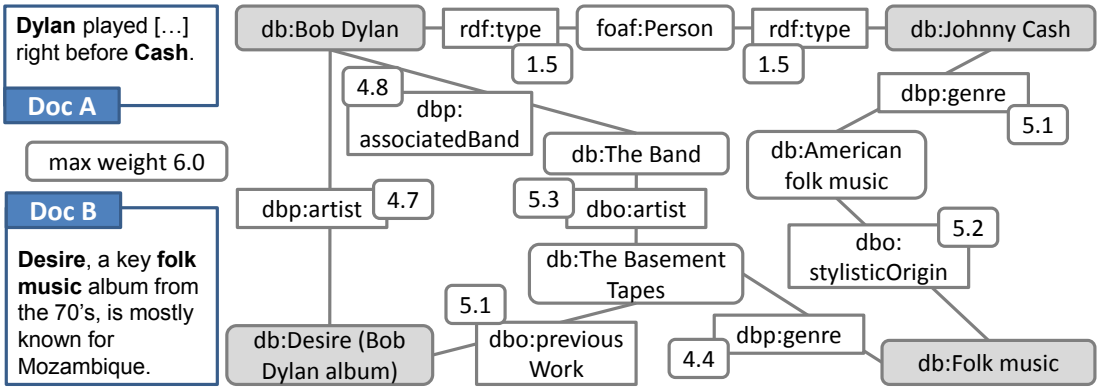

Figure 5.2: Example of a weighted KG representing two text documents for semantic document comparison (numbers on the graph edges indicate edge weights).

Distance based similarity measure, which relies on the Hungarian method, to identify the 'best' connecting paths between the documents' concepts (4). As a result, we are able to output the degree of similarity of the two input documents.

5.2.1 Document Graph Construction

Given an input text document, we first semantify the documt by identifying the set of concepts it contains. To this end, words and phrases are annotated with DBpedia entities using an arbitrary entity linking system (cf. Section 2.2) – in our experiments we opt again for DBpedia Spotlight and TagMe. Given a mention and its candidate entities, the entity linker finds its most likely meaning in context – e.g., like DBpedia Spotlight (Mendes et al., 2011) using a Vector Space Model (based on a bag-of-words approach).

Accordingly, given an input document, we are able to obtain a set of disambiguated KG entities and their associated surface form words/phrases as mentioned in the text. In the two example documents of Figure 5.2, we extract key concepts like `db:Bob_Dylan`, `db:Johnny_Cash` and `db:Desire_(Bob_ Dylan_album)`. We call these extracted concepts the source nodes V_s^d of a document graph $G^d = (V^d, E^d), V_s^d \subseteq V^d$ representing document d.

The document graph G^d is then built by applying the procedure described in Section 4.2.1, which means essential:[1] Starting from the set of input entities V_s^d, we explore the outgoing links, i.e. predicates, and add the triple objects, if they are entities, to our subgraph – literals will not be considered. This graph exploration is limited to a fixed number of two hops, $L = 2$, which is the same settings as for the entity relatedness task, cf. Section 4.2.1. We thus get a subgraph of DBpedia, in which the source entities describe the actual document content found (via the EL), and where the surrounding entities and relations/predicates contain relevant background knowledge – which will be used next to compute the semantic relatedness of two documents.

5.2.2 Graph-based Document Similarity

Since we represent documents as weighted DBpedia subgraphs, we can naturally formulate computing document similarity as a graph matching problem. While there exist exact graph matching algorithms based on graph isomorphism, we require our measure to be able to effectively quantify degrees of similarity. Consequently, we opt for an application of graph edit distance (GED) for our specific problem. GED (Gao et al., 2010) is a general, inexact graph matching method that defines the distance between two graphs in terms of the minimum cost of edit operations needed to transform one graph into the other. It thus follows the same idea as the edit distances for strings (Levenshtein, 1965).

In general, a GED measure needs to define edit cost functions for insertion, deletion, and modification for both nodes and edges. However, given our specific problem setting, we drop some of these requirements and define only cost functions for nodes. This is because, given a pair of semantic graphs, generated using the method from Section 5.2.1, these actually consist of two subgraphs of the same supergraph, namely the DBpedia KG. As a result, no cost function over edges needs to be defined, since an edge existing or not in one graph will also be present or not in the other, given the fact that both document graphs belong in fact to the same supergraph. Thus, edit operation on edges solely cannot occur and, accordingly, we define edge cost functions to yield zero.

We define cost operations for nodes as follows. Note that, since we work with a well-defined ontology that represents concepts by unique URIs, we can rely on the fact that nodes in the DBpedia graph are unique. As a result, we do not need to account for label mismatch between concepts – e.g., the entity "Bob Dylan" being identified by `db:Robert_Allen_Zimmerman` in a graph, and referred to as `db:Bob_Dylan` in another one (or vice versa). Thus, in contrast to standard GED approaches, we define node modifications on the basis of the underlying edge

[1] This is only a brief summary, please go back to Section 4.2.1 for the detailed method.

structure, i.e., weighted distances in the graph, as opposed, for instance, to the application of string similarity measures like Levenshtein distance on node labels.

The modification cost between two nodes is defined, analog to Section 4.3, as the sum of the edge costs along their connecting path (cf. Equation 4.8). By employing our edge cost function (Equation 4.6) we capture the fact that the closer (i.e., more semantically related) two nodes are, the lower the cost to modify one into the other is.

An exact solution to the GED problem can be found with a tree search over all possible edit operations, which, however, is computationally intractable for any reasonably-sized graph. In this work, we accordingly adapt an approximation method based on bipartite graph matching for finding the minimal edit cost (Riesen and Bunke, 2009). This precomputes the cheapest node modification costs for each node pair first, and stores them into a cost matrix. Since in our case there can exist multiple paths between two nodes (and, thus, multiple such modification costs), we always select the single cheapest node modification operation as the cheapest connecting path.[2] Next, the matrix is extended with the cost for node insertion and deletion – which we define as equal to the most expensive node modification operation in the matrix (see below for details). Computing the GED is now a bipartite graph matching problem between the source nodes of the two graphs, with the objective of minimizing the edit cost and subject to the restriction of a strict one-to-one matching (as every node can only be modified exactly once). We solve this minimization problem using the Hungarian method (Kuhn, 1955) – also known as Kuhn-Munkres or Munkres' algorithm. After computing the GED, we apply a simple normalization step to eliminate the effect of different graph, i.e., document sizes.

We summarize our approach in Algorithm 1. Given two semantic graphs G^i and G^j, representing documents d^i and d^j (Section 5.2.1), we perform the following steps:

i) **lines 1–9**: for each pair of source nodes $V_s^i \times V_s^j$ we find the cheapest undirected path $p^{i,j}$ with cost $c^{i,j}$ using Dijkstra's algorithm (edges along the path are weighted by one of our three measures from Section 4.2.2).[3] In the example in Figure 5.2, for instance, we compute the cheapest path between `db:Bob_Dylan` and `db:Johnny_Cash` from Doc A, and each of `db:Desire_(Bob_Dylan_album)` and `db:Folk_music` from Doc B in turn. The highest weighted edge, here `dbo:artist`, is assigned a cost of 0.7 (assuming that in this example we would have computed a global upper edge cost limit of $w_{max} = 6.0$ before, cf. Equation 4.6), whereas the lowest weighted edge, namely the two `rdf:type` relations, are both

[2] We observed that using only the single cheapest path instead of top-k paths results in superior performance, cf. Section 4.3.5 Effect of Top-k Paths.

[3] We run Dijkstra's algorithm (cf. e.g. Cormen, 2009, p. 658) to solve the single-source shortest-paths problem on our weighted, directed graph. Even though we have to run Dijkstra's algorithm multiple times, i.e. for each input node (of one document), we choose this option over computing all pairwise shortest path (using the Floyd–Warshall algorithm) because we need only the distances between the source/input nodes, and not the distances between any of the many intermediate nodes/entities. And given that the run-time complexity of Dijkstra's algorithm is much better ($O(|E| + |V| \log |V|)$ compared to $O(|V|^3)$) it is likely that even with the multiplication factor for the number of input nodes, Dijkstra's algorithm will still be the better choice, in particular for the larger graphs of 2 and 3 hops, where the input-nodes-to-vertices ration decreases drastically.

Algorithm 1 Graph-based semantic similarity

Input: Document DBpedia subgraphs $G^i = (V^i, E^i)$, $G^j = (V^j, E^j)$
Parameter: Maximal path length n_{max}

1: **function** SUBGRAPHDISTANCE(G^i, G^j)
2: $P \leftarrow \emptyset$ ▷ set of cheapest paths
3: **for all** $(v^i, v^j) \in V_s^i \times V_s^j$ from G^i, G^j **do**
4: **if** $v^i = v^j$ **then**
5: $c^{i,j} \leftarrow 0$
6: **else**
7: $c^{i,j} \leftarrow$ *DijkstraCheapestPath*(v^i, v^j)
8: $P \leftarrow P \cup \{(p^{i,j}, c^{i,j})\}$
9: $c^{max} \leftarrow \max_{p \in P_{length} \leq n_{max}}(c_p)$
10: **for all** $(p^{i,j}, c^{i,j}) \in P$ **do**
11: **if** $p^{i,j}_{length} \leq n_{max}$ **then**
12: $c^{i,j} \leftarrow c^{i,j}/c^{max}$
13: **else**
14: $c^{i,j} \leftarrow 1$
15: $D_m \leftarrow \{d_{i,j}\}_{i=1,\ldots,m,\ j=1,\ldots,m}, m = \max(|V_s^i|, |V_s^j|)$ ▷ edit cost matrix
16: **for all** $d_{i,j}$ **do**
17: **if** $i \leq j$ **then** ▷ be $i \geq j$
18: $d_{i,j} \leftarrow c^{i,j}$
19: **else**
20: $d_{i,j} \leftarrow c^{max}$
21: $M \leftarrow$ *HungarianCheapestMatching*(D_m)
22: $dist(G^i, G^j) \leftarrow (\sum_{m \in M} m_{cost})/|V_s^i \cup V_s^j|$
return $dist(G^i, G^j)$

assigned a cost of $6.0 - 1.5 = 4.5$. Given these costs, the cheapest path between, for instance, `db:Johnny_Cash` and `db:Folk_music` is the one through `db:American_folk_music`. Note that, in order to avoid long paths between very distant (and thus semantically unrelated) concepts, we limit the search based on a maximum search depth parameter n_{max}.

ii) **lines 10–14**: we next compute the node modification costs for each pair of source nodes. For paths found exceeding the path limit n_{max}, we set their cost to that of the most expensive path c^{max} found within the input graph pair. Since it might not be the case that both graphs are fully connected, we also set c^{max} as the cost for unconnected source node pairs. Finally, we normalize all cost values.

iii) **lines 15–20**: we build the final edit distance matrix D_m from the previously computed modification costs, as well as the costs of the node insertion and deletion operations, which we set to c^{max}. This is to account for the fact that, given an arbitrary document pair, the cardinality

of their sets of entities does not need to be the same: in this case, additional nodes are treated the same as very distant ones.

iv) **lines 21-22**: the edit distance matrix D_m represents a bipartite matching problem, which we solve with the Hungarian method. It finds the optimal, cost-minimal assignment in our node operations matrix, while ensuring that each node will only be edited once. We finally normalize the graph edit distance costs to account for the number of source entities in the two input documents.

As a result of the execution of the algorithm, the normalized graph edit distance between G^i and G^j is returned. In our example, we will get a mapping from `db:Bob_Dylan` to `db:Desire_(Bob_Dylan_album)` (cost 1.3) and from `db:Johnny_Cash` to `db:Folk_music` (cost 0.9 + 0.8). The final similarity score is then given by the sum of these edit costs (3.0), normalized by the number of distinct source entities in both documents (6).

5.3 Evaluation

In this section, we evaluate our idea to model documents as weighted KB graphs with a benchmarking dataset for semantic document similarity. To this end, we use the 50 documents dataset from Lee et al. (2005) (LP50) which is widely-used for evaluating semantic document similarity and thus enables us to compare our method against other state-of-the-art systems.

Note that we do not use the recent SemEval Semantic Text Similarity (STS) task Agirre et al. (2013) data for evaluation, since it focuses on very short texts, i.e., mostly sentences, which provide a too small context for our approach. Similarly, we do not evaluate on the text similarity datasets from Tsatsaronis et al. (2010) since they mostly consist of short texts with few entities. This setting is far different from our main goal, namely modeling entity-rich texts as graphs of KB entities and computing their semantic relatedness..

5.3.1 Experimental Setting

The LP50 dataset is a collection of 50 news articles from the Australian Broadcasting Corporation (ABC), which were pairwise annotated with similarity rating on a 5-points scale ranging from 1 (very different) to 5 (very similar) by 8 to 12 different human annotators. To obtain the final similarity judgments, Lee et al. averaged for each pair the scores of all annotators: however, the final collection of 1,225 relatedness scores has only 67 distinct values. Consequently, Spearman's rank correlation is not appropriate to evaluate performance on this data and we opt instead, following previous work like Gabrilovich and Markovitch (2007), for Pearson's linear correlation coefficient (r).

We report our performance figures on the LP50 dataset in Table 5.1, where we show the Pearson correlation coefficient r between the human-created gold standard and our graph-based approach (GED). In order to evaluate our method across different entity linking systems we test with both DBpedia Spotlight (Mendes et al., 2011) and TagMe (Ferragina and Scaiella, 2012), two state-of-the-art systems according to the comparative evaluation by (Cornolti et al., 2013). For each

Table 5.1: Results on the LP50 dataset (Pearson r correlation coefficient, best results are bolded).

TagMe	Jaccard			0.51
		jointIC	combIC	IC+PMI
	GED max depth @ 2	0.55	0.59	0.57
	GED max depth @ 3	0.52	0.56	0.54
	GED max depth @ 4	0.46	0.49	0.52
Spotlight	Jaccard			0.54
		jointIC	combIC	IC+PMI
	GED max depth @ 2	**0.60**	**0.63**	**0.63**
	GED max depth @ 3	0.55	0.61	0.61
	GED max depth @ 4	0.52	0.55	0.57
	DKPro (Bär et al., 2012)			0.21
	TakeLab (Šarić et al., 2012)			0.08
	Cosine BoW baseline			0.56

entity tagger, we compute the performance for predicting semantic document similarity with respect to different values for the maximum depth of the path search in the cost computation ($n_{\max}$). We compare our GED-based method with a variety of baselines:

i) a semantically-informed baseline which computes the Jaccard similarity coefficient over the set of entities identified within the input documents, namely $sim(d_1, d_2) = \frac{E_1 \cap C_2}{E_1 \cup E_2}$, where E_1 and E_2 represent the set of concepts identified by the entity tagger (i.e., TagMe or Spotlight) within documents d_1 and d_2, respectively;

ii) an unsupervised baseline computed as the cosine distance of a standard bag-of-words Vector Space Model;

iii) two strong supervised baselines based on two publicly available supervised systems, namely DKPro (Bär et al., 2012) and TakeLab (Šarić et al., 2012), both trained on standard SemEval semantic textual similarity (STS) datasets.

5.3.2 Results

Table 5.1 shows that using our graph-based approach to semantic document similarity we are able to beat all baselines by a large margin, achieving a correlation coefficient of up to 0.63 ($n_{\max} = 2$, using Spotlight and either combIC or IC+PMI weighting). This is equal to a relative improvement of 16.0% over the semantically-informed Jaccard baseline and 11.6% over the cosine bag-of-words baseline. All differences in performance are, unless noted otherwise, statistically significant at $p < 0.05$ using Fisher's Z-value transformation. The results indicate that our method is able to always perform above the Jaccard baseline for $n_{\max} \leq 3$, and achieves

Table 5.2: System comparison on the LP50 data (as reported by authors).

	r
GED-based (weighted)	0.63
GED-based (unweighted)	0.61
Bag-of-Words (Lee et al., 2005)	0.1-0.5
LSA (Lee et al., 2005)	0.60
ESA – original (Gabrilovich and Markovitch, 2007)	0.72
ESA – reimplemented (Bär et al., 2011)	0.46-0.59
ConceptGraphSim (Ni et al., 2016)	0.745
Learned Concepts (Huang et al., 2012)	0.808

the best performance for $n_{\max} = 2$. These parameter values are indeed in-line with the optimal ones found by previous research contributions making use of graphs derived from Wikipedia or DBpedia Navigli and Ponzetto (2012); Hulpus et al. (2013), which also showed the benefits of mining information from short, highly specific paths. Interestingly, this makes our model virtually parameter-free, because it implies that we can simply set the only tunable parameter of our method, the depth of the search used for entity matching, to a standard values (i.e., 2 or 3) which are known to yield good performance across many different tasks. When looking at the performance of the different weighting measures, we see that we consistently obtain the best results using either combIC or IC+PMI, which corroborates our findings on entity ranking (Section 4.3).

Finally, we notice that the different baselines show large performance variations. The simple cosine baseline turns out to be a difficult competitor – e.g., outperforming the simple Jaccard baselines computed from both TagMe and Spotlight annotations – which indicates that semantifying the input texts and applying a simple entity overlap measure is not enough to yield a robust performance. The supervised baselines, DKPro and TakeLab, both show instead an extremely low performance rate, although they were reported as being among the top systems of the SemEval STS 2012 shared task. This is because both systems are supervised in nature, and thus able to yield accurate performance only when in-domain labeled data are available.

Next, in order to better understand the performance of our method, we compare it in Table 5.2 with an unweighted version that does not use edge weighting (i.e., all edge modifications have the same cost), as well as previous results from the literature. When computing semantic distances without weighting i.e., using the Hungarian method for mapping, but applied to unweighted paths only, we achieve up to $r = 0.61$ when using Spotlight and a maximum depth of 3 – 12.5% above the semantically-informed Jaccard baseline and 8.3% over the cosine bag-of-words baseline. This indicates the overall robustness of our GED method, which exploits high-quality semantic paths from DBpedia. Similar to our results on entity ranking, additional performance gains can be achieved thanks to weighting semantic relations.

5.3.3 State-of-the-Art Comparison

When comparing our approach to the state-of-the-art systems on this dataset we see that we outperform well-established methods such as latent semantic analysis (LSA) (Deerwester et al., 1990) and strong baselines, but are nevertheless not able to achieve a performance as high as that of explicit semantic analysis (ESA) by Gabrilovich and Markovitch (2007) or Learned Concepts by Hulpus et al. (2013). However, note that our method has clear advantages over ESA as it provides a fully unsupervised approach that practically requires no tuning, and that thus can be applied to arbitrary data and domains with virtually no changes.[4]

The ConceptGraphSim system by Ni et al. (2016) builds upon our method as described here, as they also represents documents as weighted DBpedia subgraphs, however, they also include a node weighting for capturing the importance of an entity w.r.t. the original text it represents, cf. the related work in Section 5.4.3 for details. Thanks more advanced weighing and features, Ni et al. are able to achieve a performance of $r = 0.745$ on the LP50 dataset.

The best performing system for the LP50 document similarity task is the Learned Concepts approach from Huang et al. (2012), who report an – as of May 2016 unbeaten – performance of $r = 0.808$; cf. the related work in Section 5.4.3 for details.

In summary, we take these figures to be promising in that our approach to document semantic similarity, while being based on a general document model with many potential applications – e.g. ranking related entities (Chapter 4) and entity disambiguation for linking (Chapter 6) – is nevertheless able to come close to a highly specialized method like ESA, which has been tuned for this specific task and dataset.

5.3.4 Error Analysis

In order to gain additional insights into the performance of our method, we performed an error analysis of its output. To this end, we focused on the manual analysis of documents deemed closest or most distant from the human judgments. When looking at specific document pairs, we found that our knowledge-rich approach is able to estimate well the similarity between documents with little or partial word overlap: connecting paths between DBpedia entities, in fact, were found to implicitly cover a wide range of topical associations, ranging from near-synonymity ("U.S. intelligence" and "CIA") all the way to metonymic[5] relations ("White House" and "Bush administration"):

- "U.S. intelligence cannot say conclusively that Hussein has weapons of mass destruction, an information gap that is complicating White House efforts to build support for an attack

[4] On a side note, we want to highlight that the original performance figures for ESA (Gabrilovich and Markovitch, 2007) have been criticized (cf. Bär et al., 2011) for being based on a cut-off value used to prune the vectors, thus being over-fitted to the LP50 data – cf. also the much lower performance obtained by re-implementations of ESA including those from Bär et al. (2011); Hassan and Mihalcea (2011); Yeh et al. (2009).

[5] A figure of speech consisting of the use of the name of one thing for that of another of which it is [..] associated (definition from the Merriam-Webster Dictionary)

on Saddam's Iraqi regime. The CIA has advised top administration officials to assume that Iraq has some weapons of mass destruction. But the agency has not given President Bush a "smoking gun," according to U.S. intelligence and administration officials."

- "The Bush administration has drawn up plans to escalate the war of words against Iraq, with new campaigns to step up pressure on Baghdad and rally world opinion behind the US drive to oust President Saddam Hussein. This week, the State Department will begin mobilising Iraqis from across North America, Europe and the Arab world, training them to appear on talk shows, write opinion articles and give speeches on reasons to end President Saddam's rule."

However, since it relies only on DBpedia entities and their document mentions, our approach will perform badly in cases where i) the input documents contain few or no entities, or ii) they share the same entities, but describe different events. For instance, our method will give a very high similarity score to the following two sentences, although they describe completely different events:

- "Obama started his second term in the White House; [...]"
- "Obama will soon leave the White House."

But while our approach could be extended to include relations between entities which are automatically extracted from text, cf. recent work on building event graphs from documents (Glavaš and Šnajder, 2013), our results seem also to suggest that in the case of text similarity we can often get away without a deep analysis of the documents' sentences, since entity overlap is a good proxy for topical affinity. This is highlighted by the following two sentences from the LP50 data, which, albeit very different, belong to documents which were deemed highly similar by the annotators:

- "Nigerian President Olusegun Obasanjo said he will weep if a single mother sentenced to death by stoning for having a child out of wedlock is killed, but added he has faith the court system will overturn her sentence."
- "An Islamic high court in northern Nigeria rejected an appeal today by a single mother sentenced to be stoned to death for having sex out of wedlock."

5.4 Related Work

This work was, at the time of publication and to the best of our knowledge, the first to exploit a wide-coverage KG for modeling documents as graphs and computing semantic similarity based on this representation. The same idea was later adapted and refined by others, e.g. Ni et al. (2016); Paul et al. (2016).

Because an overview about semantic relatedness of words and entities was already given above in Section 4.4, this part will focus on (i) knowledge-based text representation, (ii) semantic document similarity in general, and (iii) knowledge-based semantic document similarity in particular.

5.4.1 Knowledge-based Text Representation

Booth et al. (2009) try to translate a natural language query/sentence into a database query, thus providing an natural language interface. For that purpose, they represent the natural language text as a semantic network, similar in spirit to our work, but they use WordNet concepts to represent words for understanding the query, in contrast to the entity-centric approach we take for modeling documents.

From a general perspective, our work can be viewed as building upon seminal research work in IR that explored the use of controlled vocabularies Lancaster (1972), originally introduced for library systems. The proposed method can thus be seen as instance of an advanced Knowledge Organization System (KOS) (cf. e.g. Eckert, 2012), since it relies at its core on a wide-coverage ontology to represent documents. However, as opposed to these approaches, we do not create a controlled vocabulary for a specific document collection, but instead reuse an existing, background ontology which contains general world knowledge. We use this knowledge source to represent the entities found documents, as opposed to using the documents' headings or metadata. The Jaccard similarity we report in Section 5.3 consists, in fact, of a baseline method that uses DBpedia as controlled vocabulary: we build upon this intuition and extend it by using the information encoded within the structure of the DBpedia network.

The idea of graph-based representations using DBpedia have been explored before by Hulpus et al. (2013), who aim at finding meaning full labels for a topic model; Given the words of each topic (a topic is a distribution of words from the input document corpora), they apply entity linking and then exploit the DBpedia graph to infer a meaningful entity as label for each topic. The basic assumption is that the entities of topic are somehow connected with each other – which is essentially the same idea we follow when representing a document as a DBpedia subgraph. Our approach is thus very similar in a way, but we go one step back and take the entities from the documents directly, thus creating a representation for each document within the DBpedia KG. Hulpus et al. also use unsupervised methods for graph exploration, e.g. inverse path length or random graph walks, but obviously with a different aim as the coherence of one subgraph is in focus for them, while we try to understand the matching between two subgraphs. In addition, they limit the graph construction to a set of manually selected DBpedia predicates, while we try to be agnostic including any relation and make the filtering via our edge weighting schema.[6]

Another closely related work is that of Scaiella et al. (2012), who use graph-based representations of snippets for Web search results clustering (cf. Chapter 3). Their method also builds a document-based semantic graph from Wikipedia entities, as obtained from an entity linker.

[6] As mentioned before, we reused (and extended) a list of DBpedia predicates provided by Hulpus et al. (2013) to filter out administrative predicates.

However, similarly to Shen et al. (2012), they do not exploit explicit semantic relations between entities (which we show to be beneficial for both entity ranking and semantic similarity).

5.4.2 Semantic Document Similarity

Before presenting other work on semantic document similarity that also builds upon KGs in the next section, we describe here other well-known methods for computing semantic document similarity – in particular those we compared our method against in the evaluation.

The two most-well known competitors, LSA and ESA, were already introduced in the previous chapter on entity relatedness (see Section 4.4.1) as they can be also used for computing semantic relatedness of words.

The DKPro system by Bär et al. (2012) was designed for participating in the SemEval 2012 task on STS and was the best performing system for two out of three metrics. The STS task (Agirre et al., 2012) is to compute the degree of semantic equivalence between a pair of sentences, it is thus very similar to our task of computing semantic similarity between documents. Thus, it is no surprise that the system makes use of different pairwise word similarity measures, ESA, and the similarities from a distributional thesaurus. These features are combined with a variety of string-based measures, including longest common substring, greedy string tiling, and the Jaccard coefficient on different character and word n-grams. More advanced features make use different existing systems for lexical substitution and statistical machine translation. Last, the system computes measures (often via the Jaccard coefficient) for the structural similarity of the sentence pair structural information by determining stopword n-grams, part-of-speech (POS) n-grams, or function word. In total, Bär et al. studied more than 300 different features and then trained a log-linear classifier for combining the 20 best performing feature, using a 10-fold cross validation on the training dataset provided by the STS task organizers. In our experiments, we used the pre-trained models from provided by Bär et al.. The rather low performance on the LP50 dataset seems to originate from the fact that the DKPro system was to heavily trained on the STS dataset. Our system takes in contrast a very different approach, as we neither build upon third party system, nor use a supervised method.

TakeLab by Šarić et al. (2012) was participating in the 2012 STS task, too, and is also a supervised system (using support vector regression) combining many different measures for capturing the similarity between the sentences. Features used include different n-gram and skip-ngram overlap features, word overlap, WordNet augmented word overlap, weighted n-gram overlap, but also vector space similarity, syntactic dependencies overlap, and finally named entity overlap and approximate numbers overlap. The system design follows thus the same spirit as Bär et al. (2012): Generated as many different features as possible and learn a feature combination from training data.[7]

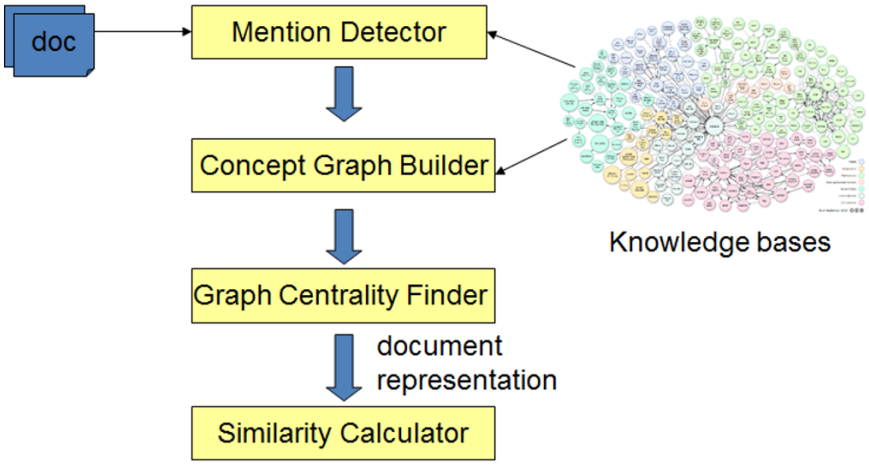

Figure 5.3: System architecture of Ni et al. (2016).

5.4.3 Knowledge-based Semantic Document Similarity

The work that is probably the closest to ours is from Ni et al. (2016), who explicitly build upon our work and can achieve a significantly higher performance on the LP50 dataset. They follow our processing pipeline by (i) taking a text document, (ii) extract entities via the TagMe entity linker, (iii) represent the document as a DBpedia subgraph of weighted entities, and then (vi) compute semantic document similarity based on this representation (cf. Figure 5.3) In contrast to our work, Ni et al. assign weights to nodes also based on their importance in the text – an aspect we did not consider as our edge weights are computed independently of the text – in addition to the entity relatedness weights both of us use (but with different implementations).

Comparing both approaches in more detail, we see that Ni et al. (2016) compute, like we do, a KG-based measure to estimate the relatedness of entities, as they build upon the same assumption that the set of entities representing a document should be closely related in the KG. However, the actual measure is more complex then our approach, as it incorporates three different graph-metrics: Degree, inverse of shortest path (like us), and the betweenness, i.e. the number of shortest paths between two nodes (which we did not find to be helpful). In contrast to our approach that is completely agnostic when weighting the predicate edges, Ni et al. (2016) introduce a special category association feature, that prunes the DBpedia-provides Wikipedia categories to construct a true taxonomy (a directed acyclic graph) first, and then computes an information content (IC)-based similarity measure from it. In addition to the DBpedia graph features, relatedness of entities is also computed via the well-established Wikipedia hyperlinks overlap (Milne and Witten, 2008a).

Another interesting feature that clearly extends our approach is the "content based weight-[ing]" of nodes: Entity-representing nodes get weighted according to the similarity between the entity and the source document they represent. The similarity is thereby computed as the cosine

[7] An approach similar in spirit to learning-to-rank (LTR) methods in IR.

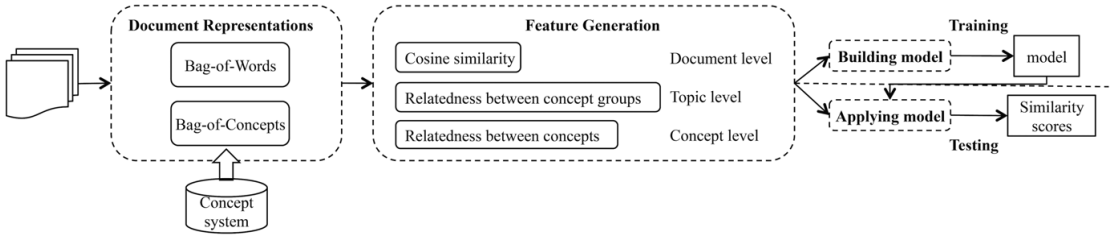

Figure 5.4: System architecture of Huang et al. (2012).

similarity of a bag-of-words vector space model of the Wikipedia article of the entity and the text document.[8] Taking all these features together, Ni et al. (2016) report a correlation of 0.745 for this ConceptGraphSim configuration, while the overall document similarity is the average of the best pair-wise entity similarity over all entities – which is somehow similar to our GED-based matching idea. When combination ConceptGraphSim linearly with ESA scores, performance increases to 0.786.

In summary, the work of Ni et al. (2016) follows the very same idea as our work, but they introduce several more DBpedia and non-DBpedia features. As a results, they have a much more divers and complex model, but also yield a much better performance.

Huang et al. (2012) achieved an average Pearson correlation of $r = 0.808$ on the LP50 dataset, which is the highest score were are aware of as of Mai 2016, thus outperforming our system as well as Ni et al. (2016). In their supervised approach as depicted in Figure 5.4, documents are represented in two different ways, as bag-of-words (BoW) and as bag-of-concepts – where concepts are either Wikipedia entities or WordNet synonym sets (synsets). However, the best performing model (where $r = 0.808$ for the LP50) does not make use of WordNet, but uses only Wikipedia entities instead. In total, this model consists of 17 different features acting as semantic similarity proxy, which the authors categorize into three groups:

1. Document level: this is essentially the cosine similarity of a *tf-idf* BoW model over the document text, plus the similarity between the Wikipedia entities added to each document. Thereby, for each document pair, the entities found within the other document are added and attached with a weight that is computed based on the relatedness of the entities form the other document with the entities from the document itself. The entity relatedness computation is based on existing work for exploiting the direct Wikipedia link structure (overlap in incoming/outgoing links). Those two features alone, called *CosineWords* and *EnrichedConcept*, are already able to achieve an performance of $r = 0.717$ – but notice how rich the information are that went into the *EnrichedConcept* feature.

2. Concept level: This group consists of 10 different features addressing the relatedness of the entities found within a document compared to the other entities. The motivation is

[8] We used the same technique in our work on relevance ranking of entities in Chapter 7 when computing the similarity between an entity and a query via the Wikipedia text (cf. feature WikiBoolean and WikiSDM in Section 7.2.6).

essentially to identify how central an entity is to the document that mentions it. In addition, each entity is compared against all other entities to identify which has the strongest overall relatedness to all concepts in a group.

3. Topic level: This contains 5 features that measure the size and relatedness of groups of entities. The groups are created by clustering together closely related entities, which those represented aspects or topics within a document.

For combining these features, the authors report on extensive experiments with different machine learning methods. The best results were finally achieved by learning regression (with 10 fold cross validation) to combine the different features using an support vector machine (SVM) with the radial basis function kernel (Smola and Schölkopf, 2004). In summary, the concept leaning approach from Huang et al. (2012) shows that the usage of KB entities and entity relatedness information can be a meaningful addition to simple BoW representations when computing semantic document similarity – which is in line with our own findings. The very high performance on the LP50 dataset indicates that the combination of many different features, and the usage of structural KB information together with standard BoW methods is more promising than our own one model approach.[9]

Another notable work sharing this chapter's ideas is by Paul et al. (2016), who also take a document, link it to DBpedia entities, and then explore the DBpedia KG to compute semantic document similarity. Interestingly, they address specifically the computational problems arising from such KG-based methods like their, ours and that from Ni et al. (2016) – and propose offline computation and indexing of shortest paths between entities as a solution. For estimating semantic document similarity, they propose a *traversal similarity* which utilized "spreading activation method: starting from a knowledge graph entity, it traverses semantic, non-hierarchical edges for a fixed number of steps". This is essentially very close to our graph exploration and path finding approach, however, unlike us and in line with e.g Ni et al. (2016), they tread the category and type predicates (call hierarchical edges) differently than the other predicates/edges. In the end, entity relatedness is computed as a combination of the graph-based relatedness within the hierarchical and within the non-hierarchical edge/predicates network. Document similarity is, similar to our GED-based method, computed as a graph matching problem, but Paul et al. allow also for 1:n mappings between document entities (instead of the 1:1 matching we enforce). The experimental evaluation on the LP50 dataset yields a Pearson correlation of $r = 0.712$ for the best system configuration, thus outperforming our numbers. Interestingly, Paul et al. also find, like us, that they achieve the highest performance when limiting the entity neighborhood exploiting to a size of two, i.e. two expansion hops in the DBpedia graph. In summary, Paul et al. (2016) proposes an approach rather similar to ours, in terms of document representation, entity relatedness, and document graph matching. However, they propose the interesting extension to separated the different types of relations (hierarchical vs. semantic predicates) when computed entity relatedness. It seems that this approach leads to the superior results, which would also be in line with the work and findings from Ni et al. (2016).

[9]However, one has to note here, that comparing our unsupervised graph matching against a supervised system combination 17 features using a 10-fold cross validation might also be questionable.

5.5 Conclusion

In this chapter, we proposed a novel method for document modeling that represents a document as a set of KB entities within the DBpedia knowledge graph (KG), thus as a KB subgraph (RQ1). Based on this model, we can view semantic document similarity as an approximate graph matching problem, using graph edit distance (GED) on the subgraph of entities and KG relations, and thus utilized the information about the relatedness of entities as encoded within the KG (RQ2). We evaluated our document model using an established dataset for semantic document similarity (50 documents dataset from Lee et al. (2005) (LP50)) and show that our approach outperforms baselines relying on traditional, i.e., 'flat', document representations, and also produces results close to those of well-known methods like explicit semantic analysis (ESA).

Our approach has the advantage of being a coherent model proposing a structured, computer- and human-readable representation for a document, and making it possible to compare documents with each other while integrating background knowledge into the process, but without the need for any supervision or extensive parameter-tuning. It is also (to the best of our knowledge), the first contribution towards making use of KGs like DBpedia for representing documents and computing semantic document similarity – an idea that was later extended and improved by other, in particular Ni et al. (2016) (cf. Section 5.4.3).

However, in terms of performance, our method seems to suffer from not integrating the KG information with the simple ("flat") bag-of-words (BoW) representation of the raw document, an approach other works found to be useful (e.g. Huang et al., 2012). Regarding the performance of the three different edge weighting schema, the experiments for document similarity are in line with the findings for the entity ranking task reported in Chapter 4.

In summary, we conclude that representing documents as subgraphs of general-purpose KGs is an interesting and promising idea to overcome the known limitations of purely surface form document representations. Subsequent research has shown that significant performance improvements are possible, however, a stronger integration with text-based representations seems to have the potential for even further improvements in the future.

Chapter 6

Entity Linking using the Knowledge Graph

In this chapter, we present another application for the entity relatedness method used above in Chapter 4 to rank entities, namely we approach the task of (a domain specific) entity linking. Thereby, the surface form mentions to be linked do not originate from an arbitrary natural language sentence or document (cf. Section 2.2), but we aim at linking the subject and object from surface form triples from OIE, here Nell, to disambiguated KB entities, here DBpedia. While the overall aim is to link a full OIE triple to a KB, here we only try to link the subject and object surface forms to their correct DBpedia entity, which is the reason why our KB-based entity relatedness computation from Section 4.2 is helpful: It discovers if (and how strongly) the subject-object-candidate-pairs are related according to the background KB.

The work presented in this chapter has been published before as: *Arnab Dutta and Michael Schuhmacher: Entity Linking for Open Information Extraction. In Proc. of NLDB'14, pages 75-80* (Dutta and Schuhmacher, 2014). The graph exploration method used for the entity linking described in that publication is the method devolved above in Chapter 4. This chapter will thus present an application of our method to a different problem.

Until now, we always worked with DBpedia as KB, which is extracted via manually created extraction rules from Wikipedia (as described in Section 2.1.3). One disadvantage of such KBs is their limited coverage – even though Wikipedia is a rather extensive encyclopedia, still many facts are not covered, and even less facts get extracted into the DBpedia KB. An alternative are KBs that got automatically generated from web text documents, using OIE systems like Nell or Reverb (Carlson et al., 2010; Etzioni et al., 2004). However, such systems, working on open domain, web-scale text data, and thus generating data of a rather wide coverage, suffer from their poor schema/ontology when comparing the generated facts against Wikipedia-based KBs (DBpedia, Yago). Thus, while being promising in terms of discovering novel facts on the web, the missing schema – which means neither entities nor relations/predicates are disambiguated, but just simple surface forms – imposes a serious limitation on such OIE approaches.

One idea to overcome this limitation is to combine both types of knowledge, those extracted from the web via OIE and those from structured KBs (cf. e.g. Dutta et al., 2013). In this chapter we present on the first step towards triple disambiguation, namely the task to link the subject and object of a Nell to their corresponding uniquely identifiable DBpedia entity instance.

We study how our KG pahts can help in this setting, and ask the following research question:

- RQ: Can knowledge graph paths be used to improve the disambiguation of surface form subject-object pairs?

6.1 Introduction

The task for this chapter was defined by Dutta et al. (2013): Given a subject-predicate-object triple as created from the open information extraction (OIE) system Nell, try to link the surface form of subject and object to disambiguated KB, i.e. DBpedia, entities. Note that linking subject and object from Nell to DBpedia is only a first step: We do not address the problem of mapping the predicates from Nell to their corresponding DBpedia properties, but focus on the entities here. The property matching was later addressed by Dutta et al. (2015).

Giving an example for our task, Nell might extract a predicate like this:

"bookwriter"("imperialism", "lenin").

where "imperialism" is the subject and "lenin" the object of the relation/predicate "bookwriter". Without any (human) background knowledge, it is difficult in this context to determine the correct entity for the subject and object terms: The surface form object "lenin" can refer to

(a) Vladimir Lenin (the Russian political theorist),

(b) Lenin (a nuclear icebreaker), or

(c) Lenin Prize

and maybe even more. But our example is talking about `db:Vladimir_Lenin`, because he is the writer of the book `db:Imperialism,_the_Highest_Stage_of_Capitalism` – a fact a human can recognize if s/he knows the book and its author. In this chapter, this very same process will be performed by our approach when finding entity relatedness via KB paths. For this example, DBpedia conveniently provides us with a direct relationship between subject and object:

```
db:Imperialism,_the_Highest_Stage_of_Capitalism
                dbo:author
          db:Vladimir_Lenin
```

From a very general perspective, the task given to us by Dutta et al. (2013) is an instance of EL (as introduced in Section 2.2), as the task is to match (link) the surface form mention of an

entity from a natural language text to its corresponding, disambiguated KB entity. However, in this chapter we will specifically link ambiguous Nell subjects and objects as found within a Nell triple – the fact that subject and object have been found to be in a specific relation as expressed in the source text of the Nell triple, makes this a very unique EL setting. Subject and object are plain surface form mentions, thus neither unique nor disambiguated; the disambiguation is thus the dominant problem there as illustrated in the Lenin example above.[1]

General purpose entity linking systems, like e.g. DBpedia Spotlight or Aida (Mendes et al., 2011; Hoffart et al., 2011), exploit, besides other features, also the textual context of the entity mention within the text. But in our case, this context information is not available or does not exist – we have only the triple itself available (Dutta et al., 2013). While having only triples is a limitation on the one hand side, the fact that subject and object are related (by some Nell predicate) also gives us an advantage on the other hand side, because we can try to exploit KB relations for finding the correct entities.

6.2 Method

In the following, we first present a strong baseline method, namely the frequency-based entity linking as used in Dutta et al. (2013). Second, we introduce our knowledge-based approach which exploits the DBpedia KB itself, following the KG exploration method introduced above in Chapter 4. Last, we study a combined approach, which incorporates the frequency-based and the graph-based approach.

6.2.1 Frequency-based Entity Linking

A simple, yet high performing approach for mapping a given surface form (Nell subject/objects in our case), to its corresponding DBpedia (or Wikipedia) entity is to link to its most frequent candidate entity Mihalcea and Csomai (2007). Even though this approach does not take any context information into account, it has proven to be effective not only for text entity linking, but also for Nell triple linking (Dutta et al., 2013). We thus use it as a baseline method.

For obtaining the frequencies, the Wikipedia hyperlinks and anchors were exploited (using WikiPrep (Gabrilovich and Markovitch, 2006)), as the link itself uniquely identifies the entities, while the anchor is a free text surface form. This approach is very effective and well used in most entity linking systems (cf. Section 2.2). Formally, if an anchor e refers to N Wikipedia articles $A_1, \ldots, A_N$ with $n_1, \ldots, n_N$ respective link counts, then the conditional probability P of e referring to A_j is given by, $P(A_j|e) = n_j / \sum_{i=1}^{N} n_i$. Thus, the pair (e, A_j), henceforth called subject/object-instance-mapping, is awarded the probability P. We rank the candidates on descending P and define a top ranked list as $E_{Subj|top-k}$ (for subject mappings) and $E_{Obj|top-k}$ (for object mappings). We select the DBpedia subject-object pair with the highest prior proba-

[1] Note that we have in principle also a NIL-linking problem, in case we have Nell entities that do not have a corresponding DBpedia entity. However, in our initial experiments we could not find this to be a significant problem, and thus do not address it in this work.

bility. Since every mapping of a subject is independent of the object mapping, we compute the prior probability as $P_{prior} = P_{Subj}P_{Obj}$.

6.2.2 Graph-based Entity Linking

As stated above, we assume that the subject and object connected by a Nell predicate are related to each other – like in the "bookwriter"("imperialism", "lenin") example from above – and that some/this relationship can be found within the DBpedia knowledge base. We thus compute the semantic relatedness between all subject-object-candidate pairs using the predicate-agnostic approach presented in Section 4.2:

1. We consider all combinations $E_{Subj|top-5} \times E_{Obj|top-5}$ and compute each pairwise cheapest path.

2. We weight the DBpedia graph edges by the best performing graph-weighting schema (CombIC, cf. Table 4.1 in Section 4.3).

3. We select the subject-object pair from $E_{Subj} \times E_{Obj}$ which has the minimal path cost on the weighted graph. The path cost between two entities is calculated as the sum of the edge costs along their undirected connecting path and is normalized as probabilities to P_{graph}

As result, we jointly disambiguate subject and object to their semantically most similar DBpedia candidate entities.

Last, we combine KB-based approach the frequency-based approach, as the latter can exploit the empirically obtained frequency data about common surface-form-to-entity mappings, while the former one is able to find relationships between subject and object in the background knowledge base DBpedia. We opt for a simple linear combination of the two approaches and select the subject-object combination with the highest combined probability

$$P_{comb} = \lambda P_{graph} + (1 - \lambda)P_{prior}$$

The influence of the λ parameter is evaluated in Section 6.3.2 (cf. Fig 6.1).

6.3 Evaluation

For evaluation we use a gold standard based on Nell triples provided by Dutta et al. (2013). It consists of if 12 different Nell predicates, like "bookwriter", "actorstarredinmovie", or "lakeinstate" – for the full list of predicates see Table 6.1 below. For each predicate, 100 triples like "bookwriter"("imperialism", "lenin") are provided. Subject and object have been manually linked to their correct DBpedia entities and our task is to perform this linking with the help of our method now. For more details on the dataset see Dutta et al. (2013).

Table 6.1: Performance scores of the three different methods on the NELL triple linking dataset (from Dutta et al., 2013). The best F_1 value per predicate is marked in bold. Bottom row shows the average F_1 gain over the Frequency baseline.

	Frequency-based			Graph-based			Combined		
	P	R	F_1	P	R	F_1	P	R	F_1
actorstarredinmovie	80.7	82.0	81.3	89.8	91.2	90.5	91.4	92.8	**92.1**
agentcollaborateswithagent	81.6	85.9	**83.7**	69.3	72.9	71.1	81.6	85.9	**83.7**
animalistypeofanimal	85.7	88.0	**86.8**	62.4	64.1	63.3	85.2	87.5	86.3
athleteledsportsteam	88.6	85.5	87.0	87.0	84.0	85.5	91.7	88.5	**90.1**
bankbankincountry	81.7	77.6	**79.6**	68.3	64.8	66.5	81.7	77.6	**79.6**
citylocatedinstate	79.0	79.4	79.2	81.5	81.9	81.7	86.0	86.4	**86.2**
bookwriter	82.2	83.1	82.6	83.8	84.7	84.2	87.6	88.5	**88.0**
personleadsorganization	83.6	79.0	81.2	78.4	74.0	76.1	84.8	80.1	**82.4**
teamplaysagainstteam	81.8	81.8	81.8	61.0	61.0	61.0	85.6	85.6	**85.6**
weaponmadeincountry	88.9	87.0	**87.9**	44.4	43.5	44.0	84.7	82.9	83.8
lakeinstate	90.3	93.0	91.6	84.7	86.6	85.6	91.5	93.6	**92.5**
Macro Average	84.0	83.8	83.9	73.7	73.5	73.6	86.5	86.3	**86.4**
Gain over baseline in %				-10.3	-10.3	-10.4	+2.7	+2.8	+2.9

6.3.1 Experimental Setting

We use Precision (P), Recall (R), and F_1-measure (F_1) as metric and evaluate each mapping individually, i.e. for each subject and for each object, in order to also take into account partially correct disambiguations/linkings where e.g. the subject was correctly linked, but the object not. From the given dataset we exclude all datum involving the Nell predicate "companyalsoknownas". This predicate, like for example in "companyalsoknownas" ("General Motors", "GM") describes different names/surface forms for the same real-world entity. Thus subject and object are then obviously not distinct entities, but different surfaces forms for the same entity. Such data cannot be handled by our approach and would need a different processing.

6.3.2 Results

We report the performance figures for each of the three approaches in Table 6.1: frequency-based, graph-based, and combined (equally-weighted linear combination of both approach). We find that – as to be expected – the baseline (most frequent entity) shows strong results with an F_1-measure of 83.9, while the graph-based method achieves only 73.6. Combining both methods yields an improvement over the baseline – which is notably a difficult competitor for unsupervised and knowledge-rich methods – of 2.9% w.r.t. the average F_1.

When analyzing our results in detail, we find that the combined approach improves the F_1-measure for all but two Nell predicates. The graph-based approach shows a larger performance variance in contrast, most likely because it cannot take into account any term frequencies information but relies solely on the, sometimes limited or incomplete, KB information. For example,

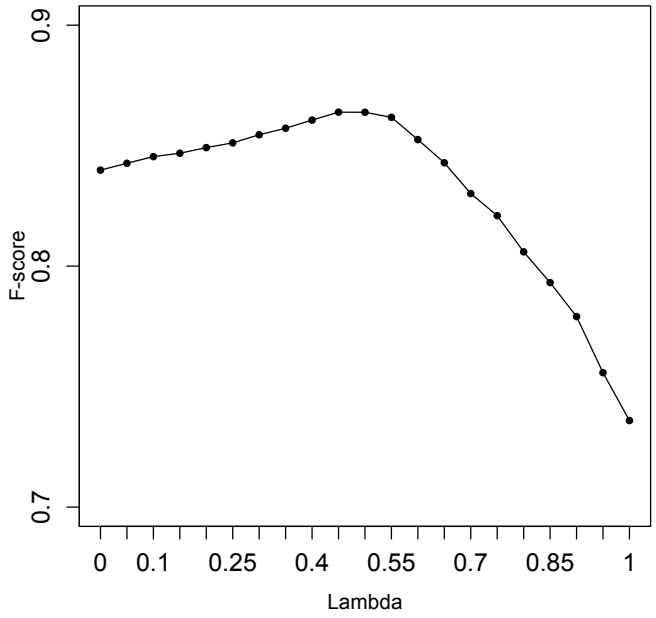

Figure 6.1: Effect of λ on the average F_1 score.

for the *actorstarredinmovie* predicate, F_1 increases from 81.3 to 90.5, but for *weaponmadeincountry*, it decreases by appr. 50%. This means that in the latter case the KB method very often selects highly related (from the perspective of the KG), but incorrect subject-object pairs.

Last, we report on the robustness of our combined approach with respect to the parameter λ, even though giving equal weight to both methods, thus setting λ to 0.5, seems to be a natural choice. Figure 6.1 shows the F_1-measure for $\lambda \in [0; 1]$. Note that $P_{joint} = P_{graph}$, when $\lambda = 1$ and $P_{joint} = P_{prior}$, when $\lambda = 0$. Confirming our initial choice, we observe a clear peak at $\lambda = 0.5$, with a clear performance decrease in either direction. Note that, while any supervised learning of the feature combinations would have not yielded a different solution here, this might be only a dataset specific characteristic. Nevertheless, it seems very plausible that (some kind of) combination of frequency-based and KG-based method will always yield superior performance.

6.3.3 Error Analysis

Taking a close look at the great variance in performances, we attribute the improvements to the fact that the underlying knowledge base had sufficient relatedness evidence favoring the likelihood of the correct candidate pairs. For example, for "actorstarredinmovie"("morgan freeman", "seven"), two possible candidate pairs (out of many others) with their probabilities are as fol-

lows:

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

With the most frequent entity method, we would have selected the former pair, given its higher prior probability of $P_{prior} = 0.227$. However, the graph-based method captures the relatedness, as DBpedia contains the directly connecting edge `dbo:starring` and thus selects correctly the later pair. In other cases, as observed often with "personleadsorganization" and "weapon-madeincountry", a low prior probability was complemented with a semantic relatedness, thus a high P_{graph}, thereby making a highly related, but incorrect subject-object-combination candidate more likely than the correct one. Consequently, the graph-based approach by itself lowers the performances, relative to the baseline.

The fact that the combined approach outperforms both the other approaches indicates that the linear combination of the two probabilities effectively yields in selecting the better of the two methods for each Nell triple. And this without processing the Nell predicate nor using any additional external supervision.

However, in addition to this effect, we observe that our combined approach also finds the correct mapping in cases where both, the frequency-based and the graph-based approach fail individually. Giving one example from the data, for the triple "teamplaysagainstteam"("hornets", "minnesota timberwolves"),[2] the frequency-based approach disambiguates it to the pair (`db:Hornet`, `db:Minnesota_Timberwolves`), which is incorrect, as `db:Hornet` is the entity of the insect hornet. But the graph-based approach also disambiguates wrongly to the pair (`db:Kalamazoo-_College`, `Minnesota_Timberwolves`), even though it discovers a very specific path in DBpedia between subject and object in this pair, via the intermediate entity `db:David_Kahn-_(sports_executive)`.[3]

The gold standard pair, (`db:New_Orleans_Pelicans`, `db:Minnesota_Timberwolves`), however, gets selected by the combined approach, which combines the medium high prior probability and a medium high relatedness originating from the fact that both instances are connected by `yago:YagoLegalActor`. Not that this last information originates from DBpedia and its unsupervised graph weighing method, not from the Nell predicate *teamplaysagainstteam*.

6.4 Related Work

We presented in this chapter an application of our entity relatedness measure to the task of linking NELL subject and object to DBpedia entities. While this was, on the one hand side, only the first step towards mapping complete NELL triples to DBpedia (as done later by Dutta

[2] "hornets" refers to `db:New_Orleans_Pelicans`, formerly known as the New Orleans Hornets.

[3] Even though the mapping is incorrect, one could argue that selecting a college in the US is a better choice than linking to an insect.

et al. (2015)), on the other hand side, it can be viewed as an domain/application-specific entity linking (EL) task, which we thus focus on our discuss of related work.

Note that the work most closely related to this chapter is from Hulpuş et al. (2015), who adapted our graph-based entity relatedness measure for a word sense disambiguation (WSD) task in the context of entity linking, and found it to have the best disambiguation capabilities. The details were already described in the previous chapter's discussion on related work, cf. Section 4.4.2.

Besides general purpose entity linking (EL), where an arbitrary text document can be the input and which is a well covered research area, see Chapter 2.2 (Entities), there exist also some work on domain-specific EL systems that gained attention. Domain-specific EL exists, because de facto all EL system contain some data-based supervision step, thus the training data has an influence on the performance of the system – which becomes a problem if a system was e.g. build for and from well-written news articles, but will then annotate informal and short twitter posts.

The first application is EL for web search queries, which is an important building block for modern web search engines, enabling them e.g. to retrieve entities like persons during the regular web search. According to Pound et al. (2010), entity queries constitute a significant proportion of web search queries,[4] see also our work in Chapter 7 (Relevance Ranking of Entities) and 8 (Finding Relevant Relations). Blanco et al. (2015) presented such an EL system with the focus on processing time and describe a system that can work in a real-world web search scenario where only some milliseconds of time are available for the EL step. Their model builds upon user-generated information from the web to link queries to KB entities. For the sake of time efficiency, in the entity disambiguation step relationships between entity candidates are ignored, and in addition advanced hashing and compression methods are used to reduce the memory footprint of contextual vectors obtained via distributional semantics.

A second well-known domain also dealing with short and underspecified, and often rather nosy, text is microblogging; which means very often EL for Twitter.[5] According to Guo et al. (2013), the main challenges originate from the fact that microblogs usually use short, noisy and informal texts with little context, and in addition often contain surface from phrases with ambiguous meanings. They also report that they find, in contrast to general EL, the mention detection to be the actual performance bottleneck, also because of the informal writing style and the many abbreviations used. Guo et al. (2013) report on experimental evaluation that shows that their domain-specific linker can outperform TagMe (considered to be a state-of-the-art general text EL system; cf. Section 2.2), by a large margin in terms of F_1 score.

Another system for the same task was proposed by Meij et al. (2012), who also created a publicly

[4] This usage scenario, to find entities instead of documents, was actually the initial motivation for web search engines to build their own large KBs, cf. Google's "Introducing the Knowledge Graph: things, not strings", `https://search.googleblog.com/2012/05/introducing-knowledge-graph-things-not.html`.

[5] Twitter (`http://twitter.com`) is probably the most well known micro-blogging services, traditionally allowing users to send messages (also microposts or microblog posts) of up to 140 characters. There are many more services like that, to mention just a few: Tumblr, Jaiku, Sina Weibo.

available benchmark dataset. Their method combines a variety of features, including different n-gram and different entity and knowledge base (KB) features, into a machine learning approach (random forest). In line with Guo et al. (2013), they report on challenges due to the informal writing style, and also that they are able to outperform TagMe as well as DBpedia Spotlight.

Many other domain-specific systems exist, all having in common that off-the-shelve EL system do not perform well enough for their specific application. Some thus proposed a combined approach like Olieman et al. (2015), who study the domain of conversations and find that superior results can be achieved by combing their own high-precision, relatively simple custom-made linker for conversations with a standard off-the-shelve EL system, thereby obtaining results with high precision and high recall. In the end, our work presented above is also just one of those domain-specific EL systems: We lack the context around the NELL triple, which makes the disambiguation harder, but have the advantage that we can exploit the subject–object relationship in this specific setting.

6.5 Conclusion

In this chapter, we studied another application for the entity relatedness metric introduced in Chapter 4, namely a domain-specific entity linking (EL) task, in which the ambiguous subject and object surface forms of an OIE system (Nell) had to be disambiguated to their corresponding DBpedia entities. Our initial assumption was, that in this specific setting, where the relation between subject and object has already been extracted from the NELL OIE system, information about the relatedness of subject and object should help in the disambiguation. We thus applied our method for computing semantic relatedness of entities as developed in Chapter 5 and by that provided another extrinsic evaluation of our KG-based entity relatedness method.

Using an existing benchmarking dataset from Dutta et al. (2013), we empirically showed that (i) the most frequent sense is a very strong baseline, but (ii) it can be improved by taking into account the semantic relatedness between subject and object as computed by our KG-based method. Even though we do not take into account the NELL predicate itself, in a way our approach mimics the human disambiguation process: What are the most likely entities for that surface form (most frequent sense baseline), and how strongly is the semantic relatedness between any subject–object combination (knowledge-based semantic relatedness). In contrast to other approaches, we presented a simple, unsupervised method that does not require any learning or parameter tuning and that achieved high overall performance – even without using the Nell predicate information. We conclude that this approach is able to overcome the lack of contextual information in context-free OIE triples by complementing it with existing background knowledge from the DBpedia KG, utilizing its power to measure entity relatedness. And we also understand these results as another hint that KG exploration is a helpful ingredient for problems where knowledge-free approaches reach, besides all their benefits, their natural limitations.

Part III

Using the Knowledge Graph for Relevance Ranking

Chapter 7

Relevance Ranking of Entities

So far in this thesis, we have focused on tasks that make use of the KG only as a means to an end, namely to text understanding, for tasks like text clustering (Chapter 3), semantic document comparison (Chapter 5), or entity disambiguation in NELL triple linking (Chapter 6). But the KG itself, in particular its entities, has not been the focus of our work. This will change with this chapter, which takes more of an IR perspective, and aims at ranking KB entities by relevance w.r.t. a user query.

The work presented in this chapter has been published before in *Michael Schuhmacher, Laura Dietz, and Simone Paolo Ponzetto: Ranking entities for web queries through text and knowledge. In Proceedings of CIKM'15, pages 1461–1470* (Schuhmacher et al., 2015).

Given a user's information need, expressed by a keyword query, one mean to fulfill the information need is to return entities as an answer, and not only documents, which leads to the task of entity retrieval. While there are different approaches towards entity retrieval (see Section 7.1), we focus in this chapter on a particular setting: Starting with Wikipedia entities extracted from query-relevant text documents, rank those entities by their relevance w.r.t. the initial keyword query (actually the information need).[1]

The research questions for this chapter are derived from the intention to explore, if, and how, entities extracted from document can be ranked to produce a satisfiable result list for the given retrieval query:

- RQ1: Do query-specific documents contain relevant entities?
- RQ2: What types of features (query-based, document-based, KB-based) improve relevance ranking of those entities?

[1] Which is why we address in this chapter not entity retrieval, but, more precisely, entity ranking (of the entities extracted from the retrieved text documents) as further described in Section 7.1.4.

Table 7.1: Different aspects of the entity retrieval task

Aspect	Alternative A	Alternative B	Section
Entity Source	Entity-linked documents	KB only (Wiki)	7.1.1
Entity Type	KB entities (Wiki)	Non-KB entities	7.1.2
Query Type	Entity/Type queries	Complex queries	7.1.3

7.1 Introduction

Information retrieval research and also commercial search engines show an increasing interested in going beyond words, in particular by integrating entities into the retrieval process. Thereby, retrieval systems, one the one hand, leverage (background) information about entities to improve the search result itself (Egozi et al., 2011; Dalton et al., 2014), on the other hand, entities are also used as an additional source of information to be presented to the user, as e.g. done by Google[2] and Yahoo[3]. We address here the latter type of entity integration, and aim at a scenario where documents and entities are return as a result set to the user. However, focusing on the new aspect of entity retrieval, we do not address the classical document retrieval part here, but take it as given assuming we already have this set of query-relevant documents and entities extracted from these documents.

The details of our specific entity retrieval setting will be described in the remaining part of this section. We discuss three different aspects relevant when studying entity retrieval, aspects are listed in Table 7.1, before giving the final task definition for our work in Section 7.1.4.

7.1.1 Types of Entity Sources

When displaying not only documents, but also entities as the result to a user query, the question which entities to be shown and in what order becomes relevant, as the number of entities an entity linking system (for an overview see Cornolti et al. (2013)) can extract from the documents retrieved is typically much higher than what a human user is able (or willing) to consume and understand. Thus, we want to understand how to determine the relevance of an entity w.r.t. the *query*, and thus w.r.t. the document result collection generated from the document retrieval system.

Note that our understanding of entity ranking, i.e. ranking entities that are extracted from a collection of documents, is rather different from the "classical" entity ranking task, especially in the context of the Initiative for the Evaluation of XML Retrieval (INEX) campaign (cf. Demartini et al., 2010; Gurajada et al., 2013). Methods developed for those initiatives aim primarily at retrieving entities directly from the knowledge base. We, in contrast, are interested in a setting where the entities are extracted from the relevant documents, and can thus act as an additional

[2]"Introducing the Knowledge Graph: things, not strings" from `http://googleblog.blogspot.de/2012/05/introducing-knowledge-graph-things-not.html`

[3]"The Y! Knowledge Base: Making Knowledge Reusable at Yahoo!" from `http://semtechbizsf2013.semanticweb.com/sessionPop.cfm?confid=70&proposalid=5187`

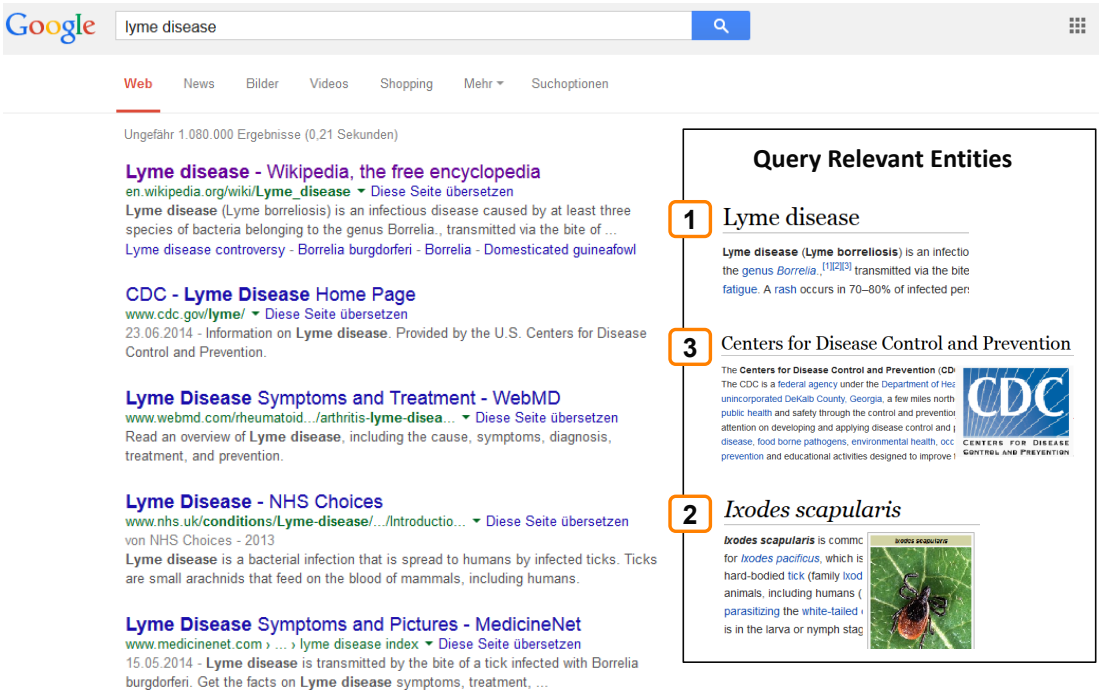

Figure 7.1: Illustrative example of a search interface showing documents and knowledge base entities.

mean to answer the given information need, while remaining the possibility to be integrated with the documents into some kind of advanced user interface (see e.g. Dietz et al., 2014; Hoffart et al., 2014) showing documents and entities combined. For illustration purposes, Figure 7.1 shows an example of such an interface with a ranked list of documents and entities.

7.1.2 Types of Entities

When aiming at extracting entities from query-relevant documents, the question what an entity is becomes relevant. Most of the recent work on entity retrieval, in particular also the INEX and TREC initiatives, have focused on entities from the controlled vocabulary of a KB, in nearly all cases Wikipedia or Freebase. This is also the entity definition we work with here.

However, one could also think of going beyond entities from a fixed vocabulary and consider also entities found only in the text. Early work on entity retrieval was actually working with a broader entity definition (see e.g. Balog et al., 2006), based on named entity recognition where any entity found in the text was a valid entity. But because of the increasing popularity and availability of approaches integrating with existing KB entities, most recent approaches focus only on KB entities. We stick here to this more strict definition and annotated the text documents (and queries) only with KB, i.e. Wikipedia (thus also a DBpedia) entities, illustrated by the intersection shown in Figure 7.2.

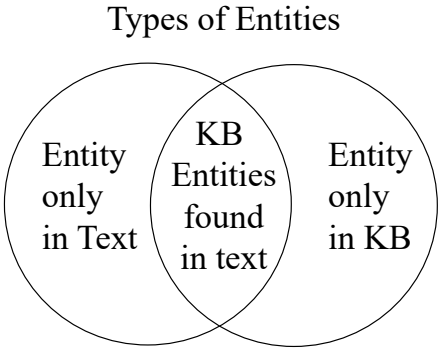

Figure 7.2: Types of entities: Non-KB vs. KB (very often Wikipedia) entities. The intersection contains KB entities found within text documents (via EL).

7.1.3 Types of Entity-related Queries

Our scenario addresses open-domain web queries, and in particular we want to be able to find relevant entities for more complex, general web queries, that do not ask specifically for a certain entity or a certain type of entries. Such a general query would be e.g. "marine wildlife", for which a user probably would like to see entities like the `National Marine Life Center`, `Marine biology`, `Whaling`, `Marine mammal`, or `Overfishing`.

For understanding what queries types exist in general, we look at a classification of types of web-search queries in the context of entity retrieval proposed by Pound et al. (2010), who also performed a query log analysis on real-world data. They define different query types, most importantly,

- entity queries that ask for a single entity ("CJ5 Jeep"),
- type queries that ask for a list of entities of a certain type ("cold medications"), and
- attribute queries that ask for the attribute of an entity ("zip code waterville maine").

From their analysis we know that 40.6% are entity queries, 12.1% are type queries, and 4.6% are attribute queries, thus leaving around 40% of all queries to address directly, but are what we consider a complex or general web query. We target in particular at such complex queries, because (a) they do not have a single entity as answer, but (b) require the retrieval system to present a selection of entities of different types, in contrast to a type query, found within the documents retrieved as relevant.

Note that our question of determining the relevance of an entity w.r.t. the *query* is not equivalent to determining the relevance of an entity w.r.t. its original source *document*. Examining the example query "marine wildlife" again and a relevant and retrieved document about "marine life studies", we would most likely find frequently the entity `bachelor of science`, which is however not very relevant for the query `marine wildlife`. Nevertheless, it can very likely

be relevant when looking at the document itself in isolation, because the specific document describes how to obtain a bachelor's degree in marine life studies.

7.1.4 Task Definition

Consequently, for the context of this chapter we define the task of entity ranking as follows. As a running example we use the query "Argentine British relation", as it is a general query that demands for different entities of different types.

- Given: a keyword query ("Argentine British relations").
- Provided: a collection of query-relevant documents that includes entity link annotations from mentions (e.g. "the conflict of the Falklands" to entities (e.g., `Falklands_War`) in a background knowledge base.
- Goal: Rank the entities by relevance w.r.t. the query.

We illustrate our problem by means of an example shown in Figure 7.3 (page 82). Initially, we are given the query "Argentine British relations" and the retrieved query-relevant documents D_q from the document collection. In our experimental evaluation of our method in Section 7.3, we used the TREC Robust04 and the TREC ClueWeb12 corpus as document pool, together with their corresponding set of queries. Next, an arbitrary document retrieval system is used to produce a ranked list of, ideally query-relevant, documents. From those documents, an arbitrary entity linking system (cf. Section 2.2) extracts entities and links the corresponding surfaces forms to their canonical Wikipedia name. This has here again the advantage of normalizing different surface forms to its common and disambiguated entity.

Obviously, the number of entities extracted from all documents, and already from a single document, is too high to be a reasonable result list for the user. Thus, at this point, after using document retrieval and entity linking, the question is what to do next – more specifically, how to rank the entities, such that the user gets shown only the most relevant entities. In this example, this would be the `Falklands_War` and the `Falkland_Islands_sovereignty_dispute`, as those entities describe one major aspect of the relations between the two countries. For that reason, both entities are annotated as highly-relevant (score 5) in this example. The annotation scores shown are actually take from the real gold standard REWQ Robust04 dataset we created (see Section 7.3.1).

7.2 Method

In this part, we develop our solution for the entity retrieval task, which is at its heart a feature-rich, supervised entity ranking based on several entity-related features. Our approach

- leverages heterogeneous features from unstructured (i.e., the documents' text) and structured (i.e., knowledge bases) knowledge source, and

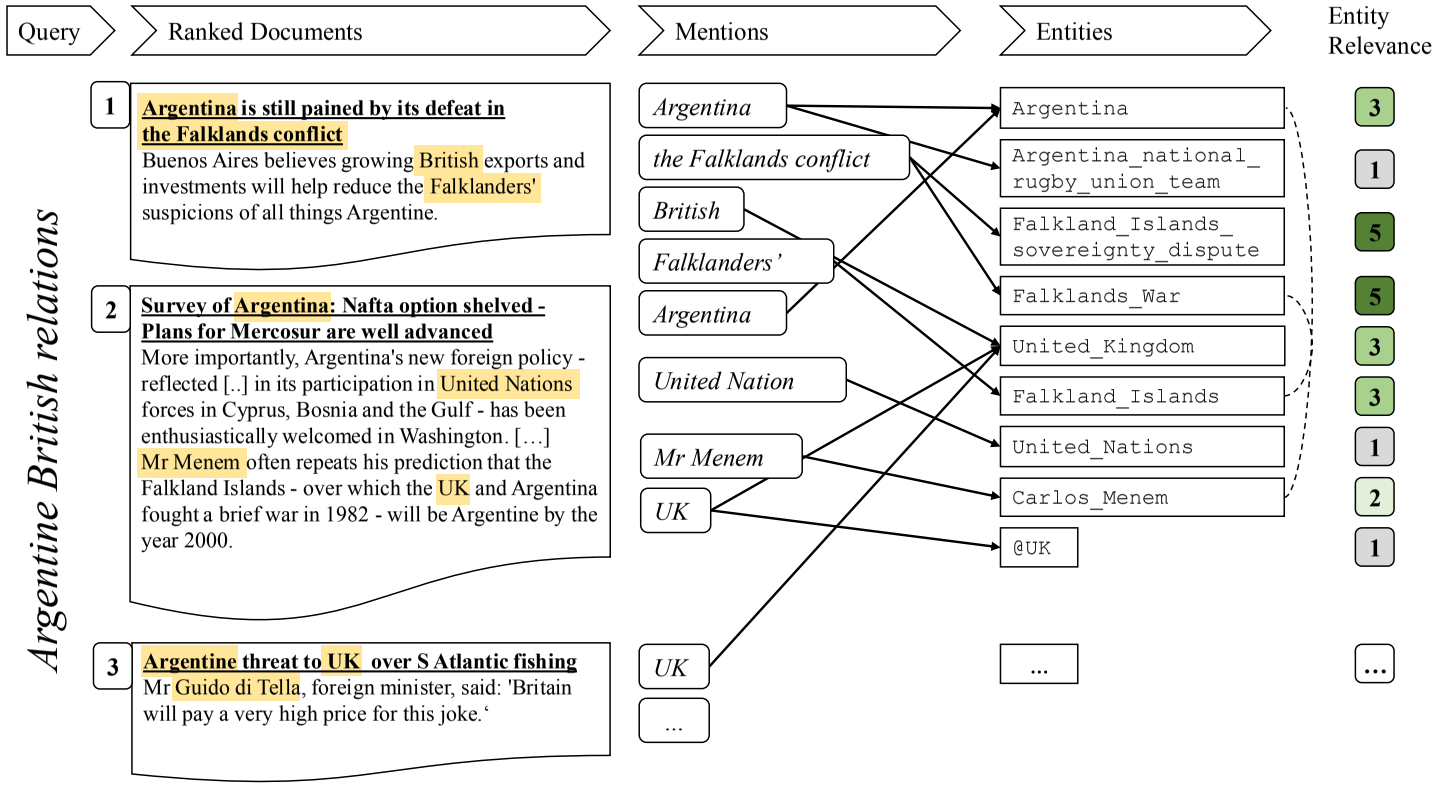

Figure 7.3: Example showing the entity retrieval flow from a given keyword query to a ranked list of documents as retrieved by a document retrieval system to a list of relevance-annotated (5-Likert scale with 1:not-relevant to 5:highly relevant) Wikipedia entities as extracted from the documents by an entity linker. Note that the entity `@UK` is an example of an linking error made by the linking system.

- combines them within a common supervised learning-to-rank approach.

In the following Section 7.2.1 we will give an intuitive explanation of the relationship between query, mentions, and entities we will exploit later, before describing the individual features in detail.

7.2.1 Method Overview

We show the potential benefits of our method using the example from Figure 7.3, where we are given the query "Argentine British relations": We assume that a list of query-relevant documents and entities extracted from those documents is already given; the details on this initial entity retrieval step will be described next in Section 7.2.2. Having a set of entities extracted from the text documents presents us thus an entity ranking problem, which can be formulated as the task of comparing the query with the document mentions and associated entities. We develop several different types of features for this comparison, as summarized by Table 7.2.

Initially, the first feature looks at the mentions in isolation, taking into account only mention frequency statistics independently of the query, as described in Section 7.2.4.

Second, our approach compares the mere surfaces forms of the entities within the documents with the query. This covers very similar terms like e.g. "Argentine" (query) and "Argentina" (mention) – whose associated entities are typically relevant for the query, cf. `Argentina` (relevance score 3) in our example. An extension of surface-level string comparison relies on distributional semantic methods, which allow us to compute term similarity on the basis of word co-occurrence information from very large corpora (Turney and Pantel, 2010). For instance, although a simple string similarity comparison between "UK" and "British" is not able to account for the 'nation-nationality' implicit relation between these two terms, this can be captured by their distributional vectors, which indicate that, in fact, they can occur in similar contexts. Query–Mention features are presented in Section 7.2.5.

Third, we compare the query *directly* with the entities. Working at the entity level makes it possible to leverage the information from knowledge bases of various kinds, including structured (here DBpedia), as well as semi-structured knowledge bases (here Wikipedia). In our example, `Carlos_Menem` is a relatively relevant entity (score 2), being a prominent former Argentinian president. We accordingly capture such relevance by (a) looking at the Wikipedia article of Mr. Menem – and find that he used to be the "President of Argentina" – on which we can apply surface-form-based approaches again; or (b) entity linking the query keywords themselves, resulting here in getting `Argentina` as one entity. We can then leverage the DBpedia knowledge graph to figure out that Carlos Menem is a citizen of Argentina, following the same idea of knowledge graph exploration as presented in Chapter 5. We describe query–entity features in Section 7.2.6.

Last, we leverage information about the relations between entities without using the query directly. Since all documents are retrieved w.r.t. the query, in fact, the frequently occurring entities within these documents should have something in common, i.e. they should be related to each

Table 7.2: Summary of the feature groups and their individual features, as computed for each (query, entity) pair; i.e. e.g. the Query–Entity feature *QEntEntSim* computes the relatedness between the given entity and any entity found within the query using the DBpedia graph for relatedness computation.

Feature groups	Feature description
Mention (Sec 7.2.4)	
MenFrqIdf	Idf-weighted frequency of the mentions per document
Query–Mention (Sec 7.2.5)	
SED	Levenshtein String Edit Distance
Glo	Similarity of the Global Vector from (Pennington et al., 2014)
Jo	Similarity of the distribution thesaurus from (Biemann and Riedl, 2013)
Query–Mention Ctx (Sec 7.2.5)	
C_SED, ...	(same as for Query–Mention, but for context window)
Query–Entity (Sec 7.2.6)	
QEnt	If query contains entity
QEntEntSim	Query to entity relatedness via DBpedia graph (cf. Chapter 4)
WikiBoolean	Query to entity relatedness via Wikipedia text (boolean retrieval)
WikiSDM	Query to entity relatedness via Wikipedia text (SDM retrieval from Dalton and Dietz (2013a))
Entity–Entity (Sec 7.2.7)	
SK (SVM only)	Semantic kernel capturing entity–entity relatedness (cf. Chapter 4)

other as they all satisfy (some aspects of) the query. Again, to account for such relatedness is possible by means of connecting relations found within the DBpedia knowledge graph (cf. Chapter 5). In our example, this holds for the entity pairs `Argentina` and `Carlos_Menem`, and `Falkland_Islands` and `Falklands_War`, which are both directly connected in DBpedia by a predicate. Entities that are not relevant for the query, e.g., `United_Nations`, instead, do not have knowledge graph predicates with other relevant entities. We captures this information by exploring the DBpedia graph and using a semantic kernel as explained in Section 7.2.7.

Before going into the details of each feature type (Section 7.2.4-7.2.7), we first briefly describe how we obtain the documents and entities (Section 7.2.2), and also introduce the two learning-to-rank methods used later to combine the entity features (Section 7.2.3).

7.2.2 Entity Candidate Retrieval

As state above, the initial document retrieval and the entity linking is not the focus of this work, but assumed to be done already. To analyze the impact of this step on the remainder of our approach, we study two retrieval models for two data sets: TREC Robust04 and ClueWeb12.

We start by issuing the query to a document retrieval system and collect the top results. For the Robust04 dataset, we use the document retrieval method EQFE from Dalton et al. (2014), which is an entity-aware document retrieval method. This system uses entity links within documents to

produce its document ranking. These entity links are created with KBBridge (Dalton and Dietz, 2013a), which we also use in our method. In the second experiment on ClueWeb12, we verify our findings by using a different, keyword-based retrieval method, the Sequential Dependency Model (SDM) (Metzler and Croft, 2005), and a different, existing set of entity links, the FACC1 dataset published by Gabrilovich et al. (2013).

In both cases, entity links in high ranked documents are used to build a pool of candidate entities. Consequently, our ranking problem is formulated as the task of comparing the query with the document mentions and associated entities. Note that we opt here for a realistic setting where a state-of-the-art entity linking system is used to disambiguate entity mentions in context: as a result of this, the approach suffers from entity linking errors, e.g. "UK" can be `United_Kingdom` but also `@UK`, namely a company.

7.2.3 Learning-to-rank

For our task, we train a supervised LTR model on labeled data, a method which has been shown to yield competitive performance for many retrieval tasks (Liu, 2011; Li, 2011). Each entity e to be ranked is represented by its feature vector $\mathbf{x}$, consisting of the features described in the next sections. The aim of any learning-to-rank method is than to find/learn a (often linear) retrieval function $h(\mathbf{x})$ such that the computed ranking scores produce the best possible ranking according to some evaluation or loss function. There exist three general types of learning-to-rank methods (Liu, 2011), namely

1. pointwise (optimizing on single entities),
2. pairwise (optimizing on pairs of entities), and
3. listwise (optimizing on the full list of entities)

approaches, whereof pairwise and listwise methods are most-commonly used.

To reduce the impact of the learning-to-rank method on your experiment results, we use two different methods: a ranking support vector machine (a pairwise method) and an greedy optimization using coordinate ascent (a listwise method). We briefly present both methods next.

Ranking SVM

The ranking SVM (SVM-rank) views the ranking problem as a pairwise classification task (Joachims, 2002b, 2006). For each pair of entity feature vectors $(\mathbf{x}_u, \mathbf{x}_v)$, it learns a retrieval function $h(\mathbf{x})$ from the labels y_u and y_v such that

$$h(\mathbf{x}_u) > h(\mathbf{x}_v) \Leftrightarrow y_u > y_v \tag{7.1}$$

holds for all pairs within the same query. Instead of maximizing (7.1) directly, SVM-rank minimizes the number of discordant pairs in Kendall's τ – namely, the number of pairs whose order

is different from the order in the ground truth. The ranking SVM learns a linear ranking function $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ that optimizes the following minimization problem:

$$\begin{aligned} \textit{min:} \quad & \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum \xi_{u,v}^{(q)} \\ \textit{s.t.:} \quad & \mathbf{w}^T(\mathbf{x}_u^{(q)} - \mathbf{x}_v^{(q)}) \geq 1 - \xi_{u,v}^{(q)} \\ & \xi_{u,v}^{(q)} \geq 0, \quad \forall q \in Q \end{aligned} \tag{7.2}$$

where $\mathbf{w}$ is a weight vector, $\xi_{u,v}^{(q)}$ the training error, and C is a parameter that allows trading-off margin size against error. Because the learned function $h(\mathbf{x})$ can always be represented as a linear combination of the feature vector, the ranking SVM can use also non-linear kernels (Joachims, 2002a). We use this to leverage a semantic kernel function that captures the relations between entities (Section 7.2.7).

Linear Model with Coordinate Ascent Optimization

As an alternative to the pairwise approach of the Ranking SVM, we also work with a linear feature-based model that optimized for the evaluation metric directly. The model builds upon a linear feature combination function $h(\mathbf{x})$, just like the pairwise approach, but for parameter estimation the real evaluation metric, e.g. MAP, is maximized directly – and not some other measure, like e.g. the number of disorders pairs as the Ranking SVM does. This direct maximization of an evaluation metric that can only be computed on the full ranking is the reason for naming such methods as listwise learning-to-rank approach – they optimize on the full list or ranked results.

Because the evaluation metric can only be computed on the full ranking, the optimization task is to find the parameter combination resulting in highest evaluation score. An intuitive solution to this task is a fine-grained discretization of the feature weights, which leads to a rather large grid of possible combinations – the number of possible combinations depends on the number of parameters and on how fine-grained we choose our grid. But because an exhaustive grid search becomes easily infeasible in a reasonable time, we use the coordinate ascent method instead. It is a local search technique that iteratively optimizes a multivariate objective function by solving a series of one-dimensional searches (Metzler and Croft, 2007). It repeatedly cycles through each parameter, holding all other parameters fixed, and optimizes over the free parameter. Because the convexity of the search space cannot be guaranteed, we perform multiple restarts to avoid ending up in a local minimum. In our experiments, we use the RankLib[4] implementation of the Coordinate Ascent method and MAP as training metric.

7.2.4 Mention Features

The first feature is based on the number of entity mentions in retrieved query-relevant documents. To this end, count statistics over all the targets of all entity links are collected (notice,

[4]Version 2.1-patched, `http://sourceforge.net/p/lemur/wiki/RankLib`

that some entity linkers retain multiple targets per link). While we study raw counts (MenFrq) for comparison, we use TF-IDF to weight mention counts across the document collection.

Mention Frequency (MenFrqIdf): The number of occurrences of each entity over all retrieved documents per query, weighted using TF-IDF as follows:

$$\text{MenFrqIdf}(e) = \text{tf}_q(e) \log \frac{N}{\text{df}(e)} \tag{7.3}$$

This feature is already a strong ranking method by itself (cf. Section 7.3), since query-relevant documents are likely to contain relevant entities. Note how this feature is similar to the document-based method for expert finding from Balog et al. (2006, Model 2), who also first retrieve query-relevant documents and then extract and rank the entities from those documents by frequency. Nevertheless, one shortcoming of this features is that the connection between the query and the entity is established only indirectly – i.e. only through the documents.

7.2.5 Query–Mention Features

We next design features focusing on the surface-level representation of the query and the documents, and compare the query keywords with the entity mentions from the documents. We define an entity mention as the sequence of words (or a single word) that the entity linking system linked to (one ore more) candidate entities in the knowledge base. Thus, mentions are surface form representations found in documents pointing to the knowledge base entities. In our example, this means we look at the mentions "UK" and "British" instead of the entity `United_Kingdom`. We apply different word similarity methods to compare the surface forms of each pair consisting of mention and query. Features capturing similarity scores from multiple pairs, because there are multiple entity mentions for the same entity, are then aggregated by averaging over all similarity scores for each entity.

Levenshtein (SED): In order to cover basic morpho-syntactic similarity, we compute the normalized Levenshtein String Edit Distance (Levenshtein, 1965) between the query and the mention (as one string).

Leaving beyond this simple syntax-level comparisons and going into semantic representations of words, we leverage existing distributional semantics models, namely GloVe and JoBimText. The general idea of these distributional thesauri is that they model words based on their global frequency co-occurrences in large text corpora. They can thus identify general semantic similarity between words, without the explicit need for a context – which we cannot provide in our case of the query words, which are by nature sometimes short and underspecified.

Glo: The global vectors (GloVe) method by Pennington et al. (2014) is a global regression model for unsupervised learning of word representations that builds vector space representations from word co-occurrences within a local context window, as previously proposed in work on the skip-gram model (Mikolov et al., 2013). The pre-trained model we employ is built from the Gigaword-5 corpus and the English Wikipedia, and contains the 400,000 most frequent words.

While it was primarily designed for word analogy tasks, it was also tested for word similarity, which is our use-case here. We compute the similarity score between query and mention as the cosine between their two word vectors. Because GloVe covers only single words, we tokenize query and mention, compute similarity for all pairwise token combinations, and aggregate these scores by taking the average, or the sum, as the overall similarity value.

Jo: JoBimText (Biemann and Riedl, 2013) is another distributional semantics approach. In this work we use its distributional thesaurus (called Jo(s)) that, similarly to GloVe, models words based on their frequency of co-occurrence in large text corpora. In contrast to Glove, JoBimText relies on text statistics obtained from grammatical dependencies, thus potentially providing a deeper representation to compute similarity. Grammatical dependencies provide context features for words, and word pair similarity can be accordingly computed as the number of shared features. In our setting, we again tokenize query and mention, compute all pairwise word similarities and aggregate the overall similarity scores into different features, as obtained by computing either the average or the sum over all word pairs built from the query and the entity mentions. Note that for this similarity metric $sim_{JoBim} \in [10, 1000]_{\mathbb{N}}$, due to cut-off thresholds in the model. However, for all experiments, we normalize all feature variables, as described in Section 7.3.

Mention Context: In many IR and NLP tasks, context helps to disambiguate and to obtain more precise rankings. We thus compute the above described similarity measures also for the 10 content words surrounding the entity mention. Context window of larger sizes had no noticeable impact on the system performance.

7.2.6 Query–Entity Features

This set of features compares the entities directly with the query. We achieve this in two different ways, by

1. applying entity linking to query keywords and compare the obtained query entities with the document entities (Section 7.2.6);

2. by collecting textual features from Wikipedia for the entity and compare it with the query words (Section 7.2.6).

For the first option, we leverage DBpedia as structured knowledge base, and for the second option Wikipedia as semi-structured textual knowledge base.

Comparing Query Entities

We first run the queries through an off-the-shelf entity linker, here TagMe (Ferragina and Scaiella, 2012), to collect entities found within the query. In case the entity linker returns more than one entity, we keep only those entities that show a linking confidence comparable to the linking with the highest confidence, thus ensuring a high precision for the entity linking. Even though being

an import step in the pipeline, we do not further improve this linking step as it was not the focus of this work.[5]

The availability of entities for both the query and the documents allows us to perform matching at the level of these disambiguated and unique entities. In the example from Figure 7.3, this is the case when comparing the entity `United_Kingdom` obtained from the query term "British" with that obtained from the document mention "UK". However, the real advantage of exploiting a background knowledge base comes from identifying relations between different, albeit related entities. For example `Carlos_Menem` is not mentioned in the query. However, as he used to be the president of Argentina, he is also to some extent relevant for the query. Accordingly, given the set of entities found in the query and its respective top-ranked documents from the retrieval systems, we build two features for each query-entity pair, namely:

- Direct Entity Match **QEnt**: binary feature whether the two entities match – i.e., both query and document mention refer to the same entity;
- Connected Entities **QEntEntSim**: query and document entities are not the same, but are connected in the DBpedia graph.

For finding interesting paths in the DBpedia graph, we make use of the entity relatedness approach from Chapter 4, where we performed a shortest path search over a weighted version of DBpedia. However, as we are here only interested in high-precision results, in contrast to the fuzzy graph matching problem for the semantic document similarity from Chapter 5, we consider only outgoing paths of length 1. For the same reason, we restrict the exploration of the DBpedia graph to

(a) direct relation predicates from the DBpedia OWL ontology, like e.g. `dbo:notableStudent`, `dbo:commander`, `dbo:knownFor`, or `dbo:routeEnd`, which capture the high-quality information in DBpedia, and

(b) the links to common Wikipedia categories (using the `dcterms:subject` predicate).

Effectively, this lets us retrieve direct KB predicates between two entities as well as common Wikipedia categories (as both entities point to the common category via `dcterms:subject`). In our example, we find a direct connection between `Carlos_Menem` and`Argentina` via the `dbo:nationality` predicate.[6] Note that with this approach, we remain agnostic towards the type of entity under consideration, and do not pick specific relations for specific entities, e.g. only looking at the `dbo:location` predicate when dealing with a geographic location.

Comparing Query Mentions

Although during prototyping manual inspection revealed that the query entity linking works usually sufficiently well, some queries can be (a) rather ambiguous and thus hard to interpret,

[5] Meij et al. (2009) presented a work specifically addressing entity linking for search queries.

[6] Note that there is no predicate expressing the being-president-of relationship directly here in DBpedia. This a good example of the principle shortcoming of KB-based approaches, namely limited coverage.

or (b) not be linkable at all for used entity linking system. Thus, we also compare the query keywords directly with textual features of the entities, as an alternative to entity linking. The textual features are retrieved from the comprehensive Wikipedia articles of each entity, which has a wide-coverage and high-quality information about entities – in particular when compared against the documents obtained from general web corpora. Given the query keywords, we apply two types of information retrieval models on the English Wikipedia, and accordingly build two kinds of different features.

WikiBoolean: all entities returned by a basic standard Boolean retrieval model, based on a full text index over all Wikipedia articles. We bind query keywords with disjunctive operators. This approach essentially tests if at least one query keyword is found within the Wikipedia article of the entity to be ranked.

WikiSDM: we further use a Galago[7] search index of an English Wikipedia dump. Using the sequential dependency model (Metzler and Croft, 2005) with collection-level Dirichlet Smoothing, we use the query to retrieve 1,000 Wikipedia articles. We use the retrieval score of the Wikipedia articles as a measure of relatedness for the entity.

7.2.7 Entity–Entity Features

In contrast to all previous features, which quantify the similarity between the entities and the query, this feature captures instead the degree of similarity between entities themselves. Consider, for instance, the two document entities `Falkland_Islands` (from mention "Falklanders"') and `Falklands_War` (from mention "the Falklands conflict") in the example in Figure 7.3. Even when not looking at the query, those two entities are obviously related to each other, in DBpedia we find that the `Falklands_War` took place (`dbo:place`) on the `Falkland_Islands`.

This relation between document entities is by itself interesting, because of the initial document retrieval: In an ideal world with perfect and full-coverage KBs, all query-relevant entities would be connected by some DBpedia/KB path, describing the explicit relationship between the entities, because the graph of entities would provide exactly the query-relevant information. This is essentially the same argumentation why we opt to represent a document as a graph of DBpedia entities in Chapter 5, but here be apply it to set of query-relevant documents instead of just a single document. In both situations, the underlying assumption is that topic coherent documents/sets of documents produce a set of entities that are closer connected to each other than unrelated entities.

In reality, however, we have of course only limited and incomplete KB relations. Thus, even the relevant entities are not fully connected by KB relations. Therefore, for this entity ranking problem we only look for direct, i.e. strong, relations between document entities and employ the same procedure as with the QEntEntSim feature from Section 7.2.6. The resulting feature is thus an indicator of a strong entity relations between document entities. The real strength of this feature becomes obvious for entities that have very different features (as computed w.r.t. the

[7] `http://lemurproject.org/galago.php`

query), but are nevertheless related to each other. This relatedness can only be capture by this feature, and not by the query–mention (Section 7.2.5) or query–entity (Section 7.2.6) features.

To include this feature into the ranking problem, we have to use a different approach than the direct query–entity features, or any linear feature combination of those, because this features is only defined for comparing entities with each other. For that reason, we opt to extend the ranking SVM, because it allows us to replace to use also other than the linear kernel (Joachims, 2002a). In particular, we do not have to limit the kernel to operate on the vectorial data, but can define a kernel function that includes this DBpedia graph information. While there has been work on kernels that operate on graphs or trees directly (see e.g. Gärtner, 2003; Moschitti, 2006), we choose to stay consistent with the procedure used for the *QEntEntSim* feature and pre-compute an entity-entity relatedness score. We then define our kernel as a mixture of a linear kernel for the standard features, combined with a semantic smoothing kernel for the entity–entity relatedness score.

The semantic smoothing kernel was originally proposed by Bloehdorn et al. (2006) to cover semantic relatedness (or proximity) between words: In a standard one-hot feature representation, each word is usually represented by a binary (or real-valued, e.g. tf-idf weighted) feature, thus there is one dimension per word.[8] For such a representation, a linear kernel $K(x_1, x_2)$ computes the inner product between two data items, here e.g. documents, $x_1 \cdot x_2$, which can however only be $\neq 0$ if both items have at least one dimension, i.e. word, in common. In case two words are semantically related but not identically, e.g. Pizza and Pasta, those words will not contribute to the inner product because they are encoded via different dimensions. The Semantic Smoothing Kernel tries to overcome this limitation by introducing an item similarity matrix Q that contains the semantic proximity between the dimensions, i.e. words of the input space. The semantic kernel is formally given by

$$K(x_1, x_2) = x_1^\intercal Q x_2 \tag{7.4}$$

(Bloehdorn et al., 2006, Def. 2) where the matrix Q can be used to encode proximity information about different dimensions/words into the kernel, thus the SVM.

In our work, we embed the DBpedia-graph-based relatedness between entities, computed by same method as the *QEntEntSim* feature, but using the real-valued score $s \in [0, 1]$ and not the binarized version from above, into the proximity matrix Q,[9] The entity data is encoded with a one-hot feature representation – in addition to the query–entity/mention features described above. This approach leads obviously to a much larger feature space (number of features + number of unique entities) as well as the need to look up the values from matrix Q (squared matrix of size $(\#ent)^2$) for each inner product computation. In summary, with our custom

[8] See e.g. `http://scikit-learn.org/0.16/modules/generated/sklearn.preprocessing.OneHotEncoder.html`

[9] Using the pairwise entity relatedness directly as Q is actually a technical violation of the definition of a kernel, as Mercer's theorem requires Q to be positive semi-definite (PSD) in order ensure that the function K can actually be used as kernel, which we cannot guarantee here. However, even though we cannot guarantee Q to be PSD in general, we find that the SVM learning converges on our datasets, supporting the observation from Burges (1998) that in practice even kernels without PSD guaranties can yield good results.

Table 7.3: Comparison of the key characteristics of both REWQ settings

	REWQ Robust04	REWQ ClueWeb12
Doc Collection	TREC Disk 4/5	TREC ClueWeb12
Queries/Topics	TREC Robust Track '04	TREC Web Track '13/14
Entity Linker	KB-Bridge	FACC1
Top-k Docs per Query	19	20
Top-k Entities per Query	Top 50	All, [50 - 488], avg 215
Relevance Judgments	Graded 1-5	Binary
Evaluation Metric	NDCG	NDCG, MAP

kernel we can incorporate the information that `Falklands_War` has a strong relation to the `Falkland_Islands` where it took place, without changing any of the query-related features directly.

7.3 Evaluation

For evaluation we designed two different, independent experimental setups, named after their base datasets Robust04 and ClueWeb12, both having different candidate retrieval settings, as described in Section 7.2.2, and different evaluation datasets, as described in the following Section 7.3.1 on the REWQ dataset.

7.3.1 REWQ Datasets

We opt to create our own dataset, the Ranking Entities for Web Queries (REWQ) dataset, as there exists no ground-truth-annotated dataset specifically addressing our needs, i.e. given a general web query and query-relevant documents, what are relevant entities from those documents. Both REWQ datasets build upon established document collections and queries for web information retrieval from the TREC evaluation campaigns, namely Robust04 and ClueWeb12. The entity relevance annotations created by us are available at `http://rewq.dwslab.de`. Details of both settings are described next, Table 7.3 gives already a brief overview.

REWQ Robust04 Dataset

For Robust04 setting, we builds upon the state-of-the-art ad-hoc document retrieval system from Dalton et al. (2014), which is already using entities itself for the document retrieval within its entity query feature expansion (EQFE) technique. The entities were extracted from the corpus documents with the KB-Bridge entity linker (Dalton and Dietz, 2013a), we re-use these entities for our purpose.

As document corpus we use the TREC Robust 2004 data set (Voorhees and Harman, 2005), because we aim at covering complex web queries, which are provided with this established dataset. In order to study the interplay between document and entity retrieval, we start with

those queries where the ad-hoc retrieval has proven to be successful, as those document results collections should have a high chance of including query relevant entities. Accordingly, we select the 25 top-performing queries of the EQFE system on the dataset (as measured by mean average precision) and collect for each query the top 19 documents.[10]

As the EQFE system used in this setting works not on a single entity e, but on the distribution of the EL system's confidence scores over the potential entity candidates per mention m, we aggregated all potential entities from all 19 documents d by computing the total reciprocal rank (TRR):

$$\text{TRR}_q(e) = \sum_d \sum_m \frac{1}{\text{rank}_{e,m,d}} \tag{7.5}$$

Computing the TRR instead of working directly with the confidence score distribution has the advantage of reducing the dependency between the EL system and the gold standard dataset we create from it. Note that using the mean reciprocal rank (MRR) would have created the same entity ordering. To filter out noise, the final dataset consist only of the 50 entities with the highest mention frequency per query.

Entity relevance was annotated separately by a pool of four different annotators, with each query being annotated by at least two annotators on a 5-level scale:

1	Non-relevant
2	Remotely relevant
3	Relevant
4	Very relevant
5	Highly relevant

Annotation disagreement were resolved by a standard adjudication process. The final relevance score is obtained by taking the arithmetic mean across all annotations, leading to the final relevance metric $rel \in [1-5]_{\mathbb{R}}$.

Figure 7.4 depicts the distribution of the annotation scores with a box-plot. The relevance scores indicate that the absolute majority of entities is not relevant. However, this category also includes incorrect entity links, as the entity linker used is of course not perfect. On the other end of the scale are the highly relevant (5) entities which are rarely found (mean 1.80) and some queries do not have any very (4) or highly (5) relevant entities at all. This is a result of our annotation guidelines, which require entities to be marked as highly relevant only if they clearly satisfy the information need expressed in the query. As a result of this, the relevance judgments provided by humans annotators are rather strict. In the case of the query "Argentine British relations", for instance, the entity `Falklands_War` receives a high relevance score (5). `Argentina`, instead is annotated only with a mildly relevant score (3.3) because, while being relevant w.r.t. query, it does not actually answer the question about the relationship between both countries.

[10] Which seems to be a reasonable approximation of the number of documents to be presented to the user by a retrieval system per results page.

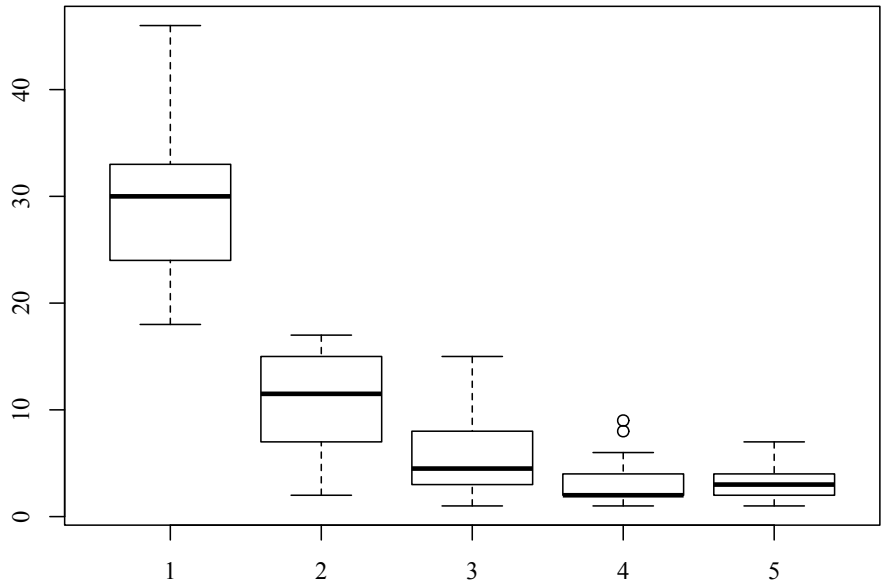

Figure 7.4: REWQ Robust04 Dataset: Boxplot for the annotations with the 5-level scale per query (25 queries, 50 entities each).

REWQ ClueWeb12 Dataset

In order to have a second benchmark with different properties, the ClueWeb12 dataset builds upon the TREC Web 2013/2014 queries together with the established ClueWeb12 corpus. Instead of focusing, as above for the REWQ Robust04 setting, on some specific subset of queries we choose here a random subset of 22 queries from the 100 TREC Web2013/2014 queries. For each query, the Sequential Dependency Model (SDM) (Metzler and Croft, 2005) as implemented in the Galago search toolkit,[11] is used to retrieve the top 20 documents. As this the SDM is a classical IR system not using entities, we ensure to eliminate any effects and potential gains given by EQFE's entity-linked documents. The ClueWeb12 datasat comes with a set of publicly available entity annotations, the FACC1 dataset (Gabrilovich et al., 2013), which allows us to also eliminate the effect of the specific entity linker (KB-Bridge Dalton and Dietz, 2013a) used in the Robust04 setting.

The final dataset consists of all entities per query – however, we heuristically filter out those entities occurring less than three times to remove many spurious entities from the data. This way we relax the assumption from REWQ Robust04 of using only the top-50 entities per query. Entity relevance is finally annotated in a standard (e.g, TREC-style) way using binary relevance judgments and not a 5-level scale.

Figure 7.5 depicts the distribution of the binary annotation scores for the 22 queries. Because the number of entities and thus annotations varies per query (between 50 and 488, average 215), the

[11] `http://www.lemurproject.org/galago.php`.

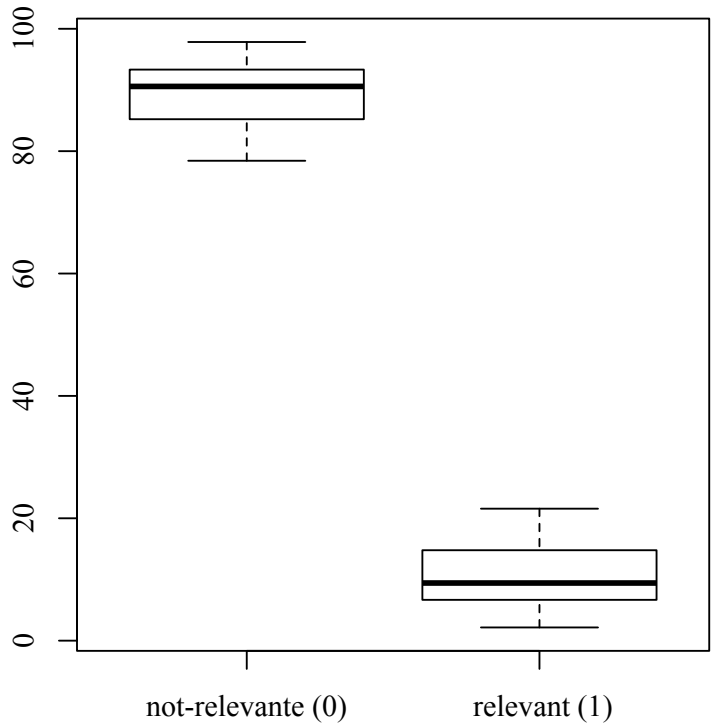

Figure 7.5: REWQ ClueWeb12 Dataset: Boxplot for the annotations with binary relevance (22 queries, different number of entities per query). Y-axis shows relative number in %.

box-plot shows the relative number of relevant (1) vs. not-relevant (0) annotations per query in percentage points (%). On average, only 9.6% of the annotated entities are labeled as relevant, emphasizing, as for the Robust04 task, who many of the entities extracted from the documents are actually not relevant w.r.t. the given query.

7.3.2 Experimental Setting

We evaluate our approach using all features from Section 7.2 within two learning-to-rank methods: (a) the SVM^{rank} implementation from Joachims (Joachims, 2006) and (b) the coordinate ascent methods as implemented in RankLib (cf. Section 7.2.3). Evaluation for both methods and datasets is performed with a linear 5-fold cross-validation. Parameter tuning for the SVM is done with an additional, random train-validation split, i.e. 3/5 training data, 1/5 parameter validation data, and 1/5 test data. For each fold, features are individually normalized with $x_{norm} = (x - \mu)/\sigma$, where mean μ and standard deviation σ are computed using only the training data folds. We compare the learned feature combinations against the following reference methods.

7.3.3 Metrics

We opt for two established evaluation metrics, depending on the nature of the ground truth annotations. For the REWQ Robust04 dataset with its graded relevance judgments (between 1-5), we choose nDCG@k in order to capture our intuition that a higher relevance should be honored by higher rank. For each query q we compute DCG by Järvelin and Kekäläinen (2000) as

$$DCG[i] = \begin{cases} CG[i], & \text{if } i < b \\ DCG[i-1] + G[i]/log_b(i), & \text{if } i \geq b \end{cases}$$

(cf. Järvelin and Kekäläinen, 2002) with log base $b = 2$ and normalized it to nDCG@k by dividing by the value of the ideal, i.e. gold standard sorted, ranking DCG@k^I. Note there exist different definitions of nDCG, for example Manning et al. (2008) define

$$DCG_q@k = \sum_{m=1}^{k} \frac{2^{rel(m)} - 1}{log(1+m)}$$

thus discounting in contrast to above also items at the first position (m=1).[12]

For binary judgments, we additionally report Mean Average Precision (Voorhees and Harman, 2005) for the REWQ ClueWeb12 dataset. All values are computed with the TREC Evaluation Script Version 9.0[13] and are reported in the following as the arithmetic mean over all queries.

7.3.4 Reference Methods

We compare the full-feature models against the following three reference methods.

Mention Frequency (*MenFrqIdf*): A ranking consisting only of the idf-weighted mention frequency feature. This feature's individual performance comes primarily from the quality of the initial document retrieval: Relevant documents should contain relevant entities. In case the entity linker provides more than one entity per mention (as for the REWQ Robust04 dataset with KB-Bridge), we take this ranked list of candidate entities into account by replacing the mention frequency ($\text{tf}_q(e)$) with the total reciprocal rank (TRR, see Equation 7.5). Ranking by TRR combines the frequency of occurrence of the mentions with the entity linker's confidence scores on the linking of the mentions to their entities.

Wikipedia Fulltext Index (*WikiSDM*): A ranking based on the scores from a Sequential Dependency Model (Metzler and Croft, 2005) retrieved from a retrieval index of Wikipedia text (using weight parameters from Dalton et al. (2014)). This baseline is closest in spirit to INEX-like entity retrieval from Wikipedia (Kaptein and Kamps, 2013) and is the alternative to our approach of issuing the query against a document retrieval system and then link the document to the knowledge base instead of querying the knowledge base directly.

[12] Resulting in lower absolute nDCG values, but – in our experience – stable relative differences between different queries or settings.

[13] `http://trec.nist.gov/trec_eval/`

Wikipedia PageRank (*WikiPR*): A ranking obtained by applying the (unpersonalized) PageRank algorithm to the link structure of Wikipedia, thus ranking entities by their global authoritativeness. PageRank scores are taken from the public dataset created by Thalhammer (2014).

7.3.5 Results on the REWQ Robust04 Dataset

We present results in Table 7.4, where we compare three learning-to-rank models, (1) SVM-rank with (w/ SK) and (2) without the Semantic Kernel (w/o SK), as well as (3) the coordinate ascent model from RankLib.

All reference methods (MenFrqIdf, WikiSDM, WikiPR) achieve high NDCG scores, with WikiSDM performing best with slightly above 0.9, indicating that the combination of entity candidate generation and external knowledge from Wikipedia is already a strong combination for entity ranking. It is in a way a combination of our idea to extract entities from query-relevant document and the approach to query the Wikipedia full-text that has been shown to perform well in the context of the INEX competitions. The low performance of WikiPR, in contrast, suggests that authoritativeness correlates, in our setting, only marginally with entity relevance. Error analysis reveals that entities ranked high by PageRank are often very general entities linked to by many other entities, e.g. `Earth`, `United_States`, etc., which obviously makes sense when applying PageRank to the undirected Wikipedia link structure.

Finally, we observe that our learning-to-rank methods performs better than the reference methods, reaching an overall NDCG score of 0.936 – the difference is statistically significant (according to a paired t-test, p-value $\leq \alpha = 0.05$). By re-ranking the entities with our method, we gain up to 3.7% in NDCG over the input ordering (MenFrqIdf), even though we have 'only' 50 entities per query. Among the different rankers, RankLib performs better than SVM-rank, with the semantic kernel (w/ SK) improving the SVM-rank results slightly (+.003 in NDCG).

When looking at the NDCG@10 scores, we observe the same trends:

1. The full-feature rankers beat all reference methods, which nevertheless achieve a very competitive performance, with WikiSDM ranking highest among them;
2. RankLib outperforms SVM-rank as learning methods, which achieves better scores when using a semantic kernel.

The larger relative improvements between baselines and supervised rankers suggests that our feature-based approach makes a difference in particular to move the relevant entities from the long tail into the top ten.

Narrative evaluation

For providing a more detailed insight into the entity ranking, Table 7.6 shows the results obtained from RankLib for the REWQ Robust04 data. Queries are sorted by the average ground truth values (*gt*) for the top 3 entities, thus showing queries with meaningful entities at the

Table 7.4: Evaluation results for REWQ Robust04 dataset. We report differences w.r.t. the best performing reference method (here WikiSDM), statistically significant improvements are denoted with † (paired t-test p-value ≤ 0.05).

Method	ndcg	Δ%	ndcg10	Δ%
RankLib	0.936	†3.7	0.817	†11.6
SVM (w/ SK)	0.926	†2.6	0.804	†9.7
SVM (w/o SK)	0.923	2.2	0.796	†8.7
WikiSDM	0.903	0.0	0.733	0.0
MenFrqIdf	0.885	-2.0	0.694	-5.3
WikiPR	0.778	-13.8	0.440	-40.0

Table 7.5: Evaluation results for REWQ ClueWeb12 dataset. We report differences w.r.t. the best performing reference method (here MenFrqIdf), statistically significant improvements are denoted with † (paired t-test p-value ≤ 0.05).

	map	Δ%	ndcg	Δ%	ndcg10	Δ%
RankLib	0.328	†9.0	0.572	†3.4	0.710	†10.0
SVM (w/ SK)	0.278	-7.8	0.545	-1.6	0.646	0.1
SVM (w/o SK)	0.308	2.2	0.563	1.6	0.675	4.4
MenFrqIdf	0.301	0.0	0.554	0.0	0.646	0.0
WikiSDM	0.234	-22.3	0.515	-7.0	0.613	-5.1
WikiPR	0.075	-75.1	0.328	-40.8	0.126	-80.5

top. Among the top queries we find e.g "poliomyelitis and post polio", for which we are able to retrieve expected and relevant, but not surprising entities like `Poliomyelitis`, `Polio_vaccine` or `Jonas Salk`, resulting in an NDCG@10 score of 0.879. Another interesting query with a very high NDCG@10 of 0.931 is "territorial waters dispute", for which not so well-known, yet relevant entities like `United_Nations_Convention_on_the_Law_of_the_Sea` are ranked high, as well as examples of specific water disputes taking place in the Mediterranean Sea (`Aegean_dispute`) and the Pacific Ocean (`Kuril_Islands_dispute`). The query on the bottom, "agoraphobia", has a low "gt" value because the initial document retrieval in combination with the entity linking could not obtain any really useful entities besides `Charles_M._Schulz`.

Error analysis

Error analysis on the low-performing queries reveals that our method suffers from errors in the entity links. For the query "Argentine British relations", for instance, the top retrieved entity is `Argentina_rugby_union_team`, which is actually an artifact of systematic errors from the entity linking system, which incorrectly links mentions like Argentine or "Argentina" to the national rugby team, and not to the country (`Argentina`). This suggests that a better entity linking could further boost our performance. Another source of errors comes from the retrieval

Table 7.6: List of all REWQ Robust04 queries sorted by average ground truth scores for top 3 entities (*gt*). Also showing the NDCG@10 score (*ndcg*) and the top-3 entities as retrieved by the RankLib coordinate ascent method (using the complete set of features). Ground truth values in brackets; abbreviated entities are denoted with *.

gt	ndcg	query	top-1 entity	top-2 entity	top-3 entity
5.0	.895	schengen agreement	Schengen_Agreement (5)	Schengen_Area (5)	Schengen_Inform._Sys. (5)
5.0	.894	magnetic levitation maglev	Maglev (5)	Shanghai_Maglev_Train (5)	Transrapid (5)
4.8	.931	territorial waters dispute	UN_Law_of_Sea* (4.3)	Aegean_dispute (5)	Kuril_Islands_dispute (5)
4.7	.865	el nino	El_Nino-Southern_Oscillation (5)	Pacific_Ocean (4)	La_Nina (5)
4.3	.942	ferry sinkings	MS_Estonia (5)	MS_Herald_of_Free_Enterprise (5)	Silja_Line (3)
4.3	.927	in vitro fertilization	In_vitro_fertilisation (5)	Fertility_clinic (5)	Shan_Ratnam (3)
4.3	.879	poliomyelitis and post polio	Poliomyelitis (4.5)	Polio_vaccine (4.5)	Jonas_Salk (4)
4.3	.787	osteoporosis	Hormone_therapy* (5)	Estrogen (5)	Nutrition (3)
4.0	.803	industrial espionage	Volkswagen (4)	General_Motors (4)	Opel (4)
3.8	.863	polygamy polyandry polygyny	Al-Arqam (4.5)	Code_of_Personal_Status_(Tunisia) (4.5)	Tunisia (2.5)
3.3	.769	amazon rain forest	Amazon_rainforest (5)	Manaus (2)	Amazon_Basin (3)
3.3	.748	argentine british relations	Foreign_relations_of_Argent. (4)	Argentina_national_rugby_team* (1)	Falklands_War (5)
3.2	.751	antarctica exploration	South_Pole (3.3)	Antarctica (4)	Antarctic_ecozone (2.3)
2.8	.860	supercritical fluids	Supercritical_fluid (5)	Euler_equations_(fluid_dyn)* (2.5)	Biodegradation (1)
2.8	.760	computer viruses	Michelangelo_(cmpt_virus)* (5)	Personal_computer (2)	Computer_industry (1.5)
2.7	.836	lyme disease	Lyme_disease (5)	Centers_for_Disease_Control* (2)	Old_Lyme,_Connecticut (1)
2.7	.669	falkland petroleum exploration	Falkland_Islands (4.3)	Falklands_War (2.7)	Stanley,_Falkland_Islands (1)
2.5	.739	hydroponics	NASA (4)	Jordan (1)	Mars (2.5)
2.3	.868	killer bee attacks	Africanized_bee (3)	Ceratitis_capitata (2)	San_Diego (2)
2.3	.831	implant dentistry	Dentistry (4)	Cochlear_implant (1)	Uni_of_Med_&_Dent_NJ* (2)
2.3	.749	agent orange exposure	Agent_Orange (5)	Agent_Orange_(band) (1)	Agent_Orange_(album) (1)
2.3	.718	king hussein peace	Hussein_of_Jordan (4.5)	Abdullah_II_of_Jordan (1)	Black_Septmbr_Jordan* (1.5)
2.0	.672	counterfeiting money	Counterfeit (3)	Los_Angeles (1)	Novosibirsk (2)
2.0	.582	unsolicited faxes	Fax (4)	Personal_computer (1)	ISDN* (1)
1.9	.966	agoraphobia	Charles_M._Schulz (3.3)	Snoopy (1.3)	UGM-27_Polaris (1)

system itself – e.g., low performance on the query agoraphobia comes from the rather noisy pool of documents we start with to collect potentially relevant entities.

In addition, we are also of course milted by the fact that some – to be expected – knowledge is not available from the KG. For example, DBpedia does not contain the direct information that `db:Carlos_Menem` was the president of Argentina. The KB relation depicted in Figure 7.3 connecting `db:Carlos_Menem` with `db:Argentina` is actually of type `dbo:nationality`. The fact that he was also Argentina's president is only encoded indirectly via the DBpedia property namespace – which we do not consider as argued above (cf. Section 2.1.3): `db:Carlos_Menem dbp:title db:President_of_Argentina`. Finally, low performance on some queries are due to their degree of difficulty, as highlighted by fine-grained queries for very specific domains (e.g., hydroponics), where additional knowledge could potentially help.

7.3.6 Results on REWQ ClueWeb12 Dataset

In Table 7.5 we report our results on the ClueWeb12 portion of the REWQ dataset. Similar to the Robust04 results, the single features perform quite well on their own. In contrast to the Robust dataset, the best single feature is the MenFrqIdf features. Again, our learning-to-rank approach outperforms all reference methods consistently across all measures, both when using RankLib and SVM-rank (up to +9.0% MAP, +3.4% NDCG, +10.0% NDCG@10).

Also in line with the Robust04 findings, the greater relative improvements of our method for the NDCG@10 value suggest that our features make a difference in particular for the top ranked entities. The performance of the SVM with Semantic Kernel (w/ SK) is worse in contrast, the MAP and NDCG scores are even below the MenFrqIdf feature. Because the NDCG@10 value is at par with the MenFrqIdf, we suspect that the knowledge base links between entities used by the Semantic Kernel are only helpful for the top entities - but fail when ranking within the long tail. Another factor is most likely the fact that this dataset has only binary annotations, and is thus not as fine grained as the 1-5 points scale of the REWQ Robust04 ground truth. In summary, we take these results to be additional evidence for our previous findings.

7.3.7 Feature analysis

To better understand the importance of the different features within our model, we study the individual ranking performance of each feature, and perform a feature ablation study.

Single features as rankers

Analyzing the individual features in isolation, Figure 7.6 (Robust04) and Figure 7.7 (ClueWeb12) show the NDCG@10 performance achieved by each feature individually. We find that the mention frequency (MenFrqIdf) and the Wikipedia fulltext search (WikiSDM) both perform individually well as ranking metric for both datasets. For the REWQ Robust04 dataset, WikiSDM is the highest performing feature. Since we are only re-ranking the most frequent entity mentions in high-ranked documents, the WikiSDM method is filtered by a very effective whitelist. This

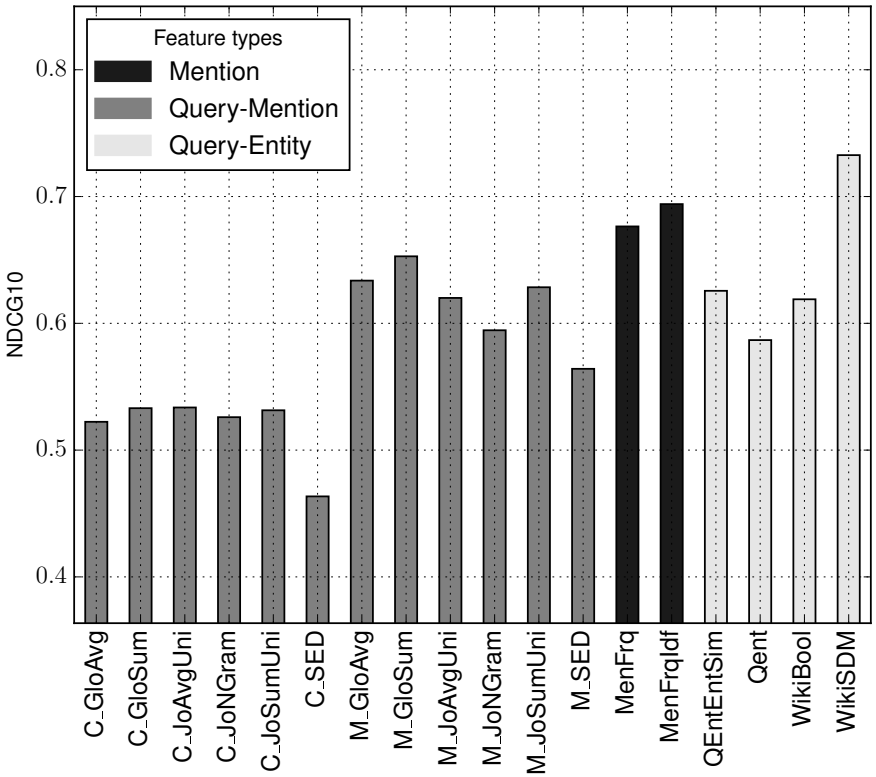

Figure 7.6: Feature-by-feature analysis for the REWQ-Robust04 Dataset (1-5 annotations, KB-Bridge EL)

confirms our intuition that entities which occur often in relevant documents are themselves relevant, but also that ranking entities based on their Wikipedia article according to WikiSDM is a non-negligible indicator. All context-based query-mention-features (indicated by prefix *C_*) perform worse than their no-context counterparts (indicated by prefix *M_*), e.g. C_GloSum vs. M_GloSum, thus letting us question their value for entity ranking. However, their benefit is only demonstrated in combination with other features.

The contribution of the other query-entity features, which are based on DBpedia, namely Qent and QEntEntSim, are in between – they perform worse than the strong WikiSDM, but than some of the mention-based approaches. On both dataset, QEntEntSim as single feature performs better than the QEnt feature. Since QEntEntSim is leveraging knowledge base paths and ontological types between entities in the query and the documents, these provide a meaningful way to connect otherwise missing entities.

In summary, the high performance of the MenFrIdf features highlights that the candidate generation strategy already provides a useful approach on its own: this finding holds for both datasets despite using different document retrieval and entity linking methods.

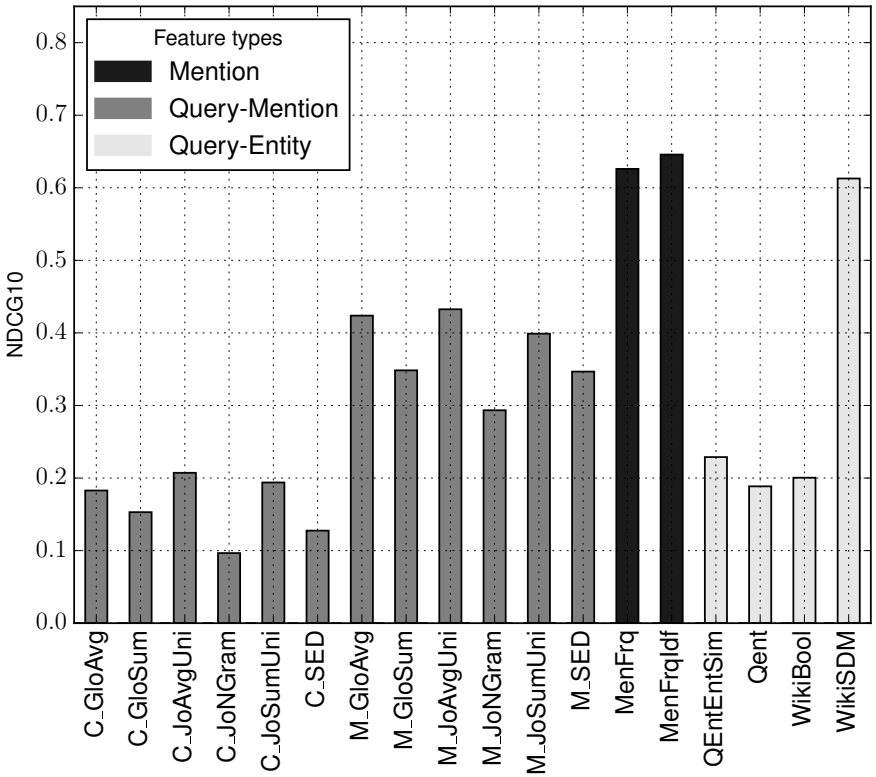

Figure 7.7: Feature-by-feature analysis for the REWQ-ClueWeb12 Dataset (binary annotation, FACC1 EL)

Ablation study on Robust04

To further analyze the individual features, we perform a features ablation study: For each single feature, or set of features, we remove it from the set of features available to RankLib, re-train it with the same parameters and compare its performance against the full-feature setting. Results for the Robust04 are reported in Table 7.7 which is sorted by relative loss caused by removing a feature (group).

Surprisingly, we find that removing all mention-based features (i.e. SED, Jo, Glo for mention and mention context) actually improves the overall performance by 0.1% in NDCG (0.7% in NDCG@10) – however, the differences are small and not statistically significant. This finding might also results from the fact that the MenContext group combines features of different quality: While the string edit distance (SED) is helpful (-1.3% NDCG, -2.8% NDCG@10), we cannot confirm this for the JoBim text features (-0.2% and +0.3%).

The DBpedia-based features (DBpedia) seem to have a positive influence on the overall performance (-1.0% and -1.9%), even though not being statistically significant. Interestingly, removing any of the two DBpedia features QEntEntSim or Qent individually would let to a different

conclusion.

The Wikipedia-based features show a strong and significant influence on the overall performance, removing them leads to a drop of -2.3% for NDCG and -5.1% for NDCG@10. The single most important feature is the mention frequency features (MenFrqIdf), thus supporting our assumption that a good initial document retrieval helps to obtain a good pool of relevant candidate entities.

Ablation study on ClueWeb12

The findings for the ClueWeb12 dataset in Table 7.8 confirm the findings from the Robust04 dataset above. Again, leaving out all mention-based features actually improves the performances – but as above, the difference is not statistically significant. On the other end of the table, and also in line with above findings, the mention frequency is the single most important features with rather large differences between 19.7% (MAP) and 6.5% (NDCG).

We can also confirm our finding that the simple SED is more effective than the Glove and Jo features. The role of the DBpedia features is slightly different, they seem to be even less helpful for the ClueWeb dataset than for the Robust04 dataset. A possible explanation is the the difference in the annotation method: The binary ClueWeb12 annotations are likely to not capture fine-grained differences between entity relevance levels, which might be expressed by knowledge-base links.

In summary, all findings for this dataset are in line with the findings for the Robust04 data, which is interesting because both datasets are rather different in nature, i.e., different ground truth labels (binary vs. 1-5 scale), different document retrieval (SDM vs. EQFE), and different entity linkers (FACC1 vs. KBBridge).

7.4 Related Work

To the best of our knowledge, there is no work that is addressing exactly the very same setting we are looking at, i.e. given a general web query and a list of retrieved documents, rank the Wikipedia entities extracted from those documents w.r.t. the query. However, there exists a large body of closely related work on (query-driven) entity retrieval and ranking for Web search.

In the remainder of this section, we study the commonalities and differences between different task definitions and our understanding. Wherever meaningful, we also present and compare selected methods. Table 7.9 gives an overview about the characteristics of the major view on entity retrieval/ranking discussed next.

Table 7.7: Feature ablation study on REWQ Robust04 Dataset. Sorted by difference ($\Delta\%$) in NDCG value. P-values (p) from two-sided paired t-test ≤ 0.05 are denoted with †.

w/o	ndcg	$\Delta\%$	p	ndcg10	$\Delta\%$	p
RankLib All	0.936	-	-	0.817	-	-
MenContext	0.937	0.1	0.68	0.823	0.7	0.56
QEntEnt	0.935	-0.1	0.73	0.824	0.8	0.58
Qent	0.934	-0.2	0.58	0.825	0.9	0.44
Jo	0.934	-0.2	0.53	0.819	0.3	0.85
Context	0.933	-0.3	0.28	0.816	-0.1	0.89
Glo	0.928	-0.8	0.10	0.803	-1.7	0.26
DBpedia	0.927	-1.0	0.06	0.802	-1.9	0.21
WikiBool	0.926	-1.1	0.11	0.809	-1.0	0.56
SED	0.924	†-1.3	0.05	0.794	-2.8	0.09
WikiSdm	0.921	†-1.7	0.03	0.781	†-4.4	0.04
MenFrqIdf	0.917	†-2.1	0.04	0.774	†-5.4	0.05
Wikipedia	0.914	†-2.3	0.01	0.776	†-5.1	0.03

Table 7.8: Feature ablation study on REWQ ClueWeb12 Dataset. Sorted by relative difference ($\Delta\%$) in MAP value. P-values (p) from two-sided paired t-test ≤ 0.05 are denoted with †.

w/o	map	$\Delta\%$	p	ndcg	$\Delta\%$	p	ndcg10	$\Delta\%$	p
RankLib All	.328	-	-	.572	-	-	.711	-	-
MenContext	.333	1.4	.41	.574	0.3	.55	.714	0.5	.70
Jo	.332	1.0	.55	.573	0.2	.69	.716	0.8	.50
DBpedia	.329	0.1	.92	.572	0.0	.90	.701	-1.4	.26
QEntEnt	.327	-0.4	.48	.572	-0.1	.68	.708	-0.4	.64
Context	.326	-0.6	.49	.570	-0.3	.34	.698	-1.7	.06
Glo	.326	-0.7	.51	.571	-0.3	.46	.698	†-1.7	.05
Qent	.326	-0.8	.63	.571	-0.2	.75	.701	-1.4	.32
SED	.326	-0.8	.35	.571	-0.3	.46	.698	-1.8	.15
WikiSdm	.320	-2.6	.25	.566	-1.1	.26	.693	-2.5	.28
WikiBool	.313	†-4.6	.05	.565	-1.3	.08	.670	†-5.7	.01
Wikipedia	.303	†-7.7	.02	.556	†-2.9	.02	.650	†-8.5	.02
MenFrqIdf	.264	†-19.7	.00	.535	†-6.5	.01	.630	†-11.4	.03

Table 7.9: Overview of the different types of entity retrieval/ranking

Section	Short Name	Retrieval from	Entity Type	Query Type
7.4.1	INEX	Knowledge Base	KB (Wiki)	Entity, Type
7.4.2	TREC	Web	Open, KB	Type, Complex
7.4.3	AOR	Web, KB	Open, KB (LD)	Entity
7.4.4	Entity Ranking	Docs	Open, KB	no query
7.2	Our work	Doc Collection	KB (Wiki)	Complex

7.4.1 Knowledge Base Retrieving of Entities for Type Queries

The task of retrieving entities (or a single entity) from a given knowledge base, very often Wikipedia, is well-known and prominently promoted by the Initiative for the Evaluation of XML Retrieval (INEX), that we introduce next.

INEX ER and LC Task 2009

The 2009 edition had two tasks (Demartini et al., 2010):

(a) The entity ranking (ER) task, where the aim was to return Wikipedia entities that satisfy a topic described in natural language, for example "art museums in Amsterdam". In addition, a preferred category was given, e.g. "art museums and galleries".

(b) The List completion (LC) task provided also a natural language query, but instead of specifying a category, entities from the correct category are given.

Comparing both INEX tasks with your problem definition, we see that entities are expected to be of a particular type (here specified by a Wikipedia category), either explicitly (ER task) or implicitly (LC task). Describing the INEX task in terms of the type of query they use, as defined by Pound et al. (2010) and described in Section 7.1.3, both tasks are entity queries: They ask for entities of a particular, given type. Our work, in contrast, aims explicitly for a collection of entities of different types, e.g. persons and location involved with in a topic.

INEX Linked Data Track 2013

In 2013, INEX ran 4 tracks, of which the Linked Data Track is most relevant to us. It consisted of two tasks (Bellot et al., 2013), of which we discuss only task (a):

(a) The Ad-hoc Search Task asked for Wikipedia entity retrieval given an information need.

(b) The Jeopardy Task asked for formulating SPARQL queries for information needs in natural language.

The Ad-hoc task provided 72 "classical" keyword queries like "best movie" and asked for a ranked list of (up to 1000) Wikipedia entities.[14] The task focuses thus on answering queries

[14]The 72 queries were also used in the INEX 2009 and 2010 edition.

"mainly by the textual contents of the Wikipedia articles" (Gurajada et al., 2013, p. 2), that is, it looks primarily at ways to retrieve ranked list of articles from Wikipedia given a keyword query using the Wikipedia text itself. In contrast, in our work we are interested in ranking entities 'in the wild', namely as found in the entity-linked content of retrieved documents. The Ad-hoc task had only 3 participants.

Kaptein and Kamps: Leveraging Wikipedia Categories

Kaptein and Kamps (2013) propose a system for the INEX ER task of retrieving entities from Wikipedia of a given type (Wikipedia category), which works for that reason with the Wikipedia categories. The system works as follows.

(a) For retrieving an initial list of entities, the query is issued against a document retrieval system, more specifically against a standard language model, with Jelinek–Mercer smoothing without length prior, built from the Wikipedia full-text articles.

(b) The retrieved entities get filtered on the target category, whereas Kaptein and Kamps propose different similarity metrics to measure if an entity belongs to the desired target category.

(c) Information on the (Wikipedia hyper-) links between entities is taken into consideration by computing the ratio between local indegree (only between retrieved entities) and global indegree (all entities). An additional use of the link information is through relevance propagation from the initially retrieved entities (as first proposed in the context by Tsikrika et al., 2008).

Finally, all feature scores are aggregated by different linear combinations.

Comparing this system with our work, we see that the language model from step (a) is very similar to our Wikipedia-based features, WikiBool and WikiSDM, which were actually inspired by this work. We find WikiSDM to be among the top-performing features as reported in Section 7.3.7. In their experimental evaluation, Kaptein and Kamps find the category information to be very helpful – which seems reasonable given the task defines target categories. Having a different task, we do not leverage categorical features in our work.

Raviv et al.: MRF for Joined Feature Model

Raviv et al. (2012) present an approach for the INEX entity ranking (ER) task (2007-2009) that is discussed here because it follows the same idea we did by combining different types of features for entity ranking. More specifically, they model (i) entities mentions occurrences in the documents, (ii) the entity type, and (iii) the entity name. While we combined our independent features linearly with learning-to-rank (and the semantic kernel for entity-entity features), Raviv et al. explicitly model the dependencies between query and entity as a Markov random field (MRF) (Metzler and Croft, 2005). However, the experimental evaluation reveals that "various dependence assumptions did not result in significant improvement in the model performance

over using the full independence assumption", thus letting their supervised ranking method be in the end very similar to the linear feature combination we learned with the RankLib implementation of the listwise LTR method. Note that the direct parameter optimization for MAP with Coordinate Ascent (Metzler and Croft, 2007) used by us via RankLib is actually also used by Raviv et al. for optimizing their model.

7.4.2 Web Retrieval of Entities for Typed Queries

An alternative to knowledge base retrieval is to retrieve entities from the web, i.e. from some arbitrary resource, instead. When retrieving not form a given KB like Wikipedia or some LOD resource, the definition of what an entities is, can also be relaxed – however, having a more open definition of what an entity is makes evaluation also more complicated, and methods harder to compare. The most notable evaluation campaign in this context is the TREC Entity Track, which runs as part of the long standing IR evaluation imitative TREC (Text Retrieval Conference).[15]

TREC Entity Track 2009

The 2009 TREC Entity Retrieval Track defined "entities as 'typed search results', 'things', represented by their homepages on the web" (Balog et al., 2010, p. 1). The main task, Related Entity Finding (REF), asked to retrieve entities related to the input entity, an example query (topic) is shown below ((Balog et al., 2010, p. 2)):[16].

```
<query>
 <num>7</num>
 <entity_name>Boeing 747</entity_name>
 <entity_URL>clueweb09-en0005-75-02292</entity_URL>
 <target_entity>organization</target_entity>
 <narrative>
   Airlines that currently use Boeing 747 planes.
 </narrative>
</query>
```

The pre-defined types for the target entities were person, organization, and product. Thus, this task is similar to the type queries from INEX, however, this TREC task does not provide a "classical" keyword query. The expected output, i.e. the search result, for the task was a list of up to 100 entities. Each entity could be described by a collection of websites (divided into homepages, Wikipedia page, and supporting documents) and a string answer that represents the entity concisely.

Comparing the expected output against our Wikipedia definition, we notice that there is only a partial overlap: If the Wikipedia page is included in a TREC entity returned, it can actually be interpreted as a Wikipedia entity with the same semantics we, and e.g. INEX, uses. However, as the Wikipedia URI is an optional information, it might be missing: either because the

[15] http://trec.nist.gov.

[16] Note instead of the true URL, the document id of the ClueWeb09 corpus is given.

entities has no Wikipedia page due to its limited coverage, or because finding/mapping to the correct Wikipedia page failed. In summary, this entity definition is rather different from our understanding.

TREC Entity Track 2011

In 2011, the REF task was extended by a linked open data (LOD) variant (Balog and Serdyukov, 2011). In this REF-LOD task, instead of homepages, URIs from the LOD cloud were given, using the Sindice-2011 LOD-crawl dataset (Campinas et al., 2011). However, the REF-LOD task had only one participant and was discontinued later.

In addition, TREC offered the Entity List Completion (ELC) task, whose definition was essentially the same as for the REF task, i.e. finding entities related to the input entity. However, type of the target entity was this time specified via its `rdf:type` from the DBpedia Ontology (cf. Section 2.1.3). The output of relevant entities where expected to be denoted by a URI from the provided Sindice LOD dataset.

In summary, the TREC Entity Track developed into the direction of our more strict entity definition by using a given LOD dataset to obtain entity-identify URIs. Nevertheless, the main idea of the track's tasks is still to find entities related to a given input entities, where the relationship between both is described by a textual narrative.

7.4.3 Semantic Search as Ad-hoc object retrieval (AOR)

Semantic search is a term with various interpretations, depending strongly on the community (e.g. Semantic Web, Information Retrieval) it is used by. Without going into further details, we look here at the Ad-hoc object retrieval (AOR) task as defined by Pound et al. (2010): Given a keyword query, return a ranked list of object. The definition of what an object is remains open, thus being similar to the open TREC definition. The experimental study by Pound et al. uses metadata (RDFa and different Microformats) embedded within website found in the query logs of search engine as object identifiers.[17] The Semantic Search 2011 dataset[18], created from Yahoo's search engine log, follows directly this understanding of AOR. In the end, AOR is in the middle between the website retrieval of TREC, and the knowledge base retrieval from INEX: It uses classical keyword queries for ad-hoc retrieval, but allows also non-Wikipedia entities as results.

Ciglan et al.: Semantic Sets for AOR Type Queries

The work by Ciglan et al. (2012) is interesting bycause they try – like us – to leverage the DBpedia graph for finding semantically related entities. Their SemSets system aims at answering type queries for the AOR task. For evaluation, the SemSearch dataset is used, but only Wikipedia

[17] For more details on RDFa and Microformats usage in websites see Bizer et al. (2013).

[18] `http://km.aifb.kit.edu/ws/semsearch11` and Tran et al. (2011)

entities are taken into account – the presented approach would thus actually also work on the INEX Ad-hoc task.

Ciglan et al. start by linking the keyword query to entities from their knowledge base (the Wikipedia entities restricted SemSearch RDF graph). This is a step we also do for feature generation, however we use a full-pipeline entity linking system. In contrast, the SemSets systems uses only the surface form to entity probabilities (which is the most important component of Wikipedia entity linking systems, (Milne and Witten, 2008b)), because they target type queries, which means e.g. `List_of_Apollo_astronauts` would be a very good entities for the query "Apollo astronauts who walked on the Moon".

For ranking this initial list of entities, "artificial" documents containing all entity properties are created and standard document retrieval methods are applied. Those textual features are combined with structural features by applying an activity spreading based method to the property graph, i.e. essentially the knowledge graph around the initial entities. The last ranking step uses the, name giving, SemSets, i.e. sets of semantically related entities – as computed via the DBpedia graph. This step is similar to our DBpedia path finding (*QEntEntSim*), but relies on different structural metrics like internal density. In line with our observations, Ciglan et al. find all graph-based metrics to have rather low MAP scores when used as single ranking method. For that reason, they set thresholds and filter the semantic sets of entities in addition by textural features, incl. the DBpedia abstract and properties. In the overall experimental evaluation, the DBpedia (property) graph structure based methods were not able to outperform the textual cosine similarity of the entity properties.

Interestingly, Ciglan et al. mention in their work that for a type query, a "human user would probably enter such a query to a web search engine and inspect several top-k results and [...] search the text of the inspected documents to find the desired set of entities" (Ciglan et al., 2012, p. 131). This is exactly the pipeline we created in our work to solve the entity ranking problem, while Ciglan et al. opt to propose a technique that does not include documents from web search.

Zhiltsov and Agichtein: LeToR Entities with RDF tensors

Zhiltsov and Agichtein (2013) present an approach to keyword search over RDF data, following the AOR definition from Pound et al. (2010). Consequently, they work also on the SemSearch query dataset, together with the Billion Triple Challenge (BTC) 2009 RDF as data collection, which contains, amongst others, the data of DBpedia, LiveJournal, GeoNames, and DBLP. Interestingly from the technical perspective, Zhiltsov and Agichtein (2013) combine, like us, textual features (e.g. name and label of RDF resource) with structural features (RDF predicates) by feeding both features types in a learning-to-rank method. However, in contrast to our entity path search between entities (QEntEntSim), Zhiltsov and Agichtein model the full structural dependencies, i.e. predicate between entities as a tensor – most likely because they retrieve entities directly from the fully structured source dataset, i.e. the RDF BTC data, which makes such an approach a more natural choice. Even though the evaluation results are not comparable, it's noteworthy that the best NDCG score reported is at 0.40, which is below the numbers for both

our REWQ datasets – thus being a good reminder how well the entity pre-filtering by the initial document retrieval works.

7.4.4 Entity Retrieval from Documents without Queries

Oppose to retrieving entities from the Web or a KB, we now turn to retrieving entities from documents. While the question of entity extraction and ranking also arises in this context, note that no query is involved here.

Dunietz and Gillick: Ranking Salience Entities in Documents

Dunietz and Gillick (2014) address the question of ranking entities from documents, which we also implicitly touch by ranking entities retrieved from the search result documents. The authors define the task of "entity salience [as] assigning a relevance score to each entity in a document" (Dunietz and Gillick, 2014). Not that while we rank entities from the whole collection of retrieved documents, because we aim at ranking entities w.r.t. the query, Dunietz and Gillick (2014) rank entities only w.r.t. to the individual document. Even though this query-independent perspective lets the work be rather different from the tasks presented above and our own work, from a technical perspective, the methods used are similar to those we use. In their presented work, Dunietz and Gillick use only entities that could be linked Freebase by an entity linker, and only those entities that contain at least one proper-name in the mention.

Besides other features, Dunietz and Gillick compute entity centrality by applying PageRank to the Freebase graph of entities found within the document. In our experiments, a PageRank on the full DBpedia graph was not a helpful feature – limiting the PageRank to entities of one document seems to be the crucial choice here. In their analysis, Dunietz and Gillick find however that the centrality features do not significantly improve accuracy over their mention features ((F_1 61.6 vs. 62.0) They conclude, that the mention features, in particular the frequency statistics on the entity mentions, are already sufficiently powerful (F_1 60.3). This finding can be confirmed by our own analysis, where the mention frequency (MenFrqId) is also a very strong.

7.5 Conclusion

In this chapter, we addressed the problem of ranking entities for complex, open-domain web queries. In contrast to direct knowledge base retrieval (like INEX), our starting point were the query-relevant documents retrieved by a document retrieval system. We investigated the performance of a variety of heterogeneous features, which were combined by established learning-to-rank methods.

Key Findings

With respect to RQ1, we find that, based on our two self-created entity relevance datasets, documents retrieved with standard IR methods indeed contain entities that are relevant for the initial user query. While not surprising, it is an interesting confirmation of our intuition and justifies

our research into the combination of document retrieval and entity retrieval in this query-specific setting.

Regarding the actual entity ranking method (RQ2), our results indicate that query-relevant documents with entity links provide a complementary source of information to direct KB, i.e. Wikipedia, retrieval, yielding an NDCG@10 score of over 0.82 on Robust04, compared to 0.73 for Wikipedia retrieval. Together with the frequency of entity mentions within the retrieved documents (0.68 NDCG@10), Wikipedia retrieval (WikiSDM) is one of the strongest individual features. It is in a way a combination of the idea to extract entities from query-relevant document and the approach to query a Wikipedia full-text that has been shown to perform well in the context of the INEX competitions.

For most other results of the ablation study we cannot find significant differences. For example, we cannot find a unique significant contribution of features based on distributional similarity for this task (JoBimText and GloVE). Likewise, incorporating relations between entities does not yield a measurable benefit (QEntEnt and Semantic Kernel). Nevertheless, combining all these signals together within a supervised learning framework is able to yield statistically significant improvements over ranking by single features, so as to yield competitive NDCG scores.

Limitations

The most sever limitation of our approach is obviously that its final performance relies on the performance of the underlying document retrieval and entity linking systems, and error analysis revealed that our ranking does actually suffer from systematic errors from these two components. However, our rather high NDCG scores on both datasets – which use different document retrieval and entity linking systems – indicate that our supervised approach is able to cope with the noise in the input data. Nevertheless, to overcome the limitation of the document retrieval and entity linking, one could think of integrating direct KB retrieval into our pipeline as a second source of query-relevant entities (similar to the work of Dalton et al. (2014) who used entity retrieval for document retrieval). This would combine document retrieval and knowledge base retrieval into one results set – however at the cost of presenting the user entities that are not aligned to the documents.

Chapter 8

Finding Relevant Relations

This last chapter presents a natural extension of the work described above, as we will extract for the first time in this thesis not only entities, but also relations from a given text. The extraction of entities and relations from text opens up a new line of research, and we are going to present initial experiments, i.e. descriptive studies using a self-created annotation dataset.

The work presented in this chapter has been published before in *Michael Schuhmacher, Benjamin Roth, Simone Paolo Ponzetto, and Laura Dietz. Finding Relevant Relations in Relevant Documents. In Proceedings of ECIR'16, pages 654–660* (Schuhmacher et al., 2016).

As in the previous chapter, we study a query-driven IR setting, while the task is here to identify query-relevant relational facts, i.e. subject–predicate–object triples where subject and object are (Wikipedia) KB entities. Besides being the next step to make an integrated usage of text and KGs, our motivation for this task was also to bridge the gap between research in document retrieval and knowledge base population, as we set out a pipeline of document retrieval and relation extraction – in contrast to Chapter 7 where we combined document retrieval with entity extraction.

This chapter will not present a system or method for the task of finding relevant relations, but instead present an extensive study of the problem using a self-created dataset. We leave the further exploitation of our findings in a working software implementation to future research. We study the following research questions:

- RQ1: Can the approach extract relevant facts for the queries?
- RQ2: What are useful document- or KG-based features for fact relevance?
- RQ3: Is relevance of entities and relevance of facts related?

8.1 Introduction

Our goal for this chapter is to obtain query-relevant facts from query-specific documents, analogue to our previous attempts to find query-relevant entities in Chapter 7.[1] In this context, for us, a fact is a subject-predicate-object triple where subject and object are KB entities, here DBpedia entities. We opt for this definition in order to enable a tight integration with existing knowledge (from the DBpedia KG).

Our vision is to create a query-specific knowledge graph (KG) as illustrated in Figure 8.1 that integrates the information we find within the documents together with the encyclopedic knowledge from the DBpedia KG. In the end, we thus want to be able to answer a query, like "raspberry pi" in Figure 8.1, with relevant information directly in a structured and machine readable format, e.g. for a deeper analysis of the topic, and not contained within documents. However, even when restricting ourselves to identify only existing KB entities in the text, a full integration would also require the relations extracted from the text documents to be combined with the DBpedia KG predicates. In this first step towards our vision, we refrain from this full integration and only analyze the relationship between textual relations and KG predicates instead.

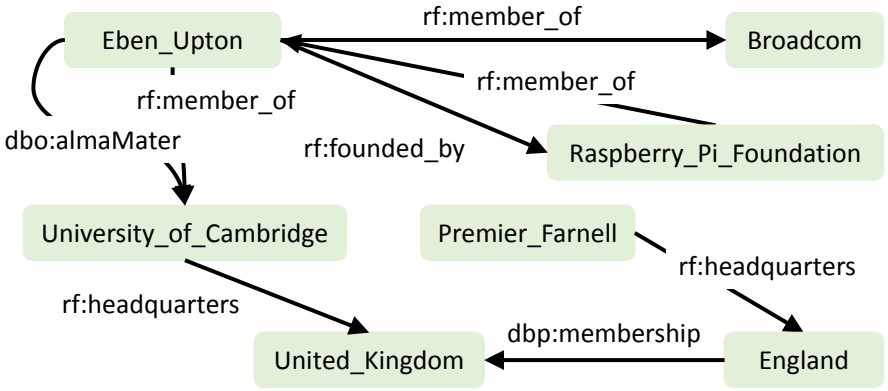

Figure 8.1: Example of a knowledge base for the query "raspberry pi". `rf:` denotes relations extracted from documents, whereas `dbp:` and `dbo:` are predicates from DBpedia. Note that the entity `Raspberry_Pi` itself is not included here, as the TAC schema of the relation extraction knows only three types of entities, namely Persons, Locations, and Organizations. The raspberry pi was invented by Eben Upton and its major distributor is Premier Farnell.

Our contribution here is thus the extraction of query-specific facts from query-relevant documents, as retrieved from an document retrieval system. We describe the entity-containing facts extracted, and evaluate if they are relevant w.r.t. the initial query. But we do not aggregate and/or integrate the extracted facts into a coherent and unified KG. The task we thus define for this chapter is as follows:

Task: Given a query Q, use the documents from a large collection of Web documents to extract

[1] We use the term query-specific and not query-relevant here, as, naturally, any document retrieval system cannot guarantee that the returned documents are actually query-relevant.

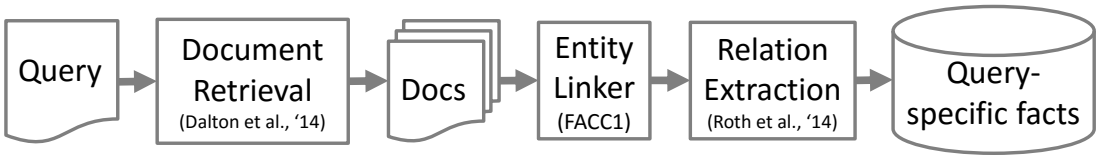

Figure 8.2: Schematic workflow combining document retrieval with information extraction

facts, i.e. subject–predicate–object triples (S, P, O) that are both correctly extracted from the documents' text and relevant for the query Q.

We illustrate the desired output KG with the example shown in Figure 8.1: Assuming a user wants to know about the Raspberry Pi computer, s/he should be provided with a KB that includes the fact that its inventor, Eben Upton, founded the Raspberry Pi Foundation, that he went to Cambridge University, which is located in the United Kingdom, and so on. This answer-like KG would be by far more expressive than to return e.g. just a list of entities, as we did before. But at the same time, this highly aggregated (and machine-readable) facts would be clearly more valuable information to the user than a list of documents. In addition, having extracted fact containing KB entities, we can complement the document-extracted information with KG facts, e.g. from DBpedia as depicted in Figure 8.1. Note that the graph shows only relations we annotated as being query-relevant in our dataset, as the perfect KB should include only facts that are of interest for understanding the query topic, e.g., `Raspberry_Pi_Foundation founded_by Eben_Upton`.

Based on our self-created benchmark dataset,[2] we present first experiments on building query-specific KBs from documents retrieved from a large-scale Web corpus in this chapter. Our pipeline combines two state-of-the-art system, one for document retrieval (Dalton et al., 2014) and one for relation extraction (Roth et al., 2014). This way, we go beyond the work on identifying relevant entities for Web queries in Chapter 7, where relations between entities were not considered, and also beyond query-agnostic knowledge base population (KBP) such as the TAC Cold Start KBP task, where determining fact relevance is not taken into account. Understanding this work as a first step that just combines established methods, we aim at quantifying how well this direct application of a relation extraction system to a set of retrieved documents solves the task of extracting query-specific facts. In order to differentiate between different error types, we evaluate the *correctness* of each fact extraction, i.e. if the extraction from text is correct, separately from the *relevance* of the fact, i.e. if the correctly extracted fact is relevant for the query.

8.2 Method

Our approach can be described by the pipeline shown in Figure 8.2 which consists of two major steps, namely a document retrieval and a relation extraction system.

[2]Dataset and additional information is available at `http://relrels.dwslab.de`.

For the first step, i.e. the initial document retrieval, we use the Galago[3] search engine to retrieve documents D from the given corpus that are relevant for the query Q. We build upon the work of Dalton et al. (2014) and rely on the same document pool and state-of-the-art content-based retrieval and expansion models, namely the sequential dependence model (SDM), the SDM model with query expansion through RM3 (SDM-RM3), and the SDM model with query expansion through the top-ranked Wikipedia article (WikiRM1). Without going into much details here, we only want to highlight that this retrieval system makes use of KB entities (from Freebase, provided by the FACC1 collection, cf. next paragraph) in addition to the standard document-based retrieval features. The choice for this system was thus not only motivated by its retrieval performance, but also because of this KB-entity aware retrieval, because we need documents with (many) entity mentions.

In the second step, for each retrieved document the facts are extracted using a relation extraction system. A prerequisite for running the relation extraction system is to first identify candidate sentences that mention two entities, acting as subject S and object O. Instead of identifying the entities ourselves by running an entity linking system, we opt to use an existing standard dataset, namely the FACC1 collection of entity links (Gabrilovich et al., 2013) for the ClueWeb12 document corpus (cf. Section 7.3.1) we used in our experiments.

Having the entities given, we select all such sentences as candidates for the relation extraction that contain at least two canonical entities of which the subject is of (Freebase) type `people` or `organization`. We limit ourselves to those types of entities, as the relations extraction system we employ, RelationFactory[4] (Roth et al., 2014), is trained only for those entities. RelationFactory was built for the TAC KBP Slot filling task and was the top-performing system in 2013. It is a modular system based on distantly supervised classifiers and patterns. Like all systems for TAC KBP, it was trained on Freebase data to extract facts for a schema of 42 relations/predicates, where the subject has to be of type `person` or `organization`, e.g. `born-in` or `city-of-headquarters`. Because we have the entity mentions already given from the FACC1 data, we skip the candidate generation phase in RelationFactory and use only its "validation modules": Each validation module, most of them per-relation SVM classifiers, but also automatically induced schemas and manually created patterns, makes a prediction if one of the predefined relations was found between the two entity mentions in the text.

8.3 Evaluation

We turn to the actual contribution of this chapter, namely the in-depth evaluation of how well the pipeline of document retrieval and relation extraction performs for finding query-relevant facts. First, the dataset construction is described, before we report on results and findings.

[3] `http://lemurproject.org/galago.php`
[4] `https://github.com/beroth/relationfactory`

8.3.1 Dataset

To our knowledge and as of April 2016, there exists no test collection for evaluating relational facts with respect to query-relevance. We created our own dataset thus, by augment an existing test collections for document-relevance and entity-relevance with assessments on the correctness and the query-relevance of facts. As queries, we sample from the test queries from the TREC Web track and retrieve documents from the corresponding ClueWeb12 corpus. [5] The fact-relevance assessments are partially build on top of the REWQ gold standard of query-relevant entities, as introduced in the previous chapter (Section 7.3.1).

As mentioned above, RelationFactory, being a closed relation extraction system, can only extracts a fixed set of relations it was designed for. This yields the obvious problem that not each of the TREC test queries can be adequately answered when being restricted to only certain relations, and thus certain types of entities acting as subject and object (here persons, organizations, and locations).[6] For that reason, we perform a shallow manual check of the TREC queries and focus in this study on the subset of 40% of TREC Web queries, such as "Raspberry Pi", for which we anticipated relevant facts to be covered by the TAC relation schema RelationFactory used.

For a random selection of 17 TREC queries (out of the 40% of queries we assessed before to be suitable at all for out setting), we manually assess the 40 most frequently mentioned facts and, in addition, all facts of which at least one of the entities was marked as relevant in the REWQ dataset. Due to the high number of annotations needed – 914 facts and 2,658 provenance sentences were assessed in total – each item was inspected by only one annotator. We ask annotators to assess for each fact,

- the correctness of the extraction from provenance sentences and
- the relevance of the fact for the query.

To assess relevance, assessors are asked to imagine writing an encyclopedic (i.e., Wikipedia-like) article about the query and mark the facts as relevant if they would mention them in the article, and non-relevant otherwise.

The number of provenance sentences per fact ranges from 1 to 82 with an average of 2.9, i.e. that on average each distinct subject–predicate–object triple for a given query was found approximately three times in the text documents. Note that this can also originate from near-duplicate documents, which the ClueWeb12 corpus, being a real Web corpus, contains of course. We define facts as correct when at least one extraction is correct, which leads to 453 out of 914 facts that are correctly extracted. Of these, only 16 facts includes both correct and incorrect extractions. The fact extraction correctness is thus at 49.6%, which is higher than the precision

[5] `http://lemurproject.org/clueweb12`

[6] An example is TREC query 223 "Cannelini beans", which most likely should include many information about plants, their types and relationships. This cannot be provided by the standard setting of RelationFactory, which was trained for persons and organizations to be the subject of a fact triple.

obtained in the TAC KBP shared task, where about 42.5% of extractions are correct. The assessment of relevance is performed on these 453 correctly extracted facts, leading to a dataset with 207 relevant facts and 246 non-relevant facts across all 17 queries, an average of 26.6 relevant facts per query. In this study we only consider queries with at least five correctly extracted facts (yielding 17 queries).

8.3.2 Results

The *relevance* of a fact is separately evaluated from *extraction correctness*, as just described above in Section 8.3.1. In the following, we focus only on the 453 correctly extracted facts and present our analysis along the research questions asked above.

Table 8.1: Experimental results for relation relevance (correctly extracted relations only) comparing different fact retrieval features: All facts (All), facts also included in DBpedia (DBp), fact mentioned three or more times ($\text{Frq}_{\geq 3}$), facts extracted from a relevant document (Doc). Significance with p-value ≤ 0.05 versus "All" marked with †.

		All	$\text{Frq}_{\geq 3}$	DBp	Doc
Per Query (macro-avg)	#Queries	17	10	17	10
	Precision	0.470	0.553	0.455	0.704
	Std Error	0.070	0.100	0.087	0.112
All Facts (micro-avg)	#Retrieved Facts	453	106	145	46
	TP	207	58	64	30
	FP	246	48	81	16
	TN	-	198	165	230
	FN	-	149	143	177
	Precision	0.457	†0.547	0.441	†0.652
	Recall	1.000	†0.280	0.309	†0.145
	F_1	0.627	†0.371	0.364	†0.237
	Accuracy	0.457	†0.565	0.506	†0.574

Applicability (RQ1). We report the results on fact relevance as micro-average across all facts (Table 8.1 bottom) and aggregated macro-averages per query (Table 8.1 top) to account for differences across queries. Among all correct facts, only every other fact is relevant for the query (0.45 micro-average precision, 0.47 macro-average precision). Factoring in the extraction precision of 0.51 we obtain one relevant out of four extracted facts on average. This strongly suggest that the problem of relevant relation finding (beyond correctness) is indeed an *open research problem*.

Indicators for fact relevance (RQ2). We study different indicators that may improve the prediction of fact relevance. First, we confirm that the frequency of fact mentions indicates fact relevance. If we classify a correctly extracted fact as 'relevant' only when it is mentioned at least three times[7] then relevance accuracy is improved by 23.6% from 0.457 to 0.565 (signifi-

[7]We chose ≥ 3 in order to be above the median of the number of sentences per fact, which is 2.

Table 8.2: Fact relevance when at least one entity ($S \vee O$) or both entities ($S \wedge O$) are relevant compared to all facts (All). Significance with p-value ≤ 0.05 marked with †.

	All	S ∨ O	S ∧ O
#Retrieved Facts	108	94	49
TP	78	76	45
FP	30	18	4
TN	-	12	26
FN	-	2	33
Precision	0.722	†0.809	0.918
Recall	1.000	†0.974	0.577
F_1	0.839	†0.884	0.709
Accuracy	0.722	†0.815	0.657

cant according to a two-sided exact binomial test with $\alpha = 5\%$). This also reduces the number of predicted facts to a fourth (see Table 8.1, column $Frq_{\geq 3}$).

Next, we check if the extracted facts already exists in the DBpedia KG, following the hypothesis that everything relevant might already be contained in DBpedia. But when classifying only extracted facts as relevant when they are confirmed – that is, both entities are related in DBpedia (independent of the relation type) – we do not obtain any significant improvements in accuracy or precision. Therefore, we conclude that confirmation of a known fact in an external KB does not indicate relevance. On the contrary, we notice that only 64 of the relevant, extracted facts are already included in DBpedia, whereas the remaining 143 are new and relevant facts, extracted from our document-centric approach (cf. Table 8.1, column *DBp*). This indicates that extracting yet unknown relations (i.e., those not found in the KB) from query-relevant text has the potential to provide the majority of relevant facts to the query-specific KB.

Considering the fact that not all retrieved documents are actually relevant, we study the impact of this factor on our final fact relevance results. Not surprisingly, we can confirm that when considering only documents assessed as relevant (document relevance annotations are take from the TREC assessment data accompanying the queries) this significantly improves accuracy and precision of the relation relevance. However, it comes at the cost of retaining only a tenth of the facts (cf. Table 8.1, column *Doc*) – which is obviously a drawback in terms of coverage.

Fact relevance vs. entity relevance (RQ3). Finally, we explore whether query-relevance of entities implies query-relevance of facts, i.e. subject–predicate–object triples. For this evaluation we make use of the REWQ ClueWeb12 test collection on entity relevance (as introduced in the previous chapter, see Section 7.3.1) and study the subset of the 108 correct facts where relevance assessments exist for both entities, subject (S) and object (O). Due to pooling strategies, this subset has a higher precision of 0.722. In Table 8.2 we consider the case where entity relevance is true for both entities (S ∧ O) as well as for at least one entity ($S \vee O$).

For only 12 correct facts, both entities are assessed as non-relevant – these facts were also assessed as non-relevant by our (different) annotators. In contrast, for 45 facts both entities

and the fact itself are assessed as relevant (we take this agreement also as a confirmation of the quality of both assessment dataset, as they have been created at different points of time and by different annotators). Using the entity assessments as a classifier, we obtain improvements in precision from 0.722 to 0.809 for either entity and 0.918 for both entities. While also accuracy improves for the case of either entity, it is actually much lower in the case of both entities. We conclude that the restriction to both entities being relevant misses 33 out of 78 relevant facts.

When further inspecting this set of 33 relevant facts with one relevant and one non-relevant entity, we find that the non-relevant entity is often rather unspecific or general in nature, being e.g a country or city. It makes sense that such generic entities are, when assessed in isolation, are considered being too generic and thus annotated as not relevant w.r.t the query – maybe also because the annotators are not aware of the relationship between the entity and the query topic. For example, in Figure 8.1 the `University_of_Cambridge` is only relevant for the query "Raspberry Pi" because its inventor `Eben_Upton` is a member of Cambridge. Thus, we conclude that fact relevance and entity relevance are not the same, and that facts seem to be the more expressive and appropriate information unit.

8.4 Related Work

The task of creating query-specific KB is rather specific and no commonly accepted definitions as provided e.g. by challenges or competitions exist. Having not presented a method for a common problem, an in-depth method comparison like we presented in some of the previous chapters is not possible here. Instead we refer to some related recent works that also operate on the interplay of text and KBs.

Voskarides et al. (2015) try to explaining relationships between entity pairs in a KG and provide a natural language, i.e. human readable, description of the relationship. They propose a setting similar to ours, where a corpus of text document is linked to the KG via the entities found within the text. However, they start from the given KG facts, while our starting point are the query-specific document – regardless of the relationships available from the KG. Voskarides et al. also generate candidate sentences by selecting those which contain an entity pair. The selection of the best explaining sentences is then modeled as a LTR problem, which a large set of features describing (i) the text itself (sentence length, token idf weights from Wikipedia, etc.), (ii) the entity and its KG attributes (entity count, direct entity KG links, distance between entity mentions, entity relatedness), (iii) the relationship (different features matching surface form to KG predicate using WordNet and different distributional semantics methods), and (iv) source features (position of sentence in document, etc.). The authors report that their method significantly improves over state-of-the-art baseline models. The main difference to our approach, besides the different motivation and the aim, is that we include also yet unknown facts from documents.

The work from Blanco and Zaragoza (2010) is closely related to Voskarides et al. (2015), as they also try to explain entity relationships, but in this case between a named entity and an ad-hoc query. Their aim is to return the user a natural language sentence, called entity support

sentences, that explains the relationship. The retrieval is based on different entity score-based, position-based, and retrieval-based features, however, the most important features turned out to be the context of a sentences. The authors report also that – not surprisingly – traditional BoW models perform well if query and entity match on a syntactic surface level, but fail for a substantial portion of entities. In contrast, Voskarides et al. look the KG relations between entities, and we look at relations between entities within the same document (even if no KG relationship exists).

The more general task of construction KBs from text (documents) is rather a rather well-studied research area. Systems for extracting facts without adhering to a predefined relation schema, as otherwise would be taken e.g. from the target KB to be populated/completed, are known as open information extraction (OIE) systems. Well-known pioneer work in this area are the NELL (Carlson et al., 2010) and the Reverb (Fader et al., 2011) systems, see also Chapter 6.

8.5 Conclusion

In this last chapter, we investigated the idea of extracting query relevant facts from text documents to create query-specific KBs. It represents the contentious development of our motivation to explore the KG, as, in contrast to the previous chapters, relations from text and from the KB are combined. Because this chapter's intention was more to describe the state-of-the-art of relations extraction and to point to future research opportunities then to solve an established problem, in the following we will not only conclude about our findings, but also describe possible future extension of our work, and also possible future applications for query-specific KBs.

8.5.1 Conclusion

Our study combines publicly available data sets and state-of-the-art systems for document retrieval and relation extraction to answer research questions on the interplay between relevant documents and relational facts for this task. We can summarize our key findings as follows:

(a) Query-specific documents contain relevant facts, but even with perfect extractions, only around half of the facts are actually relevant with respect to the query.

(b) Many relevant facts are not contained in the DBpedia KG, suggesting the importance of extraction for query-specific KBs.

(c) Improving retrieval precision of documents increases the ratio of relevant facts significantly, but sufficient recall is required for appropriate coverage.

(d) Facts that are relevant can contain entities (typically in object position) that are – by themselves – not directly relevant.

From a practical perspective, we conclude that the combination of document retrieval and relation extraction is a suitable approach to query-driven knowledge base construction, but it remains an open research problem.

8.5.2 Future Work

The next obvious step would be to extend our descriptive work into an automatic fact extraction system that generates a significantly higher rate of relevant facts, e.g., by investigating joint models of relation extraction and passage retrieval. For further advances in the relation extraction method, we recommend to explore the potential of integrating document retrieval and relation extraction – as opposed to simply applying them sequentially in the pipeline architecture.

Another improvement might yield from the usage of an OIE system, in contrast to the closed system, i.e. RelationFactory, we used. While on the one hand the precision of the extracted relations from OIE might be lower because the relation/predicate schema is not predefined, on the other hand we would gain a much higher coverage, in particular for those facts not yet contained in an existing KG, when using an unrestricted relation schema.

But as mentioned in the introduction of this chapter, with OIE we face the challenge of how to integrate the OIE facts with the KB facts – because in the end we want to combine all information into one common knowledge base. Note that this is exactly the problem from Dutta et al. (2013) we already discussed, and partially addressed, in Chapter 6: How to come from a OIE (Nell) fact, e.g.

"studiedAt"("Eben Upton", "Cambridge")

to its KB (DBpedia) triple, e.g.

`db:Eben_Upton dbo:almaMater db:University_of_Cambridge .`

This task is in particular challenging for the predicate matching, as the semantics of the relation have to be considered, i.e. is "studiedAt" equivalent to `dbo:almaMater` or do they describe different types of relation between the entities. A solution to the property matching problem was proposed by Dutta et al. (2015), who cluster Nell predicates together before attempting to match them to existing DBpedia predicates. This seems to be an important step towards our vision, and a further integration of such a predicate matching is recommended.

Besides the enhancements on the knowledge integration, for improvements on the problem understanding are also necessary. An important next steps would be to perform an additional study on different types of queries, such as entity vs. complex queries, in order to study their influence on our experimental results. It seems possible that we would need different approaches for our query-specific KG generation, depending on the type of query, in particular if it is an entity query or not.

8.5.3 Future Applications

Turning to future applications, we see the construction of query-specific KBs as an important input for different high-end applications, in particular in Web IR.

First, personalized or topic-specific KB construction (or maybe completion, on top of existing KBs) seems to be an interesting future application. As we discovered in our experiments, many relations that are relevant to a specific query were not contained in DBpedia, which can have two reasons: (i) the KB is incomplete, and should be extended with the missing facts, or (ii) the facts are query-relevant, but too specific – or for other legitimate reasons not suitable – to be added to a general purpose encyclopedia like Wikipedia and DBpedia are. In the latter case, topic specific KBs, that are automatically created from a given user query as proposed e.g. from Dalton and Dietz (2013b), would solve this conflict between an individual user's information need and the standards of a general purpose KB. Structured KB-based search results, like e.g. the entity info boxes about persons provided by search engines like Google, Yahoo, or Bing, on the right hand side of the screen, could thus be adapted to the search query, but also personalized to the individual user.

Another interesting end-user application would be the automatic construction of query-specific, *human-readable* KBs like Wikipedia. This idea was proposed e.g. by Dietz et al. (2014); Dietz and Schuhmacher (2015) and by Sauper and Barzilay (2009), who both aim at an automatically created Wikipedia article. Dietz et al. envision a system that automatically generates a human-readable text document, structured and organized just like a Wikipedia article, that compiles all relevant information about the user-defined information need. Our query-specific KB could be a first step into such an entity-centric information aggregation system.

However, the question remains how one would generate a coherent text from a structured KG – even when this KG would contain all relevant facts. One method would be to retrieve natural language sentences, and use the KG information just as a means to the sentences selection (like Dietz et al. suggest). Another approach would be to further explore natural language generation, and generate the article text directly from the KG facts. Recent work on natural language generation has already focused on RDF KGs (Cimiano et al., 2013), and even on generating text specifically from the DBpedia KG (Unger et al., 2013). Nevertheless, there is still a way to go from sentences to coherent text articles. It thus remains to be decided by future research, what the best way will be for making structured knowledge accessible for human users.

Coming back the above stated reasons for incomplete KBs, Fetahu et al. (2015) proposed a technique for adding missing facts about events to Wikipedia articles, thus in the long term, also aiming at the generation of Wikipedia articles. Starting from news articles, entities mentioned within the text are identified, and based on a rich feature set (including entity salience, relative authority and novelty of the article), those entities which should mention the information from the news article at hand are identified. In addition, the target structure of the article is also considered. While aiming, from a conceptual point of view, for a different aim, namely the competition of the Wikipedia knowledge base, the technical problems of creating a human-readable Wikipedia article are similar.

While mentioned applications will most likely not be realized in the near future, an application which is likely to become reality soon are web search interfaces that combine KB and free text search into a single integrated search experience. A prototype pointing towards such a systems was developed by Hoffart et al. (2014), who depict the combination of entity and free text search,

also allowing for refining the search by entity attributes (categories).

Last, we only want to briefly note that facts by themselves, as collected by current KGs, will in the long term not be enough to satisfy web search users. Users are often interested in opinions instead of facts, or in facts being put in context of an option, e.g. in political debates. In addition, the boarder between an "objective fact" and an "opinionated statement" is of course in reality broad and blurry, as soon as we go beyond simple facts like a person's date of birth. [8] Thus, not only facts, but also opinions will, somehow, have to find their way into the applications described above.

[8] Which is why e.g. the Wikidata project allows conflicting facts and introduced fact provenance data for the information source (Vrandečić and Krötzsch, 2014).

Chapter 9

Thesis Conclusion

In this thesis, we studied the potential of knowledge graphs (KGs) like DBpedia for various text understanding problems in the area of natural language processing (NLP) and information retrieval (IR). Our encompassing hypothesis throughout all chapters was thereby, that applying entity linking (EL) to text to obtain knowledge base (KB) entities has the potential to bring the information from text together with the information available from general-purpose KGs, in particular DBpedia, and thus improve the understanding task at hand. In each chapter, we looked at this integration of text and KB information from a different angle and for a different task, while, form chapter to chapter, increasing the degree of KG exploration and/or the degree of integration of text and KG information. In the end, the key contribution of this thesis is that it fosters our understanding of the role and the potential of state-of-the-art KGs in the interaction of text, entities, knowledge bases, and knowledge graphs – it provides the reader with reasonable hints for the question: what to do with all this knowledge?

9.1 Part I – Using Knowledge Base Entities

We started with the hypothesis that adding background knowledge to a short sentence fragment (search result snippet) should improve the clustering of those texts into semantically coherent clusters. The given problem was that short text documents, the search results snippets from Chapter 3, do not have sufficient syntactic, i.e. word or token, overlap to be compared and clustered easily. This problem was the test case for our proposed pipeline of (i) taking a text document, (ii) extracting KB entities (here Wikipedia) via EL, and (iii) retrieving additional background knowledge from a KG (here DBpedia). For Chapter 3 this means specifically, that we obtained DBpedia types, like `dbo:MusicalArtist`, and categories, like `dbc:American_folk_rock_musicians` and added them as additional features into the text clustering. The categories and types provided thus primarily topical information about the found entities, which was helpful for the given setting of sense disambiguation of search results, e.g. between Apache the helicopter and Apache the software project for the query “apache”. On the benchmark dataset, we found this approach to yield competitive, while not top performance, results. A limitation of this part was the only partial exploration of the KG relations, which motivated the extension described in Part II.

9.2 Part II – Using the Knowledge Graph for Understanding

In Part II, we explored two main ideas: (i) to compute entity relatedness using the DBpedia KG, and (ii) to represent and compare documents using the DBpedia KG.

The natural extension of the previous part is to leverage the full KG, and not only entity types and categories. Thus, instead of testing if two entities share the same category, we aimed at understanding if there is any relatedness between two entities at all. One intuitive solution is to compute shortest path in the KG between entities, but we found this to be a too naive approach, as there are too many relations (and thus KB paths). For that reason, in Chapter 4 we proposed different unsupervised, information-theoretic weighting schemata for the KG. Based on a benchmark dataset for entity ranking, we found that these schemata can help to select the meaningful KG paths for computing entity relatedness (with *combIC* being the best measure). Even though this approach is not able to capture all semantic information expressed by the KG predicates, it is a robust, KG vocabulary agnostic, and unsupervised approach applicable to any RDF KG. In Chapter 6, we demonstrated that our entity relatedness measure also helps to improve EL. In the chapter's specific setting where the subject and objects from an open information extraction (OIE) system (NELL) have to be disambiguated to DBpedia entities, the semantic relatedness between subject and object complements the statistical information about the most frequent sense and thus improves the disambiguation accuracy. This finding is in line with later work from Hulpuş et al. (2015), who evaluated entity relatedness measures and found our *combIC* metric to be the best KG-based measure for entity disambiguation.

Building upon the means to compare single entities, we proposed in Chapter 5 a method to compute semantic similarity of documents: (i) represent a document by the entities extracted from it, (ii) compute document similarity as an entity subgraph matching problem in the DBpedia KG using graph edit distance (GED). Our method has the advantage of providing a computer-readable as well as a human-readable document modeling, in contrast to e.g. continues vector representations of words, that incorporates explicitly external background knowledge by not integrating KB information into the document, but vice versa, representing the document as a subgraph of a KG (here DBpedia). For the graph matching, the edge-weighted KG paths between the entities are utilized as edit distance cost (in the GED), thus capturing the intuition that two documents are similar if they talk about many related entities. The experimental evaluation on a standard benchmarking dataset (LP50) show that we achieve competitive performance better than or close to well-known methods like LSA or ESA. However, we cannot beat methods that build upon a much more feature-rich KB exploration and apply machine learning for feature combination. Nevertheless, we understand our model, also because it got adapted and improved by other researchers, as an important and interesting contribution for knowledge-based text comparison methods.

9.3 Part III – Using the Knowledge Graph for Relevance Ranking

The last part shifts the focus of the applications from text understanding in Part I and II to relevance ranking, thus taking more of an IR approach where the fulfillment of a user's information

need, expressed by a keyword query, is in focus. In Chapter 7, we defined a task that combines, in the same spirit as above, document retrieval with entity retrieval: Given a keyword query, return not only a document ranking, but also a ranking of the entities found within those documents. Due to the novelty of the task and the lack of existing benchmarks, we created two benchmarking dataset, both building upon long standing datasets for document retrieval. We study extensively different query, document, entity, and KB features for ranking, including an SVM-based semantic kernel to capture KB relations between entities. Our findings on both datasets show consistently that (i) the retrieved documents contain query relevant entities, and (ii) that for ranking those entities, information from the document (the *tf-idf* mention frequency) and from the background KB (Wikipedia articles) are important features. However, we cannot confirm a significant improvement from the structural DBpedia KG features that capture relatedness to entities in the query: It appears that the direct query-to-Wikipedia retrieval feature is already a strong ranking signal here.

In the last Chapter 8, we extend the idea of retrieving entities to relations, i.e. finding relevant subject–predicate–object triples for a given query in a document. It is a natural extension of the previous chapter's work, as, in contrast those chapters, not only entities are the binding element between text and KB, but fact statements consisting of predicates and entities. As this work is only a first step towards query-specific KG construction, we created a new evaluation dataset based on established IR datasets and ran a pipeline of document retrieval, followed by a relation extraction on the retrieved documents. We find that (i) query-relevant documents contain query-relevant facts only at a medium precision, (ii) relations extracted from the documents and the DBpedia KG complement each other, (iii) that entities in relevant fact triples are not necessarily relevant by themselves. This chapter leaves open questions for further research regarding both method improvements for the KB construction and for future end-user applications for query-specific KGs.

9.4 Open Issues and Limitations

Leaving the individual, task-specific limitations we already discussed at the end of each chapter behind, we discuss here the overall perspective on this thesis.

All our work tried to combine information from text with information from knowledge bases (KBs). The triangle of information, composed of (a) text, (b) the semi-structured KB Wikipedia, and (c) the KG DBpedia, was thereby the source of our information, but we never created a tight integration into one model. It became clear by our experiments, that exploiting the DBpedia KG is only one ingredient needed: Ignoring the high-coverage information from the text, as we did in Chapter 5 for example, is a clear limitation.

Thus, the missing integration of the facts from all three types of data sources seems consequently to be the most sever limitation of our work – and makes it for that reason the most interesting major extension of our work. As pointed out in Section 8.5, in particular the integration of facts extracted from text with the facts from an existing KG seems promising. Such an integration

could also increase the coverage of our KG subgraphs from Chapter 5: Instead of projecting a document into the KG, a document would be a fusion of the facts extracted from the document together with the KG's background knowledge.

Another notable limitation of our work is due to the limited expressivity of the KGs we worked with. The DBpedia KG, like all publicity available KGs, does not provide any information about the time, i.e. at what point in time a given fact is true. For example, the fact that Brad Pitt is married to Jennifer Anderson is wrong as of today, but it was true some years ago. In addition, when combining information from text and KG, they can contradict each other. We did not consider these problems as we did not make use of the semantics of the KGs – which is actually also a fundamental limitation of our work: We treat the KG only as a graph, thus deliberately ignoring any semantics of the predicates and thus any (potentially possible) reasoning. While this might be acceptable for today, given the low expressivity of DBpedia KG that contains only a few terminological axioms, this might change in the future when KGs become more complex and adapt more expressive languages like e.g. OWL.

9.5 Future Research

In a broader context, we view our work as one of many contributions in IR and NLP that study the combination of structured and unstructured information (for different tasks). Historically, when looking back on the field of KB exploitation, the idea to use Wikipedia (for various tasks) was a significant step for the research community (cf. Hovy et al., 2013). In the light of this legacy, we see future research going forward into two different directions, leaving Wikipedia as KB behind: On the one hand side, more structured KGs, with increasing levels of expressivity beyond RDF, are about to become the dominant knowledge representation form for many Web IR problems – cf. the knowledge graphs (KGs) created by web search engines or the Wikidata project. On the other hand side, much effort has been invested into filling these KGs with information, i.e. facts, currently only available from unstructured resources, most often (Web) text document. Information extraction, open and closed, will continue to be an important area of innovation – together with the many (often commercial) efforts that integrate humans in the extraction process to encode complex knowledge that cannot be extracted automatically yet, at least not with a high enough precision. An interesting exception is here the Wikidata project, which does not aim at information extraction, but to create a KG (also) editable by humans and that serves facts data into the (human-readable) Wikipedia articles.[1]

Future KGs will combine all these information, and thus increase the need to answer the question how to handle contradicting KB information. While maintaining provenance information about the origin of an KB fact is an important foundation,[2] introducing uncertainty seems a promising approach. Uncertainty information at the fact level allows for probabilistic reasoning, which can be a way to overcome the currently rather limited expressivity of Web KGs, in which often the rather imperfect data quality prevents the meaningful usage of reasoning: One incorrect fact in

[1] Cf. Vrandečić and Krötzsch (2014) and `https://www.wikidata.org`.

[2] Which is getting addressed for hat reason by the Wikidata project.

a non-probabilistic ontology makes reasoning impossible.

In the medium term, the increasing popularity of knowledge graphs (KGs) for Web search will foster a renaissance of the Semantic Web vision: A web, in which terms have semantics attached and reasoning over different data sources is possible. It seems that entities, from KB as well as from text, will continue to play an important role in this setting, not only because they form the connections between different data sources, but also because many user information needs involve entities. This need for entities and relations will also continue to increase the importance of information extraction, which itself relies heavily on high-performing NLP methods. The Wikidata project clearly points towards this KG-centric future, as it aims at creating a fully machine-readable KB at very high quality with the help of humans, just like Wikipedia. Such developments for enlarging the machine-readable Web will make a true Semantic Search on the Web possible at some point in the future, and classical document search will be combined and supported by entity search and knowledge graph (KG) retrieval.

We believe that these developments will continue to shift the attention of industry and research from the document-centric information processing towards a facts or KG-based perspective in which information will no longer be a collection of natural language text (documents) – but instead be handled at the level of (single or connected) facts. In the long-term perspective, however, when natural language generation from such fact knowledge bases will work sufficiently well, the users will no longer interact directly with neither documents nor facts, but only ask questions that an advanced search-engine interface will answer, while the complexity of KG generation, information processing, reasoning, and question answering will be hidden.

Bibliography

Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A., and Guo, W. (2013). *SEM 2013 shared task: Semantic Textual Similarity. In *Proc. of *SEM-2013*, pages 32–43.

Agirre, E., Diab, M., Cer, D., and Gonzalez-Agirre, A. (2012). Semeval-2012 Task 6: A pilot on semantic textual similarity. In *Proc. of *SEM-2012*, pages 385–393.

Balog, K., Azzopardi, L., and de Rijke, M. (2006). Formal Models for Expert Finding in Enterprise Corpora. In *Proc. of SIGIR-06*, pages 43–50.

Balog, K., de Vries, A. P., Serdyukov, P., Thomas, P., and Westerveld, T. (2010). Overview of the TREC 2009 Entity Track. In *Proc. of TREC-09*.

Balog, K. and Serdyukov, P. (2011). Overview of the TREC 2011 Entity Track. Available from `http://krisztianbalog.com/files/trec2011-entity-overview.pdf`.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction for the web. In *Proc. of IJCAI'07*, pages 2670–2676.

Bär, D., Biemann, C., Gurevych, I., and Zesch, T. (2012). UKP: computing semantic textual similarity by combining multiple content similarity measures. In *Proc. of SemEval-2012*, pages 435–440. ACL.

Bär, D., Zesch, T., and Gurevych, I. (2011). A Reflective View on Text Similarity. In *Proc. of RANLP-11*, pages 515–520.

Bellot, P., Doucet, A., Geva, S., Gurajada, S., Kamps, J., Kazai, G., Koolen, M., Mishra, A., Moriceau, V., Mothe, J., Preminger, M., SanJuan, E., Schenkel, R., Tannier, X., Theobald, M., Trappett, M., and Wang, Q. (2013). Overview of INEX 2013. In *Proc. of CLEF'13*, volume 8138 of *LNCS*, pages 269–281.

Biemann, C. and Riedl, M. (2013). Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1:55–95.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

Bizer, C., Eckert, K., Meusel, R., Mühleisen, H., Schuhmacher, M., and Völker, J. (2013). Deployment of RDFa, Microdata, and Microformats on the Web – A Quantitative Analysis. In *Proc. of ISWC'13*, pages 17–32.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. (2009). DBpedia – A Crystallization Point for the Web of Data. *Journal of Web Semantics*, 7(3).

Blanco, R., Ottaviano, G., and Meij, E. (2015). Fast and Space-Efficient Entity Linking in Queries. In *Proc. of WSDM'15*, pages 179–188.

Blanco, R. and Zaragoza, H. (2010). Finding support sentences for entities. In *Proc. of SIGIR-10*, pages 339–346.

Bloehdorn, S., Basili, R., Cammisa, M., and Moschitti, A. (2006). Semantic Kernels for Text Classification based on Topological Measures of Feature Similarity. In *Proc. of ICDM'06*, pages 808–812.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. of SIGMOD'08*, pages 1247–1250.

Booth, J., Eugenio, B. D., Cruz, I. F., and Wolfson, O. (2009). Query Sentences as Semantic (Sub) Networks. In *Proc. of ICSC-09*, pages 89–94.

Bruni, E., Uijlings, J., Baroni, M., and Sebe, N. (2012). Distributional semantics with eyes: Using image analysis to improve computational representations of word meaning. In *Proc. of MM'12*, pages 1219–1228.

Burges, C. J. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Campinas, S., Ceccarelli, D., Perry, T. E., Delbru, R., Balog, K., and Tummarello, G. (2011). The Sindice-2011 dataset for entity-oriented search in the web of data. In *Proceedings of the 1st International Workshop on Entity-Oriented Search (EOS)*, pages 26–32.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Proc. of AAAI'10*, pages 1306–1313.

Carpineto, C., Osiński, S., Romano, G., and Weiss, D. (2009). A survey of Web clustering engines. *ACM Computing Surveys*, 41:17:1–17:38.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Ciglan, M., Nørvåg, K., and Hluchý, L. (2012). The SemSets Model for Ad-hoc Semantic List Search. In *Proc. of WWW-12*, pages 131–140.

Cimiano, P., Lüker, J., Nagel, D., and Unger, C. (2013). Exploiting ontology lexica for generating natural language texts from RDF data. In *Proc. of European Workshop on Natural Language Generation (ENLG'13)*, pages 10–19.

Committee, T. K. E. O. (2016). Entity Discovery and Linking Task Description. Task Definition Ver 1.0 of March 9, 2016, NIST. `http://nlp.cs.rpi.edu/kbp/2016/taskspec.pdf`.

Cormen, T. H. (2009). *Introduction to algorithms*. MIT Press, Cambridge, Mass., 3rd edition.

Cornolti, M., Ferragina, P., and Ciaramita, M. (2013). A framework for benchmarking entity-annotation systems. In *Proc. of WWW-13*, pages 249–260.

Dalton, J. and Dietz, L. (2013a). A Neighborhood Relevance Model for Entity Linking. In *Proc. of OAIR-2013*, pages 149–156.

Dalton, J. and Dietz, L. (2013b). Constructing query-specific knowledge bases. In *Proc. of AKBC'13*, pages 55–60.

Dalton, J., Dietz, L., and Allan, J. (2014). Entity Query Feature Expansion Using Knowledge Base Links. In *Proc. of SIGIR'14*, pages 365–374.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Demartini, G., Iofciu, T., and de Vries, A. P. (2010). Overview of the INEX 2009 Entity Ranking Track. In *Proc. of INEX 2009*, volume 6203 of *LNCS*, pages 254–264. Springer-Verlag.

Dietz, L. and Schuhmacher, M. (2015). An Interface Sketch for Queripidia: Query-driven Knowledge Portfolios from the Web. In *Proc. of ESAIR'15 Workshop*, pages 43–46.

Dietz, L., Schuhmacher, M., and Ponzetto, S. (2014). Queripidia: Query-specific Wikipedia Construction. In *Proc. of AKBC'14*, pages 1–5. Available from `http://www.akbc.ws/2014/`.

Dunietz, J. and Gillick, D. (2014). A New Entity Salience Task with Millions of Training Examples. In *Proc. of EACL-2014 Short Papers*, pages 205–209. ACL.

Dutta, A., Meilicke, C., Niepert, M., and Ponzetto, S. (2013). Integrating open and closed information extraction: Challenges and first steps. In *Proc. of Intl. Workshop on NLP & DBpedia at ISWC'13*, pages 50–61. Available from `http://ceur-ws.org`.

Dutta, A., Meilicke, C., and Stuckenschmidt, H. (2015). Enriching structured knowledge with open information. In *Proc. of WWW'15*, pages 267–277.

Dutta, A. and Schuhmacher, M. (2014). Entity Linking for Open Information Extraction. In *Proc. of NLDB'14*, pages 75–80.

Eckert, K. (2012). *Usage-driven Maintenance of Knowledge Organization Systems*. PhD thesis, Universität Mannheim.

Egozi, O., Markovitch, S., and Gabrilovich, E. (2011). Concept-Based Information Retrieval using Explicit Semantic Analysis. *ACM Transactions on Information Systems*, 29(2):8:1–8:34.

Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., and Vrandečić, D. (2014). Introducing Wikidata to the linked data web. In *Proc. of ISWC'14*, pages 50–65.

Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. (2004). Web-scale information extraction in KnowItAll (Preliminary results). In *Proc. of WWW'04*, pages 100–110.

Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying Relations for Open Information Extraction. In *Proc. of EMNLP-11*, pages 1535–1545.

Fellbaum, C., editor (1999). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass, 2nd edition.

Ferragina, P. and Scaiella, U. (2012). Fast and Accurate Annotation of Short Texts with Wikipedia Pages. *IEEE Software*, 29(1):70–75.

Fetahu, B., Markert, K., and Anand, A. (2015). Automated News Suggestions for Populating Wikipedia Entity Pages. In *Proc. of CIKM-15*, pages 323–332.

Frey, B. J. and Dueck, D. (2007). Clustering by Passing Messages Between Data Points. *Science*, 315:972–976.

Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. (1987). The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971.

Gabrilovich, E. and Markovitch, S. (2006). Overcoming the Brittleness Bottleneck using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. In *Proc. of AAAI-06*, pages 1301–1306.

Gabrilovich, E. and Markovitch, S. (2007). Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proc. of IJCAI-07*, pages 1606–1611.

Gabrilovich, E., Ringgaard, M., and Subramanya, A. (2013). FACC1: Freebase annotation of ClueWeb corpora, Version 1. `http://lemurproject.org/clueweb12/FACC1/`.

Gao, X., Xiao, B., Tao, D., and Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113–129.

Gärtner, T. (2003). A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58.

Glavaš, G. and Šnajder, J. (2013). Recognizing Identical Events with Graph Kernels. In *Proc. of ACL'13*, pages 797–803.

Guo, S., Chang, M.-W., and Kiciman, E. (2013). To Link or Not to Link? A Study on End-to-End Tweet Entity Linking. In *Proc. of HLT-NAACL'13*, pages 1020–1030.

Gurajada, S., Kamps, J., Mishra, A., Schenkel, R., Theobald, M., and Wang, Q. (2013). Overview of the INEX 2013 linked data track. In *CLEF Working Notes*. Available from `http://ceur-ws.org`.

Hachey, B., Radford, W., Nothman, J., Honnibal, M., and Curran, J. R. (2013). Evaluating entity linking with Wikipedia. *Artificial intelligence*, 194:130–150.

Hassan, S. and Mihalcea, R. (2011). Semantic Relatedness Using Salient Semantic Analysis. In *Proc. of AAAI-11*, pages 884–889.

Hayes, P. and Patel-Schneider, P. (2014). RDF 1.1 Semantics. W3C recommendation, W3C. `http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/`.

Hees, J., Khamis, M., Biedert, R., Abdennadher, S., and Dengel, A. (2013). Collecting Links between Entities Ranked by Human Association Strengths. In *Proc. of ESWC-13*, pages 517–531.

Hobbs, J. R. (1985). Ontological Promiscuity. In *Proc. of ACL'85*, pages 60–69.

Hoffart, J., Milchevski, D., and Weikum, G. (2014). STICS: Searching with Strings, Things, and Cats. In *Proc. of SIGIR-14*, pages 1247–1248.

Hoffart, J., Seufert, S., Nguyen, D. B., Theobald, M., and Weikum, G. (2012). KORE: Keyphrase Overlap Relatedness for Entity Disambiguation. In *Proc. of CIKM-12*, pages 545–554.

Hoffart, J., Suchanek, F. M., Berberich, K., and Weikum, G. (2013). YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust Disambiguation of Named Entities in Text. In *Proc. of EMNLP'11*, pages 782–792.

Hovy, E., Navigli, R., and Ponzetto, S. P. (2013). Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27.

Huang, L., Milne, D., Frank, E., and Witten, I. H. (2012). Learning a concept-based document similarity measure. *Journal of the American Society for Information Science and Technology*, 63(8):1593–1608.

Hulpus, I., Hayes, C., Karnstedt, M., and Greene, D. (2013). Unsupervised graph-based topic labelling using DBpedia. In *Proc. of WSDM-13*, pages 465–474.

Hulpuş, I., Prangnawarat, N., and Hayes, C. (2015). Path-Based Semantic Relatedness on Linked Data and Its Use to Word and Entity Disambiguation. In *Proc. of ISWC'15*, pages 442–457.

Järvelin, K. and Kekäläinen, J. (2000). IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proc. of SIGIR-00*, pages 41–48.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Ji, H. and Grishman, R. (2011). Knowledge Base Population: Successful Approaches and Challenges. In *Proc. of ACL-11*, pages 1148–1158.

Ji, H., Grishman, R., Dang, H. T., Griffitt, K., and Ellis, J. (2010). Overview of the TAC 2010 knowledge base population track. In *Proc. of TAC '10*, pages 3–3.

Joachims, T. (2002a). *Learning to Classify Text Using Support Vector Machines*. Springer Science + Business Media, New York, USA.

Joachims, T. (2002b). Optimizing search engines using clickthrough data. In *Proc. of SIGKDD-2002*, pages 133–142.

Joachims, T. (2006). Training Linear SVMs in Linear Time. In *Proc. of SIGKDD-2006*, pages 217–226.

Jong, Y. L. S.-H. N. and Lee, H. (2008). Search Result Clustering Using Label Language Model. In *Proc. of IJCNLP'08*, pages 637–642.

Kamps, J. and Koolen, M. (2009). Is Wikipedia Link Structure Different? In *Proc. of WSDM'09*, pages 232–241.

Kaptein, R. and Kamps, J. (2013). Exploiting the category structure of Wikipedia for entity ranking. *Artificial Intelligence*, 194:111–129.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Lancaster, F. W. (1972). *Vocabulary Control for Information Retrieval*. Information Resources Press.

Landauer, T. K. and Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Lau, J. H., Cook, P., and Baldwin, T. (2013). unimelb: Topic Modelling-based Word Sense Induction for Web Snippet Clustering. In *Proc. of SemEval-13*, pages 217–221.

Lee, M. D., Pincombe, B., and Welsh, M. (2005). An Empirical Evaluation of Models of Text Document Similarity. In *Proc. of CogSci 2005*, pages 1254–1259.

Levenshtein, V. I. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17.

Li, H. (2011). *Learning to Rank for Information Retrieval and Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publ.

Liu, T.-Y. (2011). *Learning to rank for information retrieval*. Springer-Verlag, Berlin.

Liu, Y., Li, W., Lin, Y., and Jing, L. (2008). Spectral geometry for simultaneously clustering and ranking query search results. In *Proc. of SIGIR '08*, pages 539–546.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Meij, E., Bron, M., Hollink, L., Huurnink, B., and de Rijke, M. (2009). Learning Semantic Query Suggestions. In *Proc. of ISWC'09*, pages 424–440.

Meij, E., Weerkamp, W., and de Rijke, M. (2012). Adding semantics to microblog posts. In *Proc. of WSDM'12*, pages 563–572.

Mendes, P. N., Jakob, M., and Bizer, C. (2012). DBpedia for NLP: A Multilingual Cross-domain Knowledge Base. In *Proc. of LREC-12*.

Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). DBpedia spotlight: shedding light on the web of documents. In *Proc. of I-Semantics'11*, pages 1–8.

Metzler, D. and Croft, B. W. (2007). Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274.

Metzler, D. and Croft, W. B. (2005). A Markov random field model for term dependencies. In *Proc. of SIGIR-05*, pages 472–479.

Mihalcea, R. and Csomai, A. (2007). Wikify! Linking documents to encyclopedic knowledge. In *Proc. of CIKM'07*, pages 233–241.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic Regularities in Continuous Space Word Representations. In *Proc. of NAACL-HLT-2013*, pages 746–751. ALC.

Milne, D. and Witten, I. H. (2008a). An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proc. of AAAI-08 Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy*, pages 25–30.

Milne, D. and Witten, I. H. (2008b). Learning to Link with Wikipedia. In *Proc. of CIKM'08*, pages 509–518.

Moschitti, A. (2006). Efficient convolution kernels for dependency and constituent syntactic trees. In *Proc. of ECML'06*, pages 318–329. Springer.

Nastase, V. (2008). Topic-driven multi-document summarization with encyclopedic knowledge and activation spreading. In *Proc. of EMNLP'08*, pages 763–772.

Nastase, V. and Strube, M. (2012). Transforming Wikipedia into a large scale multilingual concept network. *Artificial Intelligence*, pages 62–85.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10.

Navigli, R. and Di Marco, A. (2013). Clustering and Diversifying Web Search Results with Graph-Based Word Sense Induction. *Computational Linguistics*, 39(3).

Navigli, R. and Ponzetto, S. P. (2012). BabelNet: the Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193:217–250.

Navigli, R. and Vannella, D. (2013). SemEval-2013 Task 11: Evaluating Word Sense Induction & Disambiguation within an End-User Application. In *Proc. of SemEval-13*, pages 193–201.

Ni, Y., Xu, Q. K., Cao, F., Mass, Y., Sheinwald, D., Zhu, H. J., and Cao, S. S. (2016). Semantic Documents Relatedness using Concept Graph Representation. In *Proc. of WSDM'16*, pages 635–644.

Olieman, A., Azarbonyad, H., Dehghani, M., Kamps, J., and Marx, M. (2014). Entity linking by focusing DBpedia candidate entities. In *Proc. of Int'l Workshop on Entity Recognition & Disambiguation at SIGIR'14*, pages 13–24.

Olieman, A., Kamps, J., Marx, M., and Nusselder, A. (2015). A Hybrid Approach to Domain-Specific Entity Linking. *Proc. of Posters and Demos Track of SEMANTiCS'15*, pages 55–58.

Osiński, S. and Weiss, D. (2005). A Concept-Driven Algorithm for Clustering Search Results. *IEEE Intelligent Systems*, 20(3):48–54.

Passant, A. (2010). Measuring Semantic Distance on Linking Data and Using it for Resources Recommendations. In *AAAI Spring Symposium: Linked Data meets Artificial Intelligence*, volume 77, page 123.

Patwardhan, S., Banerjee, S., and Pedersen, T. (2003). Using Measures of Semantic Relatedness for Word Sense Disambiguation. In *Proc. of CICLing-03*, pages 241–257.

Paul, C., Rettinger, A., Mogadala, A., Knoblock, C. A., and Szekely, P. (2016). Efficient Graph-based Document Similarity. In *Proc. of ESWC'16*. Springer.

Pehcevski, J., Vercoustre, A.-M., and Thom, J. A. (2008). Exploiting Locality of Wikipedia Links in Entity Ranking. In *Proc. of ECIR-08*, pages 258–269.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Proc. of EMNLP-2014*. ALC.

Ponzetto, S. P. and Strube, M. (2007). Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30:181–212.

Pound, J., Mika, P., and Zaragoza, H. (2010). Ad-hoc object retrieval in the web of data. In *Proc. of WWW-10*, pages 771–780.

Quine, W. V. (1948). On what there is. *The Review of Metaphysics*, 2(1):21–38.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proc. of CoNLL'09*, pages 147–155. Association for Computational Linguistics.

Raviv, H., Carmel, D., and Kurland, O. (2012). A Ranking Framework for Entity Oriented Search Using Markov Random Fields. In *Proc. of JIWES '12*, pages 1–6.

Riesen, K. and Bunke, H. (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision and Computing*, 27(7):950–959.

Roth, B., Barth, T., Chrupała, G., Gropp, M., and Klakow, D. (2014). RelationFactory: A fast, modular and effective system for knowledge base population. In *Proc. of EACL-14*, pages 89–92.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A modern approach*. Prentice-Hall.

Sauper, C. and Barzilay, R. (2009). Automatically Generating Wikipedia Articles: A structure-aware Approach. In *Proc. of ACL'09*, pages 208–216. Association for Computational Linguistics.

Scaiella, U., Ferragina, P., Marino, A., and Ciaramita, M. (2012). Topical clustering of search results. In *Proc. of WSDM-12*, pages 223–232.

Schuhmacher, M., Dietz, L., and Paolo Ponzetto, S. (2015). Ranking Entities for Web Queries through Text and Knowledge. In *Proc. of CIKM'15*, pages 1461–1470.

Schuhmacher, M. and Ponzetto, S. P. (2013). Exploiting DBpedia for Web Search Results Clustering. In *Proc. of AKBC'13*, pages 91–96.

Schuhmacher, M. and Ponzetto, S. P. (2014a). Knowledge-based Graph Document Modeling. In *Proc. of WSDM-14*, pages 543–552.

Schuhmacher, M. and Ponzetto, S. P. (2014b). Ranking Entities in a Large Semantic Network. In *Proc. of ESWC'14 Satellite Events*, pages 254–258. Springer.

Schuhmacher, M., Roth, B., Ponzetto, S. P., and Dietz, L. (2016). Finding Relevant Relations in Relevant Documents. In *Proc. of ECIR'16*, pages 654–660.

Shen, W., Wang, J., Luo, P., and Wang, M. (2012). LINDEN: linking named entities with knowledge base via semantic knowledge. In *Proc. of WWW-12*, pages 449–458.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2008). YAGO: A Large Ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6(3):203–217.

Thalhammer, A. (2014). DBpedia PageRank dataset. `http://people.aifb.kit.edu/ath#DBpedia_PageRank`.

Tran, T., Mika, P., Wang, H., and Grobelnik, M. (2011). SemSearch'11: The 4th Semantic Search Workshop. In *Proc. of WWW'11*, pages 315–316.

Tsatsaronis, G., Varlamis, I., and Vazirgiannis, M. (2010). Text Relatedness Based on a Word Thesaurus. *Journal of Artificial Intelligence Research*, 37:1–39.

Tsikrika, T., Serdyukov, P., Rode, H., Westerveld, T., Aly, R., Hiemstra, D., and De Vries, A. P. (2008). Structured document retrieval, multimedia retrieval, and entity ranking using PF/Tijah. In *Focused Access to XML Documents*, pages 306–320. Springer.

Turney, P. D. and Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Unger, C., McCrae, J., Walter, S., Winter, S., and Cimiano, P. (2013). A lemon lexicon for DBpedia. In *Proc. of Intl. Workshop on NLP & DBpedia at ISWC'13*, pages 103–108. Available from `http://ceur-ws.org`.

Usbeck, R., Röder, M., Ngonga Ngomo, A.-C., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., et al. (2015). GERBIL: general entity annotator benchmarking framework. In *Proc. of WWW'15*, pages 1133–1143.

Voorhees, E. M. and Harman, D. K. (2005). *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press.

Voskarides, N., Meij, E., Tsagkias, M., de Rijke, M., and Weerkamp, W. (2015). Learning to Explain Entity Relationships in Knowledge Graphs. In *Proc. of ACL-15*, pages 564–574.

Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Šarić, F., Glavaš, G., Karan, M., Šnajder, J., and Dalbelo Bašić, B. (2012). TakeLab: Systems for Measuring Semantic Text Similarity. In *Proc. of SemEval-2012*, pages 441–448.

Wang, X. and Zhai, C. (2007). Learn from web search logs to organize search results. In *Proc. of SIGIR '07*, pages 87–94.

Wikipedia (2016). Bob Dylan — Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/w/index.php?title=Bob_Dylan&oldid=715945707`.

Wood, D., Lanthaler, M., and Cyganiak, R. (2014). RDF 1.1 Concepts and Abstract Syntax. W3C recommendation, W3C. `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/`.

Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proc. of ACL'94*, pages 133–138. ACL.

Yeh, E., Ramage, D., Manning, C. D., Agirre, E., and Soroa, A. (2009). WikiWalk: Random walks on Wikipedia for Semantic Relatedness. In *Proc. of TextGraphs Workshop at ACL'09*, pages 41–49.

Zhang, Z., Gentile, A. L., and Ciravegna, F. (2012). Recent advances in methods of lexical semantic relatedness – A survey. *Natural Language Engineering*, 1(1):1–69.

Zhiltsov, N. and Agichtein, E. (2013). Improving entity search over linked data by modeling latent semantics. In *Proc. of CIKM'13*, pages 1253–1256.