Entity Linking on Text and Queries

Ph.D. Thesis

October 2016

Marco Cornolti
# Contents

## 1 Introduction: Bridging the Gap between Words and Semantics

1.1 Motivating examples ........................................ 4  
1.2 Contribution of this thesis .................................. 7  
1.3 Thesis outline ............................................... 8  
1.4 Published papers ............................................ 9  
1.5 Writing style conventions ................................. 9

## 2 Background

2.1 Entities, ontologies and knowledge bases ............... 11  
2.2 Wikipedia ................................................ 12  
2.3 Piggyback: search engines as a source of information ........................................... 15

## 3 Related Work and State of the Art

3.1 State of the art for entity linking ........................... 17  
3.2 Applications of entity linking .............................. 26

## 4 A framework for entity linking

4.1 Formalizing entity linking .................................... 35  
4.2 An adaptability hierarchy for entity linking problems ........................................... 41  
4.3 Evaluation metrics for entity linking .................... 45  
4.4 Problem-specific match relations .......................... 54  
4.5 Similarity between systems ................................. 58  
4.6 GERBIL: the BAT-Framework made easy ............... 60  
4.7 GERBIL and the BAT-Framework: a preliminary evaluation ........................................... 66
# Contents

5 Evaluation of Entity Linkers
   5.1 Experimental setting ........................................ 69
   5.2 Results of evaluation ......................................... 73

II Entity linking on queries

6 Entity Linking on Queries
   6.1 The ERD Challenge ........................................... 84
   6.2 Entity linking on queries: state of the art ................... 88
   6.3 Entity linking on queries: our contribution .................. 89

7 Building a Benchmark Dataset for Queries
   7.1 Dataset construction workflow ............................... 92

8 SMAPH: a System for Entity Linking on Queries
   8.1 Candidate entity generation ............................ 101
   8.2 SMAPH-1: Individual entity pruning .................... 105
   8.3 SMAPH-S: Local entity link-back ....................... 108
   8.4 SMAPH-2: Joint entity link-back ....................... 113
   8.5 Evaluation of SMAPH ....................................... 119

Appendix A Machine Learning algorithms used by SMAPH
   A.1 Support Vector Machines (SVM) for binary classification and regression ........................................ 135
   A.2 Learning-to-rank with LambdaMART ...................... 139

References .......................................................... 143
Introduction: Bridging the Gap between Words and Semantics

Representing information is a key challenge for all applications that process and organize documents. Given the growth of digital data globally produced, and the increasing complexity of user needs when it comes to managing information, it becomes a priority to overcome traditional syntactic-based text representation and shift to a deeper understanding of documents. Making a computer understand a document, as opposed to simply analyzing its form, has been identified as a key challenge that would open an unprecedented range of new applications. For this reason, in recent years the need to shift from a purely syntactic representation of documents towards a semantic representation gained consensus and spin among academic and industrial researchers, and a big effort has been dedicated in this direction.

Thanks to the frontiers that the Web opened in terms of horizontal cooperation between users, there are examples of semi-structured knowledge bases created with the contribution of millions of people, most notably Wikipedia (Wilkinson and Huberman 2007), DBpedia (Auer et al. 2007), and Wikidata (Vrandečić and Krötzsch 2014). The information contained in these knowledge bases is extremely rich and easy to process, thanks to their open nature and well formed structure. Once the mapping between a document (being it a natural language text, an image, a video, etc.) and its semantics is done correctly, these knowledge bases can be exploited to treat information retrieval problems more deeply and accurately.

This thesis aims at being a step ahead in the task of building semantic representations of short and long text, from queries to long articles.

Recently a big effort has been put into finding a solution to the problem of detecting sequences of terms in a natural language text that mention entities and link them to the mentioned entity (Mihalcea and Csomai 2007). This process is called entity linking and is a preliminary step for building more sophisticated
solutions which aim at reconstructing the semantics of phrases or the whole document. Entities are usually drawn from a knowledge base, and each entity represents an unambiguous concept. The kind of entities covered by a knowledge base depends on the application.

Since entities are unambiguous, representing a textual document as the set of entities mentioned by it overcomes issues related to synonymy and polysemy that are inherent to natural language terms. In addition to that, entities are elements of a knowledge base that offers structured information about an entity and a graph of relations between entities. This way we can capture semantic connections to other entities and other documents. Since the ontology is generated by humans, these connections have a high accuracy. In other words, entity linking adds a layer of structured information to the original unstructured text.

A diverse set of algorithms have been proposed to do entity linking (Mihalcea and Csomai, 2007) and, despite being in a early stage of development, the systems achieve surprisingly good results and account for a significant improvement in the applications that employ this representation of texts, such as documents clustering (Scaiella et al., 2012), classification (Vitale et al., 2012), and others.

For all the problems mentioned above, it becomes of crucial importance to define abstractions like similarity of concepts, disambiguation of mentions contained in a natural language text, relevance of a topic (Milne and Witten, 2008a; Abowd et al., 1999), and to find a formal description, and a following efficient software implementation, of all these abstractions.

Considering the size of the knowledge bases at issue (Voss, 2005; Vrandečić and Krötzsch, 2014; Singhal, 2012), the tasks presented above offer important algorithmic challenges over several kinds of data types of significant size, such as labeled graphs with millions of nodes and edges.

This thesis will investigate both the theoretical aspects and the algorithm engineering issues that arise in this context.

1.1 Motivating examples

Many fields could be dramatically improved by an efficient and accurate entity annotation system. Here we cite a few of them through examples. This section has the purpose of giving the intuition behind the possible applications of entity linking, and presents approaches that are still open research problems. Refer to Section 3.2 for a review of applications that have actually been built on top of entity linkers.

Question answering. Take a human-machine dialogue in which the following question appears:
1.1. Motivating examples

Q: How old was Hendrix when he played at Woodstock?

A possible procedure to answer such a question would be to:

1. Analyse the syntactic structure of the question, and understand that it aims at finding the age of a person in a particular moment.

2. Find the entities mentioned in the question: The term Woodstock may refer to a number of entities, including the festival Woodstock, the city of Woodstock, New York, or the city of Woodstock, Ontario, but the sentence structure and context clarifies that it refers to the first. Also hendrix is a mention of the artist Jimi Hendrix. Now we know that question Q mentions two unambiguous entities: a guitar player named Jimi Hendrix and the historic event of the Woodstock music festival. These two entities get mapped to items in a knowledge base through a unique ID. For example, in Wikidata their IDs are Q164815 (Woodstock Festival) and Q5928 (Jimi Hendrix). This builds a bridge between unstructured text and a structured knowledge base.

3. A database of structured information, such as Wikidata, might include two properties related to the entities we found, namely that (i) Jimi Hendrix was born on November 27, 1942 and that (ii) the Woodstock Festival happened from August 15 to 17, 1969.

4. From these two pieces of information, a reasoning system (see e.g. Wang et al. (2004)) can infer that Jimi Hendrix was 26 when he played at Woodstock.

In this application it is crucial to correctly identify mentions of entities and link them to a knowledge base (Step 2 above).

Document semantic similarity. Finding how close are two documents from the semantic point of view is a basic task that underlies many applications (Mihalcea et al., 2006). This task has traditionally been solved by representing documents as multi-sets of words, or other syntactic features (such as collections of n-grams or similarity of the dependency graphs) or, more recently, as word or document embeddings. By considering the entities, we could increase precision and recall. For example, once the sub-graph of entities is obtained for each of the two documents, we can measure the distance of these sub-graphs according to similarity measures that take into account structured data provided by the knowledge base, including the relations among them. This approach would overcome issues related to synonymy and polysemy, e.g. by considering similar two textual documents mentioning similar concepts, even though they share a small amount of terms.
1.1. Motivating examples

Topical Categorization. The process of categorizing textual documents into a pre-defined set of topical categories depends upon the understanding of which topics are mentioned in the documents. Suppose we want to categorize two documents:

\[ D_1 = \text{Exhibition in Milan features Picasso} \]
\[ D_2 = \text{Milan ties against Barcelona 0-0} \]

into categories Sport and Culture. A categorization system that only analyzes syntactic contents of the documents (as classically done with TF-IDF (Manning et al., 2008)) would consider the two mentions of Milan as representing the same piece of information, losing information and possibly leading to an erroneous classification of both documents in the same category. An entity-annotation system would instead inform the categorization algorithm that the mention in \( D_1 \) refers to the city of Milan, while that in \( D_2 \) refers to the Milan soccer club, and this would enforce the closeness of \( D_1 \) to Culture and \( D_2 \) to Sport.

Vitale et al. (2012) is an actual algorithm that does document categorization through entity linking.

Summarization. Understanding what a document talks about is the first step for identifying the most relevant sentences in the document (Gong and Liu, 2001). For extractive summarizers\(^1\) an annotation system that assigns to the mentions a score of relevance would give important signals to select the most relevant sentences to include in the summarized text. Entity linking is even more crucial in abstractive summarization, in which it can help in the creation of highly accurate sentence patterns (Pighin et al., 2014).

Event detection. Similarly, detecting events from documents (such as news) is based on understanding what entities the text refers to. Let’s think of an application that automatically updates a knowledge base. By analyzing the sentence \( \text{Apple acquired GarageBand creators Emagic for 30 million dollars reported by a trusted source, we need to add to the knowledge base a fact that involves entities Apple Inc. and Emagic linked by the acquired property. A first step is obviously to link the term Apple to the Cupertino company rather than to the fruit.} \)

\(^1\)Summarization techniques divide in two main kinds: extractive summarization aims at finding the sub-list of terms or sentences included in the input text that contain the most important information; abstractive summarization builds a semantic representation of the document and generates a new sentence, not necessarily using the original sentence terms.
1.2 Contribution of this thesis

The main contributions provided by this thesis can be divided in two parts:

Part I. A framework for entity linking. We defined the BAT-Framework \(\text{\cite{Cor}}\)\text{\textnormal{nolti et al., 2013}}, a formal framework for entity linking, including a set of problems and a computational hierarchy among them. The framework also defines metrics to evaluate the performance of entity linking systems so that they can be compared to each other. The need for a framework in entity linking comes from the fact that the problem, in literature, has been treated by people with different backgrounds, though focusing on the same basic problem: that of understanding what a document talks about. This led to noisy results that did not give a clear suggestion about where to aim the research in this field. In this thesis, we propose a very general framework that became the de-facto standard and contributed in simplifying the research in the field. We also proposed GERBIL \(\text{\cite{Usbeck et al., 2015}}\), a user-friendly software for running the evaluations defined by the BAT-Framework.

Part II. Entity linking on queries. We then moved our focus on entity annotation of a very specific domain: that of search engine queries. This task is harder than that of annotating longer documents, because of the lower context a query offers. A query is formed by an average of 2-3 keywords with no grammatical structure. Moreover, the semantic relation between entities the keywords refer to is often either weak or not captured by knowledge bases, and this undermines the basic principle of annotators for longer text, that, in the disambiguation process, enforces semantic closeness of the entities. Considering this peculiar aspects of the domain we chose, we made two main contributions: we built and released to the public GERDAQ \(\text{\cite{Cornolti et al., 2014}}\), a benchmark dataset for entity linking on queries, and SMAPH \(\text{\cite{Cornolti et al., 2016}}\), a family of systems to solve the problem of entity linking on queries, currently constituting the state of the art performance.

We organized this thesis to keep our original contribution sharply separated from the discussion of the work of other authors. The reader will find the discussion of the background and about the state of the art, in which we cover other authors’ work published in literature, limited to Chapters 2, 3 and 6. All other chapters, unless otherwise stated, wholly consist of original contributions made by the author of this thesis, in a conjoint effort with other researchers.
1.3 Thesis outline

The thesis begins with a discussion of the background (Chapter 2), that will present a high-level overview of the works of other authors that serve as building blocks for the rest of the thesis. These methods will be presented focusing on the aspects that will be recalled in the next chapters of the thesis. Appendix A will cover the machine learning algorithms employed in the thesis.

Chapter 3 presents an overview of the contributions offered by other authors in the field of entity linking, and in particular a description of entity linking systems that have been developed, with a discussion on their internal mechanisms; an overview of the benchmark datasets for evaluating entity linkers, with a description of their features; an overview of systems and methods that apply entity linking on tasks of question answering, information retrieval, information visualization, document clustering and classification, event modeling.

The first part of the thesis will cover our contribution in terms of a framework for entity linking. Chapter 4 will present a formalization of knowledge bases, of the problems related to entity linking, and how these problems are related to each other in a hierarchy of adaptability among problems. We will also see what this means in terms of the possibility to compare two systems even though they don’t natively solve the same problem. To measure the quality of an entity linking algorithm we will present a set of general-purpose evaluation metrics for measuring their performance. We propose a set of basic measures that can be instantiated to focus on specific aspects we need to evaluate in a system. We also present GERBIL, an implementation of the BAT-Framework that makes it easy to use. Chapter 5 will give an overview of the results on entity linking systems, with a discussion on the performance of the state of the art approaches.

The second part of the thesis will cover our contribution on the task of entity linking on search engine queries. In Chapter 6 we will present the specific challenges of this problem and cover a few approaches available in the literature. In Chapter 7 we will describe the process that led to the construction of GERDAQ, a dataset for benchmarking query entity linking systems. In Chapter 8 we will present SMAPH, a family of entity linking systems specifically designed to work on queries. We will evaluate it and show that it outperforms the state of the art by significant margins.

Finally, we will present the conclusions that can be drawn from this thesis and the directions of future research.
1.4 Published papers

During his Ph.D., the author published a total of five papers. The first paper (Cornolti et al., 2013), published in the Proceedings of the 22nd International Conference on World Wide Web (WWW’13) presents the BAT-Framework. A follow-up work presenting GERBIL (Usbeck et al., 2015), has been proposed at the same conference two years later (WWW’15). Both the BAT-Framework and GERBIL will be discussed in Chapter 4.

The SMAPH-1 system for entity linking on search engine queries has been first proposed as a contribution to the ERD Challenge (ERD’14) and appears in the proceedings of its workshop, hosted by the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR’14). Two years after the challenge, we proposed important improvements that led to the development of SMAPH-2 in a paper that appeared at WWW’16. The SMAPH family of query annotators will be presented in chapters 7 and 8.

The author pursued another line of research that is not covered in this thesis, proposing a way of representing events as a cluster of sentence patterns, that was successfully employed in a system for abstractive text summarization of news. This led to a publication (Pighin et al., 2014) in the Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL’14).

For a list of papers published by the author, see the author’s bibliography at Page 155.

1.5 Writing style conventions

Throughout this thesis, we follow the following writing style conventions:

- Entities are represented by their title, in underlined italics:

  Philosophy  
  Barack Obama  
  ER (TV Series)

- If the knowledge base they are part of is not clear from the context, their whole URI is specified:

  en.dbpedia.org/Philosophy  
  en.wikipedia.org/Barack_Obama  
  wikidata.org/ER_(TV_Series)

- Knowledge base properties and relations are in bold:
Barack Obama **position held** President of the United States

Barack Obama **height** 185 centimetre

- Text from a document is rendered with a plain monospace font:

  How are you?
  
  Not bad.
2

Background

In the next chapters of this thesis, we will employ methods, tools and formalizations developed by others. Understanding them is necessary to follow the rest of the thesis: in this chapter we present the minimal background in entity linking and related fields necessary to fully understand the contents of this thesis, while in Appendix A we will review the machine learning algorithms employed in this thesis. The reader familiar with an argument may skip it, though we suggest to go through the whole chapter and the appendix.

2.1 Entities, ontologies and knowledge bases

An ontology is a system of entities, attributes, classes, and relations. The physical manifestation of an ontology is a knowledge base, namely a database providing the ontology’s information. Entities are the basic objects, they may have attributes, which are key-value features about entities. Classes are basic-type objects too. Relations represents facts involving two entities.

For example, Wikidata (that will be presented in more detail in Section 2.2.3) contains an entity called Neil Armstrong. This entity has associated, among others, an attribute with key birthDate and value 1930-08-05. The entity also appears in a relation with another entity, Neil Armstrong birthPlace Wapakoneta, Ohio. birthPlace is a relation defined as “where the person was born”, having as domain an entity of class human and as range an entity of class place. Another relation Neil Armstrong instance of human links it to the class human.

It is important to point out a subtle difference between classes in ontologies and classes in programming languages. In particular, attributes and relations linked to classes are not inherited by its instances. This lets us define attributes that refer to the class itself. For example, the class human can be in relation instance of with species, though Neil Armstrong (which is a human) is not a species.

https://www.wikidata.org/wiki/Q1615
Nonetheless, all instances of a class are also instances of its superclass. For example, given that human is a sub-class of omnivore, this implies that Neil Armstrong is an omnivore.

Each ontology has a distinct scope, i.e. has a distinct policy about the coverage of entities and relations. For example, an ontology for biology would cover all discovered living species, plus their taxonomy. Wikipedia and derived ontologies cover entities from all aforementioned domains, but only those of encyclopedic interest. For example, it covers Broadway (the road in Manhattan), but not all existing roads. Industrial knowledge bases such as the Google Knowledge Graph aims at covering as much information as possible.

Ontologies can be interconnected to each other with links from entities of one datasets to entities of other datasets. For example, the entity Neil Armstrong in Wikidata has a property Freebase ID having as value a reference to the Freebase entity about the same person.

2.2 Wikipedia

The Wikipedia online encyclopedia is probably the most significant collaborative project ever developed on the Internet. It is supported by the Wikimedia Foundation, a U.S.-based non-profit organization, and based upon the MediaWiki software.

The content of Wikipedia is edited by users (also called Wikipedians). Anyone, even with little technical expertise, can create new articles or edit their contents. The lack of a central authority may suggest that contribution from non-expert users might lead to unreliable content, but since the review process is distributed as well, involuntary or malicious mistakes are quickly found and corrected (Liu and Ram, 2011; Kittur et al., 2007). Community control over the content does not impede errors, but many studies (Giles, 2005) found that the amount of errors and bias in Wikipedia articles is comparable to that of other commercial encyclopedias edited by experts.

All versions of Wikipedia are entirely released under the Creative Commons Attribution/Share-Alike 3.0 license, that lets anyone freely distribute its contents and the derived works. Dumps of the whole encyclopedia are downloadable in open formats. This open approach makes it easy to manipulate the data and to automatically process it.

2.2.1 Wikipedia in numbers

Wikipedia comes in 283 languages. The english version of Wikipedia, the biggest of all language editions, has about 5.6 million visits per hour and counts 5.2 mil-
2.2. Wikipedia

Lion articles. The active editors (users that made at least 5 edits in a month) are 29,285 and the encyclopedia is growing at the rate of 795 new articles per day.\[^2\] Pages link to each other, and this forms a graph (where nodes are pages and directed edges are links between them). The Wikipedia graph is a small-world network, with an average shortest path from any node to another of just 4.5 steps. In the graph, there are more than 70 million edges (an average of about 17.5 outgoing edges per node).

The growth rate of Wikipedia is shown by the chart in Figure 2.1. As it can be seen, the creation of new articles started to gain momentum from 2003 on, reaching a steady level in early 2007 - late 2008. Since 2009 the creation of new articles slowed down to the current stable rate. According to a study of Suh et al. (2009), this may be motivated to two main factors:

- To keep a good standard of quality, Wikipedia users have started to make it harder for occasional editors to add new content, in some cases rejecting inappropriate edits, in other cases by subjecting them to a collective discussion that needs coordination and thus an increased overhead;

- A big number of topics of encyclopedic interest are already covered, in particular easy topics that do not require expertise of a field. As a consequence, it is harder for the average user to propose new topics to cover.

### 2.2.2 Wikipedia as a Knowledge Base

Wikipedia is a huge mine of semi-structured information. First of all, Wikipedia articles can be seen as a representation of specific and unambiguous entities. Their abstract and their content give a detailed description of the entity, metadata like the hit count and the revisions give information about its popularity

\[^2\]All data and charts refer to September 2016 and are taken from [http://stats.wikimedia.org/EN/](http://stats.wikimedia.org/EN/)
and how frequently its description changes, articles are categorized by a rich set of categories, and anchors of links to a Wikipedia page offer a set of commonly used synonyms for the concept the page is about. Wikipedia also offers some degree of structured information through templates, i.e. tables of structured information that need to be filled for specific types of articles.

Interesting information also lies into the structure of its graph, that can tell much about the relation among entities (Strube and Ponzetto, 2006; Rahimtoroghi and Shakery, 2011). An edge from article \( a_1 \) to article \( a_2 \) suggests some kind of semantic relation between them, since the text describing \( a_1 \) unambiguously cites the article \( a_2 \). Unfortunately, this does not always indicate an actual semantic proximity, because entities cited in a text may be not very relevant for the text. For example, the page about Shoes has a direct link to Bone even though the two entities are not strongly related. Given the Small-world property of the graph, this loose correlation gets milder increasing the length of the path in the graph: starting from the page about Astronomy in medieval Islam, in just two steps, the page about Jimi Hendrix can be reached. A mutual direct link between two pages indicates a stronger semantic relation, and many other graph-based relatedness functions have been developed to address the task of finding how semantically close two pages are (Milne and Witten, 2008a; Ratinov et al., 2011; Han and Zhao, 2010; Gabrilovich and Markovitch, 2007; Globerson et al., 2016; Kulkarni et al., 2009).

### 2.2.3 Wikidata

Wikidata (Vrandečić and Krötzsch, 2014) is a sister project of Wikipedia, and is run by the same organization, the Wikimedia Foundation. It was launched in October 2012. The goal of the project is to construct a clean and structured knowledge base. The editing process follows the same philosophy of Wikipedia: it is entirely curated by a community of users anyone can be part of, and is released under the open license Creative Commons CC0, which is specific for databases and lets anyone edit and redistribute the data. Data is mostly derived from Wikipedia. Wikidata, unlike Wikipedia, is multilingual by design, meaning that there is only one knowledge base that covers all languages. For example, entity Rome (the capital of Italy) has a German title Rom and an Italian title Roma. Wikidata currently features 21.6 million entities and 100 million statements. A statement is a property-value pair associated to an entity. For example, Rome has property population with value 2,777,979 and a property country with value Italy. But Rome has not always been in Italy: until year 258 b.C. it was in the Roman

---

3The anchor is contained in the sentence The foot contains more bones than any other single part of the body.

4Statistics gathered in October, 2016.
2.3. Piggyback: search engines as a source of information

*Empire.* Wikidata models this complex information with *qualifiers*, a sort of meta-properties, i.e. a property that relates to another property. With qualifiers we can model information relative to the statement such as periods of time in which the statements was true and sources according to the statement is true. For example, the property

\[
p = \textit{Rome country Roman Empire}
\]

may feature two associated qualifiers:

1. \textit{p end time 285}
2. \textit{p source The History of the Decline and Fall of the Roman Empire}

The *source* qualifier also lets us express different points of view on the same piece of information.

2.3 Piggyback: search engines as a source of information

By piggyback we refer to the idea of using search engines as sources of information. To our knowledge, it was first introduced by [Rocchio](1971), and later employed by [Xu and Croft](1996), to obtain pseudo-relevance feedback for query expansion. Search results or other signals from search engines have been used for sentiment analysis ([Turney](2002)), measuring document similarity ([Sahami and Heilman](2006)), or measuring sentence grammaticality ([Fujita and Sato](2008)). [Rüd et al.](2011) introduced the idea of piggybacking on search engines to enrich statistical NLP with a new set of features derived by search results. We will use a similar approach in Chapter 8 to draw important features about entities in web queries.

The Web is arguably the widest repository of human knowledge, and web search engines have the purpose of organizing the information provided by the Web and make it easily accessible by users. Web search engines are among the most complex software systems, and a thorough description of their internal functioning goes beyond the scope of this thesis. In addition to that, the algorithm behind commercial search engines is typically covered by trade secret. We limit ourselves to consider them at black boxes.

Search engines are typically accessed through a web site (see e.g. Figure 3.5 at page 30) that lets a user specify a query, run a search, and get results for that search. In their simplest definition, which will suffice the needs of our thesis, search engines associate a query to a ranked list of web pages related to the query.
A page should be ranked according to the likelihood that it provides the information the user needs, with relevant pages being assigned a higher rank. To facilitate the browsing of results search engines also show, for each retrieved page, *snippets* of text drawn from that page. Snippets are intended to give an overview of the content of that page, and in particular how the query keywords appear in that page. Most search engines emphasize query keywords appearing in snippets by rendering them in bold form, to let the user easily spot them.

In Chapter 8 we will use snippets returned by search engines to assist mention detection and entity disambiguation in queries.
3

Related Work and State of the Art

In this chapter, we present the published work made by both academy and industry in the field of entity linking. In particular, we give an overview of the efforts done by the community in developing entity linkers and in using them.

3.1 State of the art for entity linking

In recent years, a good deal of work has attempted to move beyond the representation of textual documents as a bag-of-words, and shift towards a semantic representation, by designing entity linkers (throughout the thesis we will also call them annotators or entity linking systems). Most recent works (Suchanek and Weikum, 2013) adopt information somehow drawn from Wikipedia, possibly in structured forms such as DBpedia or Wikidata, for both detecting entity mentions and disambiguating them. This is motivated by the fact that Wikipedia offers today the best trade-off between catalogs with a rigorous structure but low coverage (such as WordNet, CYC, TAP), and a large text collection with wide coverage but unstructured and noisy content (like the whole Web). The process of entity annotation generally involves three main steps, not necessarily in this order:

1. mention detection of the input text, which is the task of detecting parts of the text that may mention entities;

2. candidate entity generation, which is the task of finding for each mention a set of candidate entities that may be mentioned by it;

3. disambiguation, which is the task exploiting the context in order to select, for each mention, the most pertinent entity among the candidates, if such an entity exists.

The focus around entity annotators has increased significantly in the last few years (see e.g. Cucerzan (2007), Ferragina and Scaiella (2010, 2011), Kulkarni et al.
3.1. State of the art for entity linking

(2009); Meij et al. (2012); Milne and Witten (2008b); Ratinov et al. (2011); Gabrilovich and Markovitch (2009); Mihalcea and Csomai (2007)), with several interesting and effective algorithmic approaches to solve the mention-entity match problem, possibly using knowledge bases other than Wikipedia, such as DBpedia (Mendes et al., 2011), Wikidata (Geißand Gertz, 2016) or Yago (Yosef et al., 2011).

Unfortunately, the research has investigated some specific tasks using non-uniform terminology, non-comparable evaluation metrics, and limited datasets and systems. As a consequence, we only have a partial picture of the efficiency and effectiveness of known annotators which makes it difficult to compare them in a fair and complete way. This is a particularly important issue because those systems are being used as black-boxes by more and more IR tools, built on top of them, such as those proposed by Jiang et al. (2013); Scaiella et al. (2012); Vitale et al. (2012). Part of the work of this thesis aims at providing a common language for researchers.

3.1.1 Entity-linking systems

A number of entity linkers have been developed in recent years. Here we present an overview of some of them, selected because they employ interesting and original techniques or give particularly interesting results. This section covers entity linkers designed to deal with text providing some level of grammatical structure. For the state of the art of entity linkers designed to work on queries, see Chapter 6.

Wikify! (Mihalcea and Csomai, 2007) (University of North Texas, USA) was the first work to use Wikipedia as a catalog of entities, proposing the first method to link unstructured text against encyclopedic knowledge. Mentions are identified by searching the text for Wikipedia article titles and anchors. Candidate entities for a mention are ranked according to three methods: (i) \( tf \cdot idf \) between the mention and the candidate entity article, (ii) \( \chi^2 \) independence test that measures whether the mention refers to the entity more often than by chance, and (iii) a sort of link probability, measuring how much the mention is likely to be an anchor in Wikipedia. The annotation process consists in disambiguating entities via a machine learning classifier whose feature space exploits both local (i.e. term frequency) and topical (i.e. commonness) attributes of the candidate entities. Wikify! has been designed to deal with web pages. It cannot be accessed or downloaded (as of October 2016).

Collective Annotation of Web Text (Kulkarni et al., 2009) (IIT Bombay, India) formulates the problem of entity linking by capturing the tradeoff between the local bond between mention and candidate entity (computed by a SVM
### State of the art for entity linking

<table>
<thead>
<tr>
<th>Name</th>
<th>Problem</th>
<th>Avail.</th>
<th>Language</th>
<th>Code</th>
<th>KB</th>
<th>Ent. covg.</th>
<th>Doc. type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikify!</td>
<td>Sa2KB</td>
<td>N/A</td>
<td>EN</td>
<td>Not distributed</td>
<td>WKP</td>
<td>all</td>
<td>web pages</td>
</tr>
<tr>
<td>Collective ann.</td>
<td>Sa2KB</td>
<td>N/A</td>
<td>EN</td>
<td>Not distributed</td>
<td>WKP</td>
<td>all</td>
<td>web pages</td>
</tr>
<tr>
<td>TagMe 2</td>
<td>Sa2KB</td>
<td>Local/WS</td>
<td>EN/IT/DE</td>
<td>OS (Apache 2)</td>
<td>WKP</td>
<td>all</td>
<td>tweets</td>
</tr>
<tr>
<td>Illinois Wiki.</td>
<td>Sa2KB/D2KB</td>
<td>Local</td>
<td>EN</td>
<td>custom license</td>
<td>WKP</td>
<td>all</td>
<td>any len.</td>
</tr>
<tr>
<td>AIDA</td>
<td>Sa2KB/D2KB</td>
<td>Local/WS</td>
<td>EN</td>
<td>OS (Apache 2)</td>
<td>YG2</td>
<td>all</td>
<td>any len.</td>
</tr>
<tr>
<td>Wikipedia M.</td>
<td>Sa2KB</td>
<td>Local</td>
<td>Any WKP</td>
<td>OS (GPL2)</td>
<td>WKP</td>
<td>all</td>
<td>any len.</td>
</tr>
<tr>
<td>DBpedia Spotl.</td>
<td>Sa2KB</td>
<td>Local/WS</td>
<td>Any WKP</td>
<td>OS (Apache 2)</td>
<td>DBP</td>
<td>all</td>
<td>any len.</td>
</tr>
<tr>
<td>AGDISTIS</td>
<td>D2KB</td>
<td>Local</td>
<td>EN/DE/ZH</td>
<td>OS (LGPL3)</td>
<td>DBP</td>
<td>NE</td>
<td>any len.</td>
</tr>
<tr>
<td>Babelify</td>
<td>Sa2KB</td>
<td>WS</td>
<td>Any WKP</td>
<td>Not distributed</td>
<td>BBN/DBP</td>
<td>all</td>
<td>any len.</td>
</tr>
<tr>
<td>PBoH</td>
<td>D2KB</td>
<td>N/A</td>
<td>EN</td>
<td>Not distributed</td>
<td>WKP</td>
<td>all</td>
<td>any len.</td>
</tr>
<tr>
<td>DoSeR</td>
<td>D2KB</td>
<td>Local</td>
<td>EN</td>
<td>OS (GPL2)</td>
<td>DBP/YG3</td>
<td>all</td>
<td>any len.</td>
</tr>
<tr>
<td>WAT2</td>
<td>Sa2KB/D2KB</td>
<td>WS</td>
<td>EN</td>
<td>Not distributed</td>
<td>WKP</td>
<td>all</td>
<td>any len.</td>
</tr>
</tbody>
</table>

Table 3.1: Basic information about entity linkers. Column **Problem** indicates the problems natively solved by the system; for column **Availability**, **Local** means that the software can be downloaded and deployed locally, **WS** means that it is available as a Web Service; column **Language** show the languages of documents supported by the entity linker, **Any WKP** show the languages for which exists a Wikipedia edition are covered; column **Code** shows the code license and availability; column **KB** shows the knowledge base towards which mentions are linked: **WKP**=Wikipedia, **YG2**=Yago 2, **DBP**=DBpedia, **BBN**=BabelNet; column **Entity Coverage** shows whether mentions of all entities are linked or just named entities (see Section 2.1); column **Document type** shows if the system was designed specifically for some kinds of documents or is general purpose.

Since the optimal solution is NP-H to find, authors propose a hill climbing (greedy) disambiguation algorithm. This algorithm has been designed to deal with web pages. It cannot be accessed or downloaded (as of October 2016).

**TagMe 2** ([Ferragina and Scaiella, 2010](#)) (University of Pisa, Italy) searches the input text for mentions defined by the set of Wikipedia page titles, anchors and redirects. Each mention is associated with a set of candidate entities. Disambiguation exploits the structure of the Wikipedia graph and a combination of prior probability of the mention-entity link and the relatedness measure introduced in ([Milne and Witten, 2008a](#)), which takes into account the amount of common incoming links between two pages. TagMe’s disambiguation is enriched with a voting scheme in which all possible bindings model), and the collective bond between chosen entities across the document (based on the relatedness measure by [Milne and Witten](#) and the category-based relatedness by [Cucerzan](#)), to enforce coherence.
between mentions and entities are scored and then they express a vote for each other binding, enforcing annotation coherence. A mix of heuristics is eventually adopted to select the best annotation for each mention. TagMe 2 has been designed to deal with short texts. It is available as an online demo, can be queried through a public API and is open source\(^1\) (as of October 2016). Its release 2.0 has been provided in August 2012.

**Illinois Wikifier** (Ratinov et al., 2011) (University of Illinois, USA) searches the input text for mentions extracted from Wikipedia anchors and titles, using the Illinois NER system. Disambiguation is formulated as an optimization problem which aims at global coherence among all mentions. It uses a novel relatedness measure between Wikipedia pages based on NGD (Normalized Google similarity distance) and point-wise mutual information. Illinois Wikifier has been designed to deal with documents of arbitrary length. It is available as an online demo and its code can be downloaded\(^2\) (as of October 2016).

**AIDA** (Yosef et al., 2011) (Max Planck Institute for Informatics, Germany) searches for mentions using the Stanford NER Tagger (Klein and Manning, 2003) and adopts the YAGO2 knowledge base (Hoffart et al., 2011a) as catalog of entities, including their semantic distance. Disambiguation comes in three variants: PriorOnly (a baseline in which each mention is bound to its most-commonly linked entity in the knowledge base), LocalDisambiguation (each mention is disambiguated independently from others, according to a set of features which describe the mention and the entities), Cocktail-Party (YAGO2 is used to perform a collective disambiguation which aims at maximizing the coherence among the selected annotations, via an iterative graph-based approach). AIDA has been designed to deal with documents of arbitrary length. It is available as an online demo, it can be queried through a public API and its code can be downloaded\(^3\) (as of October 2016).

**Wikipedia Miner** (Milne and Witten, 2012) (University of Waikato, New Zealand) is an implementation of the Wikification algorithm presented in (Milne and Witten, 2008b), one of the first approaches proposed to solve the entity-annotation problem. This system is based on a machine-learning approach that is trained with links and contents taken from Wikipedia pages. Three features are then used to train a classifier that selects valid annotations discarding irrelevant ones: (i) the prior probability that a mention refers to a

\(^1\)https://tagme.d4science.org/tagme/
\(^2\)https://cogcomp.cs.illinois.edu/page/demo_view/Wikifier
\(^3\)http://www.mpi-inf.mpg.de/yago-naga/aida/
specific entity, (ii) the relatedness with the context from which the entity is extracted, given by the non-ambiguous spotted mentions, and (iii) the context quality which takes into account the number of terms involved, the extent they relate to each other, and how often they are used as Wikipedia links. Wikipedia Miner has been designed to deal with documents of arbitrary length. Its code can be downloaded\(^4\)(as of October 2016).

**DBpedia Spotlight** (Daiber et al., 2013) (Freie Universität Berlin, Germany) searches the input text for mentions extracted from Wikipedia anchors, titles and redirects; the parser is the LingPipe Exact Dictionary-Based Chunker\(^5\). It then associates a set of candidate entities to each mention using the DBpedia Lexicalization dataset. Given a spotted mention and a set of candidate entities, both the context of the mention and all contexts of each entity are cast to a Vector-Space Model (using a BOW approach) and the candidate whose context has the highest cosine similarity is chosen. Note that no semantic coherence is estimated among the chosen entities. Spotlight has been designed to deal with documents of arbitrary length. It is available as an online demo, it can be queried through a public API and is open source\(^6\)(as of October 2016).

**AGDISTIS** (Usbeck et al., 2014) (University of Leipzig, Germany) searches the text for named entities, after applying to the text a step of normalization (elimination of plural and genitive forms) and an expansion policy (a co-reference resolution of mentions inside a document, based on matching substrings). The disambiguation uses the graph of entities in the knowledge base, where nodes are entities and edges are relations that link two entities in the knowledge base. AGDISTIS considers a sub-graph of it, generated by traversing the graph starting from candidate entities and stopping at a fixed distance. The disambiguation, which consists in a subset of nodes of the generated sub-graph, is implemented by the HITS algorithm that selects most authoritative nodes. AGDISTIS has been designed to deal with documents of arbitrary length. It is available as an online demo and is open source\(^7\)(as of October 2016).

**Babelfy** (Moro et al., 2014) (Sapienza University of Rome, Italy) solves the problem of entity linking and word sense disambiguation (WSD) with a unified graph-based approach, using BabelNet (Navigli and Ponzetto, 2010) as its knowledge base. Disambiguation is done by combining semantic signatures

\(^{4}\)https://github.com/dnmilne/wikipediaminer/
\(^{5}\)http://alias-i.com/lingpipe/
\(^{6}\)https://github.com/dbpedia-spotlight/dbpedia-spotlight/
\(^{7}\)http://agdistis.aksw.org/demo/
of candidate entities and by measuring properties of the subgraph generated by them. The semantic signature of an entity is the set of nodes most related to it, as computed by a random walk on the graph of the whole knowledge base. Eventually, an heuristic chooses the densest subgraph (the subgraph that maximizes average node-degree). Babelfy has been designed to deal with documents of arbitrary length. It is available as an online demo or can be queried through a public API\footnote{http://babelfy.org/} (as of October 2016).

**PBoH - Probabilistic Bag of Hyperlinks** (Ganea et al., 2016) (ETH Zurich, Switzerland) makes use of a graphical model to perform collective mention disambiguation. The graph is built with a probabilistic approach combining a document-level probability of entity co-occurrences with local information captured from mentions and their surrounding context. The model is based on simple sufficient statistics extracted from data, thus relying on few parameters to be learned. The system uses loopy belief propagation to perform approximate inference. PBoH has been designed to deal with documents of arbitrary length. It cannot be accessed or downloaded (as of October 2016).

**DoSeR** (Zwicklbauer et al., 2016a) (University of Passau, Germany) uses a graph-based disambiguation algorithm based on entity embeddings and document embeddings. Semantic entity embeddings are derived using \texttt{word2vec} (Mikolov et al., 2013), while document embeddings are computed using \texttt{doc2vec} (Le and Mikolov, 2014). A graph of candidate entities is build, where nodes are entities and edges are labeled with the harmonic mean between semantic and context similarity of two entities. This similarity is computed as the cosine similarity between \texttt{word2vec} (semantic) and \texttt{doc2vec} (context) embeddings. The final set of disambiguated entities is found with PageRank. DoSeR has been designed to deal with documents of arbitrary length. Its code can be downloaded\footnote{https://github.com/quhfus/DoSeR} (as of October 2016).

**WAT2** (Piccinno, 2016) (University of Pisa, Italy) models disambiguation as a Learning to Rank task, in which candidate entities for each mention are ordered by the likelihood of them being the pertinent entity for that mention. As ranking algorithm, the author uses LambdaMART (Burges, 2010b). Features associated to a candidate entity and used for ranking are based on a voting scheme (similar to that proposed for TagMe) and to two kinds of word embeddings: (i) one that takes into account the words appearing near a mention of the candidate entity in Wikipedia, and (ii) a random walk on the Wikipedia graph similar to \textit{DeepWalk} (Perozzi et al., 2014). The top-ranking candidate entity for each mention is linked to the mention. WAT2
3.1. State of the art for entity linking

has been designed to deal with documents of arbitrary length. It will be possible to query it through an API interface\(^{10}\) (as of October 2016).

Table 3.1 shows basic data about the entity linkers covered in this chapter\(^{11}\)

3.1.2 Datasets for benchmarking entity linkers

To evaluate the performance of entity linkers, a number of benchmarking datasets have been created and distributed to the public. A dataset provides a set of instances and their associated ground truth. This section discusses datasets covering text with some level of grammatical structure. All datasets feature English documents. For datasets covering queries, see Chapter 6. Here follows a description of all publicly available benchmarking datasets for entity linking.

News datasets

**AIDA/CoNLL** (Hoffart et al., 2011b) features documents from the Reuters Corpus V1 and consists of newspaper articles. Average document length is 1130 characters. A large subset of mentions (though not all of them), including the most important ones, are annotated. Entities are annotated at each occurrence of a mention. This is currently the largest and most accurate dataset for article-length documents. It is divided in three parts: Train (for training), Test-A (for validation) and Test-B (for blind evaluation).

**MSNBC** (Cucerzan, 2007) features newspaper articles from the MSNBC news network. Only important entities are annotated and all occurrences of mentions that refer to those entities are annotated. Average document length is 3316.

**AQUAINT** (Milne and Witten, 2008b) features a subset of the original AQUAINT corpus, consisting of newswire text data. The average document length is 1415 characters. Not all occurrences of the mentions are annotated: if more than one mention in a document refers to the same entity, only the first mention is actually annotated. Moreover, only the mentions that refer to entities considered important are annotated. This reflects the Wikipedia-style linking.

**ACE2004** (Ratinov et al., 2011) features a subset of the original ACE 2004 Multilingual Training Corpus, originally created for training and testing information extraction tasks. Annotations are found using Amazon Mechanical

---

\(^{10}\)Personal communication by author F. Piccinno.

\(^{11}\)The second column reports the entity linking problem natively solved by the system. These problems will be presented in Chapter 4 and the reader may ignore them for now.
3.1. State of the art for entity linking


**N3-Reuters-128** ([Röder et al., 2014](#)) is a subset of the Reuters-21578 corpus featuring economic news articles. It consists of 128 news articles containing at least one named entity.

**Web page datasets**

**IITB** ([Kulkarni et al., 2009](#)) features web pages of popular sites belonging to a handful of domains that include sports, entertainment, science and technology, and health. The dataset consists in a total of 107 documents with a total of 17,200 entity mentions. The average document length is 3879 characters.

**Tweets, RSS and single-sentence datasets**

**Meij** ([Meij et al., 2012](#)) features microblog messages publicly available on twitter. Average document length is 80 characters. All entities contained in each tweet are tagged.

**Microposts2014** ([Cano et al., 2014](#)) features 3,505 tweets collected for the period 15th July 2011 to 15th August 2011 (31 days). A number of (but not all) tweets are related to a notable event that happened in that time frame, including the death of Amy Winehouse, the London Riots and the Oslo bombing. The dataset is split among a Train portion (2,340 tweets) and a Test portion (1,165 tweets). In average, each document consists of 5.4 words.

**N3-RSS-500** ([Röder et al., 2014](#)) features 500 sentences extracted from newspaper RSS feeds, mostly reporting events and factual content (e.g. “Italy’s PM said that...”).

**KORE 50** ([Hoffart et al., 2012](#)) is a subset of sentences of the AIDA/CoNLL corpus, limited to mentions that are particularly hard to disambiguate, for example first names of persons whose identity can only be inferred by analyzing the context in which they appear. It comprises 50 sentences from different domains, such as music, celebrities, and business.

Some excerpts of documents, together with the ground truth proposed by the dataset, are given in Figures [3.1](#), [3.2](#), [3.3](#) and [3.4](#).
3.1. State of the art for entity linking

Figure 3.1: Excerpt of an example document (news story) given in the AIDA-/CoNLL dataset with its annotations, constituting the ground truth for this document.

Figure 3.2: Excerpt of an example document (news story) given in the MSNBC dataset with its annotations, constituting the ground truth for this document.
3.2 Applications of entity linking

In this section, we present works that make a central use of entity linking. Entity linking has been successfully applied to improve the performance on a range of problems and build products and systems. Applications of entity linking can be

---

The second column reports the entity linking problem for which the dataset provides a ground truth. These problems will be presented in Chapter 4, and the reader may ignore them for now.
3.2. Applications of entity linking

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Problem</th>
<th>KB</th>
<th>Docs type</th>
<th>Docs</th>
<th>Ann (tags)</th>
<th>Distinct entities</th>
<th>Avg. Ann/Doc</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE2004</td>
<td>A2KB</td>
<td>WKP</td>
<td>news</td>
<td>35</td>
<td>257</td>
<td>255</td>
<td>7.3</td>
</tr>
<tr>
<td>AIDA/CO-NLL</td>
<td>A2KB</td>
<td>YG2/WKP</td>
<td>news</td>
<td>1393</td>
<td>27815</td>
<td>5591</td>
<td>20.0</td>
</tr>
<tr>
<td>AQUAINT</td>
<td>A2KB</td>
<td>WKP</td>
<td>news</td>
<td>50</td>
<td>727</td>
<td>572</td>
<td>14.5</td>
</tr>
<tr>
<td>MSNBC</td>
<td>A2KB</td>
<td>WKP</td>
<td>news</td>
<td>20</td>
<td>658</td>
<td>279</td>
<td>32.9</td>
</tr>
<tr>
<td>N3-Reuters-128</td>
<td>A2KB</td>
<td>DBP</td>
<td>news</td>
<td>128</td>
<td>650</td>
<td>299</td>
<td>5.0</td>
</tr>
<tr>
<td>IITB</td>
<td>A2KB</td>
<td>WKP</td>
<td>web pages</td>
<td>103</td>
<td>11245</td>
<td>3739</td>
<td>109.1</td>
</tr>
<tr>
<td>Meij</td>
<td>C2KB</td>
<td>WKP</td>
<td>tweets</td>
<td>502</td>
<td>812</td>
<td>567</td>
<td>1.6</td>
</tr>
<tr>
<td>Microposts2014</td>
<td>A2KB</td>
<td>DBP</td>
<td>tweets</td>
<td>3505</td>
<td>5277</td>
<td>2696</td>
<td>1.5</td>
</tr>
<tr>
<td>KORE50</td>
<td>A2KB</td>
<td>DBP</td>
<td>sentences</td>
<td>50</td>
<td>144</td>
<td>127</td>
<td>2.9</td>
</tr>
<tr>
<td>N3-RSS-500</td>
<td>A2KB</td>
<td>DBP</td>
<td>sentences</td>
<td>500</td>
<td>524</td>
<td>400</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 3.2: Column Problem indicates the problem for which the dataset offers a set of instances and their ground truth solutions; column KB shows the knowledge base towards which mentions are linked: WKP=Wikipedia, YG2=Yago 2, DBP=DBpedia, BBN=BabelNet; Docs is the number of documents (instances) contained in the dataset; Ann (Tags) is the number of overall annotations or tags in the dataset; Distinct entities is the number of distinct entities that appear in the dataset; Avg. Ann/Doc is the average number of annotations or tags per document.

3.2.1 Entity as an objective

In this kind of applications, the final aim is to find information related to an entity, or find the entity itself. A prominent example of such applications is semantic search [Meij et al., 2014], where a user query can be translated into a query to a knowledge base. For example, consider queries like Who is the current president of the U.S.? or When did the Waterloo battle happen?. Other examples of applications include topic suggestion, where a user reading a document gets automatically generated references to interesting topics, and entity-enhanced user interaction, in which entity properties are used to facilitate a user’s work-flow. Here follows a brief description of the most interesting ones:

Knowledge base question answering [Bao et al., 2014] (Microsoft research) provides answers to questions in natural language. Authors model the task as a machine translation problem: questions are parsed with a CYK parser, then each cell (part of the question) is translated into its answer, retrieved from a knowledge base. The answer to the question is that of the root cell. This lets authors answer complex questions like director of movie starred by
Tom Hanks. Authors show that the system overcomes the state-of-the-art on questions of general domain.

**Google Knowledge panel** *(Google)* shows, as results of a web search, a rich amount of information drawn the Google Knowledge Graph, a knowledge base partially derived from Wikipedia. A panel on the right of the screen shows information about the entities that are mentioned in the query. See Figure 3.5 for an example.

**Google direct result** *(Google)* shows, as results of a web search, a single piece of information that represent a *solution* to the query. See Figure 3.6 for an example.

### 3.2.2 Entity as a feature

In this kind of applications, entities are used as internal means of representing information. Example applications include document classification, clustering, retrieval, and event modeling.

**DEESSE** *(Van Laere et al., 2014) (Yahoo Labs)* is a search system that lets a user navigate through documents, providing the user with suggestions of other interesting documents. The user issues an entity as query, and the system returns documents mentioning that entity. It also suggests other interesting entities using an algorithm based on random walk on the entity graph. Entities are extracted from documents with a machine-learning approach *(Zhou et al., 2010)* and ranked by their *aboutness* as proposed by *(Paranjpe, 2009)*.

**EQFE** *(Dalton et al., 2014) (University of Massachusetts Amherst)* employs entities to overcome the limits of the classical task of document retrieval. Authors propose a document retrieval system based on entity query expansion, in which queries and documents are represented with entity identifiers and other entity features retrieved from the knowledge base. The entity-based search engine reached a solid performance on different document collections, outperforming state of the art systems.

**Document relatedness** *(Ni et al., 2016) (IBM Research)* deals with the problem of finding the semantic similarity among documents by representing them as a *concept graph*, i.e. a graph where nodes are DBPedia entities mentioned by the documents. Documents similarity is computed through the similarity of their entities, taking into account the frequency of co-occurrence, their position in the category structure, and their structural information
provided by the knowledge base. The centrality of each node in the generated graph is computed, and document similarity is computed as a measure of sub-graphs similarity. Authors show that such a similarity measure outperforms state of the art measure like ESA (Gabrilovich and Markovitch, 2007).

**Topical clustering** (Scaiella et al., 2012) (University of Pisa) proposes an entity-based method for clustering short documents (in this case, snippets of search results). Authors model this task as a labelled clustering problem, building a snippet-entity graph, in which edges either represent the similarity of two entities, or the membership of an entity to a snippet. Spectral properties of the graph are used to build a clustering algorithm that enforces inter-cluster coherence. Authors show that this method outperforms state of the art snippet clustering methods developed by both industry and academia.

**Topical classification** (Vitale et al., 2012) (University of Pisa) proposes a new method for the classification of short textual documents. As opposed to enriching features of text and categories with latent or explicit topic modeling and then building a ML-based classifier, authors represent documents and categories as a set of Wikipedia entities (acting as topics treated by the text/category) and classifies new documents based on the similarity of entities, that is computed through the Wikipedia graph. Document are annotated using TagMe. Authors show that topical classification overcomes, or in some cases reaches the same results, as state of the art classification methods that often need expensive training.

**Memory-based event modeling** (Pighin et al., 2014) (Google) presents a method for abstractive summarization of news (the task of summarizing a document, not necessarily using the document’s own words). In a stage of its pipeline, the system generates event patterns based on entities: if, in a short period of time, the same pair of entities appear in two news from different sources, that is a strong indicator that the two sentences in which the entities appear are paraphrases. For example, if two news sources report two sentences:

1. Angelina Jolie files for divorce from Brad Pitt
2. Angelina and Brad’s marriage comes to an end

the system learns that the two patterns

1. [PER] files for divorce from [PER]
2. [PER] and [PER]’s marriage comes to an end
3.2. Applications of entity linking

Figure 3.5: An example search on Google, the first search engine to show as result, in addition to web pages, a knowledge panel that shows basic information about the entities mentioned in the query drawn from a knowledge base (Google Knowledge Graph).

are paraphrases, i.e. ways of expressing the same concept. Patterns generated in this way, that are grammatical because they have been observed in well-formed text, are used to generate new compressed sentences.
Figure 3.6: An example search on Google for query *how far is the moon?*, where the search engine returns a *direct result*, i.e. a solution for the query drawn from a knowledge base (the Google Knowledge Graph).
Part I

A framework for entity linking
The BAT-Framework: a Formal Framework for Entity Linking

Formalizing concepts related to entity linking is key to build a common language for the field. In this chapter, we present a formal definition of concepts related to entity linking and used in this thesis, such as what is an entity and what types of entities exist, we define a terminology, a set of problems related to entity linking, and a computational hierarchy among them. All these formalizations, along with the metrics built on top of them that will be presented in Section 4.3, constitute the BAT-Framework\(^1\) our proposal for what concerns the formalization of entity linking and its evaluation metrics, that became the de-facto common language among researchers.

4.1 Formalizing entity linking

The task of entity linking has been considered starting of 2008 by people with different backgrounds, having in mind distinct domains of application. This led to a lack of uniformity in terminology, in the formal definition of problems, and in evaluation methods. The BAT-Framework, a formal framework we proposed in 2013 (Cornolti et al., 2013), overcomes this issue by proposing the following contributions:

- The definition of a root-level categorization of entities in three domains;
- The definition of five problems related to entity-linking;
- A hierarchy of adaptations between those problems, that lets us evaluate a wide range of algorithms;

\(^1\)BAT stands for Benchmarking Annotators of Text. Any resemblance to actual superheroes is purely coincidental.
• A set of basic measures that generalize standard evaluation measures such as true/false positives, true/false negatives, precision and recall by defining them on top of a binary relation which represents a “correct match” between either two tags or two annotations.

• A set of match relations to instantiate the basic measures, each focusing on a particular aspect we want to measure.

• A set of measures for the similarity of systems, defined on top of a binary relation $M$.

We also proposed a software implementation of these evaluation metrics. It is released under the GPLv3 license as a contribute to the scientific community. The main aim of this framework is to facilitate the development of novel systems, algorithms, and techniques. The framework shows weaknesses and strengths of an entity-annotation system, inspecting its result and comparing it against a ground truth made by human annotators, thus treating the system as a black box. In Section 4.6 we will present GERBIL, a software built on top of the BAT-Framework that provides easy access to experiments.

### 4.1.1 Root categorization of entities

Let us start by presenting the root-level categorization of entities that we will use throughout the thesis. It consists of three main domains:

**Named entities.** Specific things that exist as themselves only once. Named entities typically have (at least one) proper name. Examples of named entities are *Neil Armstrong; Batman; Microsoft; Yesterday* (The Beatles’ song); *New York City; September 11 attacks; Space elevator.*

**Abstract entities.** Abstractions that exist as themselves only once. For example, *peace; intelligence; mathematics; centripetal force.*

**Categories of entities.** Sets of entities sharing some properties. Can be sets of named entities, e.g. *astronaut; superhero; terrorist attack; family car,* or sets of abstract entities, e.g. *branch of knowledge; mental faculty; physical concept.*
4.1. Formalizing entity linking

4.1.2 Terminology

Here follows the terminology proposed by the BAT-Framework, that will be used in the next chapters of this thesis.

Let us give the following definitions:

**Definition 4.1.** An entity is an item of a specific knowledge base. It is identified by a unique ID value such as an URI.

**Definition 4.2.** A mention is a sequence of terms located in a text document that explicitly refer to an entity. It can be codified as a pair \((b, l)\) where \(b\) is the index of the first term and \(l\) is the index of the last term (exclusive).

**Definition 4.3.** A score is a real value \(s \in [0, 1]\) that can be assigned to an annotation or a tag. Higher values of the score indicate that the annotation (or tag) is more likely to be correct.

**Definition 4.4.** A tag is a statement according to which a document, in at least one of its parts, explicitly mentions an entity. It is codified as the entity \(e\) the document refers to. A tag may have a score: a scored tag is encoded as a pair \((e, s)\), where \(s\) is the score.

**Definition 4.5.** An annotation is a statement according to which a mention in a specific text document refers to an entity. It can be codified as a pair \((m, e)\) where \(m\) is the mention and \(e\) is the entity. An annotation may have a score: a scored annotation can be codified as \((m, e, s)\), where \(s\) is the score.

To underline the idea of linking, we will also write an annotation \((m, e)\) using the notation \(m \rightarrow e\), for example, we will write the annotation in Figure 4.1 as \(Armstrong \rightarrow Neil Armstrong\).

\footnote{https://github.com/marcocor/bat-framework}{Note that in this context, existence is not bound to having a manifestation in the real world, but more broadly as something that has appeared in human culture. This includes concrete, actual, physical, as well as potential, abstract, and imaginary things.}
4.1.3 The entity linking problems

The task of entity linking has historically been defined with slight differences, depending on the specific application. This led to the definition of a range of computational problems, and a set of algorithms for solving them. In this section we will describe them and in the next section we will provide a hierarchy for those problems, and explore the implications of that.

In our original work (Cornolti et al., 2013), we defined all problems with respect to linking towards Wikipedia. We later generalized the definition to linking towards a generic knowledge base in (Usbeck et al., 2015). In this thesis, we present the generalized version of the problems.

Here follows the list of problems related to entity linking. Table 4.1 shows, for each problem, the expected input (that is, the type of a problem instance) and generated output (the type of the solution). The reader might find easier to follow the description of problems by keeping an eye on Figure 4.2, that shows some examples of input and correct output for the given problems.

Disambiguate to Knowledge Base (D2KB) Given a text and a set of mentions, assign to each input mention the entity it refers to. Note that for solving this problem it is not requested to spot other mentions in the input text, but typically D2KB algorithms somehow leverage on the rest of the text to assist disambiguation.

Annotate to Knowledge Base (A2KB) Given a text, identify the set of annotations in a document. This problem consists in both finding the mentions and linking them to the proper entity.

Score-annotate to Knowledge Base (Sa2KB) As A2KB, but each annotation is also assigned a score representing the likelihood that the annotation is correct.

Concepts to Knowledge Base (C2KB) Given a text, identify the set of tags in a document. In other words, a C2KB algorithm must focus on finding the entities mentioned in a text, but there is no need to specify their position in the text.

Score-concepts to Knowledge Base (Sc2KB) As C2KB, but each tag is also assigned a score representing the likelihood that the tag is correct.

---

4 This problem has been introduced in Cucerzan (2007).
5 This problem has been introduced in Milne and Witten (2008b).
6 This problem has been introduced in Milne and Witten (2008b).
7 This problem has been introduced in Meij et al. (2012).
4.1. Formalizing entity linking

<table>
<thead>
<tr>
<th>Problem</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2KB</td>
<td>Text, Set of mentions</td>
<td>Set of annotations</td>
</tr>
<tr>
<td>A2KB</td>
<td>Text</td>
<td>Set of annotations</td>
</tr>
<tr>
<td>Sa2KB</td>
<td>Text</td>
<td>Set of scored annotations</td>
</tr>
<tr>
<td>C2KB</td>
<td>Text</td>
<td>Set of tags</td>
</tr>
<tr>
<td>Sc2KB</td>
<td>Text</td>
<td>Set of scored tags</td>
</tr>
</tbody>
</table>

Table 4.1: Entity linking problems, their input and output.

![Figure 4.2](image)

Figure 4.2: For each problem, an example input and the corresponding output. Annotations are in green, tags in blue, mentions in red.
A2KB is the problem of finding mentions in a given text and annotate them with entities. It can be divided in two sub-problems: that of finding sequences of terms which are mentions (i.e. refer to an entity), usually referred to as entity recognition, and that of choosing the right entity for a mention, referred to as entity disambiguation (algorithms that solve A2KB do not necessarily solve these sub-problems in this order). The performance of a system solving A2KB depends on both entity recognition and disambiguation.

In D2KB, the correct mentions are fed to the algorithm as input. In other words, the problem instance already provides the position of sub-lists of terms referring to an entity, and the algorithm only needs to link them to the correct entities.

C2KB is the problem of finding entities mentioned in a text, without providing information on where they are (i.e. what terms mention those entities). It is anyway worthy to underline that entities found with C2KB are those explicitly mentioned by the text, and this makes finding the solution to this problem very similar to solving A2KB.

Sa2KB and Sc2KB are the same problem as A2KB and C2KB, respectively, with the only difference that results are associated a score reporting the confidence of the algorithm in a specific annotation or tag. It is important to note that these are not (necessarily) probability scores, i.e. they are not necessarily computed according to probability laws. They must rather be interpreted as a way to define a pre-order over the objects. For example, given scored tags \langle e_1, 0.3 \rangle, \langle e_2, 0.6 \rangle, we can infer that \( e_1 \) has a higher confidence than \( e_2 \), but not that it has double the probability of being correct.

Implicit and explicit mentions

Both annotations and tags, by their definitions 4.5 and 4.4, consider only explicitly mentioned entities. For example, the sentence I loved the first movie by Tarantino does not explicitly mention Reservoir Dogs, whereas it explicitly mentions entities such as movie and Quentin Tarantino.

Finding entities implicitly mentioned by a document, or entities semantically related to a document might as well have an important impact on many applications. For example, a clustering algorithm might increase clustering precision by considering the sentence Largest planet in the solar system as associated to entities such as Astronomy and Jupiter, even though they are not mentioned in the text.

Covering the problems related to finding implicit entities goes beyond the scope of this thesis. We limit to pointing out that it is often harder, even for humans, to define what entities are implicitly mentioned in a document and what
Entity linking problems and applications

As said, problems have been defined in accordance to specific applications that use entity linking. To explain the necessity of all these problems, we give examples of such applications for each problem.

A topical clustering of documents can be built on top of a document representation as a bag of entities. Based on this, we can define document similarity distances based on the distance of a document’s entities. For such an application, we need an algorithm that solves C2KB (or its scored variant Sc2KB), i.e. finds the entities mentioned in a document. We do not need information about where in the document an entity has been mentioned (A2KB). Successful applications of this problem are presented in (Hu et al., 2008, 2009b; Scaiella et al., 2012).

On the other hand, an information extraction system based on the analysis of the syntax of a sentence, would need to be fed with the exact location in the text in which an entity has been mentioned, so to know that a specific node in the dependency tree refers to an entity. Such an application would need an algorithm that solves A2KB (or its scored variant Sa2KB), i.e. finds the annotations (entities and their mention) in a document.

The resolution of the D2KB problem has no practical use per-se. This problem has rather been introduced to focus on the challenge of entity disambiguation. A D2KB system cannot be used on its own, but it can be employed as a step in a pipeline, preceded by an entity recognition step.

4.2 An adaptability hierarchy for entity linking problems

In some cases, the computational problems faced by distinct studies are closely related to each other, and under certain conditions, algorithms natively solving distinct problems can be compared to each other with an acceptable margin of loss.

In this section, we present a method for achieving this comparison. The method lets us compare the output (for the same set of instances) of two algorithms $S_P$ and $S_{P'}$, respectively solving problems $P$ and $P'$. The core of the method is the adaptation of an instance for problem $P$ to an instance of problem $P'$, that can be fed as input to $S_{P'}$, and the reverse adaptation of the output of $S_{P'}$ to an output of problem $P$, so that this approximate solution found by $S_{P'}$ is comparable to the output of $S_P$ for the same instance. If such an adaptation loses only a minimal...
amount of information, \( S_P \) and \( S_{P'} \) can be compared in a fair way with respect to their ability to solve \( P \). Note that an adaptation always exists, but it is not always feasible to define adaptations that introduces a minimal loss. We will define a set of adaptations for entity linking problems in Table 4.2 that, we argue, brings a minimal loss and keeps the evaluations fair.

Let us give some definitions concerning the evaluation of algorithms on benchmarking datasets. Note that in this section we will keep evaluation metrics general, and only assume that they are comparable with each other, while in Section 4.3 we will instantiate them.

**Definition 4.6 (Benchmark dataset).** A benchmark dataset for problem \( P \) is a collection of instances for problem \( P \) and their associated ground-truth solutions.

**Definition 4.7 (Metric).** A metric \( m \) for problem \( P \) is a function 

\[
m : L_P^N \times L_P^N \mapsto \mathbb{R}
\]

where \( L_P^N = \{ (d_1, \cdots, d_i, \cdots, d_N), \ d_i \in T_P \} \) and \( T_P \) is the domain of solutions for problem \( P \).

Metrics (Definition 4.7) are functions that get as arguments two vectors of \( N \) outputs of problem \( P \) and output a real number. We will use them to compare the approximate solution for a list of instances found by an algorithm (the first argument of function \( m \)) against the list of ground truths provided by the dataset for those instances (the second argument of function \( m \)). In order to make sense, such functions must be chosen to be monotonic with respect to the distance between the approximate solutions and the ground truths, i.e. the value of a metric for a solution closer to the ground truth (with respect another solution, and according to some aspect we want to measure) must be higher. Note that we keep the notion of distance between solution and ground truth deliberately vague, and will instantiate it in the next sections according to specific aspects of quality we need to measure. The quality of all solutions is measured by a single real number, so that it can be compared to other measures.

**Definition 4.8 (Approximate adaptation).** An approximate adaptation of problem \( P' \) towards problem \( P \) is a pair of algorithms \((A_I, A_O)\) such that:

- \( A_I \) generates, from any instance of \( P \), an instance of \( P' \)
- \( A_O \) generates, from any approximate solution for \( P' \), an approximate solution for \( P \).

We express with the relation \( P \preceq P' \) the fact that an approximate adaptation of problem \( P' \) towards problem \( P \) is defined in our framework. In this case, we say
that $P'$ is adaptable to $P$. From a practical point of view, $P \preceq P'$ implies that an instance of $P'$ can be adapted to become an instance of $P$; the latter instance can be fed into an algorithm that solves $P$, and its solution be adapted into a solution of $P'$. In other words, we can find an approximate solution for $P'$ without solving directly $P'$. Similarly to metrics, to our ends of evaluating the quality of an algorithm, approximate adaptations are interesting only in case they are monotonic, i.e. given an instance, improving the solution $S_P$ found by the system for $P$ also improves (or does not alter) the adapted solution $A_O(S_P)$ for $P'$. The reader might find similarities between our notion of approximate adaptation and the theory of problem reduction (Garey and Johnson 1990), despite the resemblance, they apply to different types of problems.

Lemma 4.1 (Transitivity of $\preceq$).

$$P \preceq P' \land P' \preceq P'' \Rightarrow P \preceq P''$$

Proof. From $P' \preceq P'' \land P'' \preceq P'''$ it follows that there exist two pairs of algorithms $(A'_I, A'_O)$ and $(A''_I, A''_O)$, such that

- $A''_I$ generates, from any instance of $P'''$, an instance of $P''$
- $A'_I$ generates, from any instance of $P''$, an instance of $P'$
- $A'_O$ generates, from any approximate solution for $P'$, an approximate solution for $P''$.
- $A''_O$ generates, from any approximate solution for $P''$, an approximate solution for $P'''$.

An approximate adaptation of problem $P'$ towards problem $P'''$ can be defined as $(A_I, A_O)$ where $A_I(x) = A'_I(A''_I(x))$ and $A_O(x) = A''_O(A'_O(x))$. [4]

Lemma 4.2 (Reflexivity of $\preceq$).

$$P \preceq P$$

Proof. There always exists a pair of algorithms $(A_I, A_O)$ that verify Definition 4.8 $A_I = A_O = I$, where $I$ is the identity function.
4.2. An adaptability hierarchy for entity linking problems

Relation \( \trianglerighteq \) is transitive and reflexive. To our ends, the latter property has an obvious consequence: an algorithm solving \( P \) can be tested for its ability to solve problem \( P \) itself, with no adaptation. These two properties tell us that \( \trianglerighteq \) forms a preorder. As we will see, this has consequences on the comparability of entity linking problems.

**Definition 4.9** (Approximate evaluation). Let \( S \) be an algorithm that provides an approximate solution for problem \( P' \), \( D \) be a benchmark dataset for problem \( P \), \( (A_I, A_O) \) be an approximate adaptation of problem \( P' \) towards \( P \) (hence \( P \trianglerighteq P' \)), and \( m \) be a metric. An approximate evaluation of algorithm \( S \) on \( D \) performed by means of \( (A_I, A_O) \), with metric \( m \) is computed as:

\[
m((s_1, \ldots, s_i, \ldots, s_{|D|}), (g_1, \ldots, g_i, \ldots, g_{|D|}))
\]

where \( s_i = A_O(S(A_I(d_i))) \), and \( d_i, g_i \) are respectively the \( i \)th instance and corresponding ground truth provided by benchmark dataset \( D \).

Approximate evaluations put all the pieces together: intuitively, we define a way to evaluate the quality of an algorithm \( S \) solving \( P' \) against a benchmark dataset for problem \( P \), under the constraint \( P \trianglerighteq P' \), i.e. there exists an approximate adaptation \( (A_I, A_O) \) from \( P' \) towards \( P \). From an algorithmic point of view, approximate evaluations are done in the following way: each dataset instance \( d_i \) (of problem \( P \)) is adapted to an instance \( A_I(D_i) \) of \( P' \). Algorithm \( S \) is given as input \( A_I(D_i) \), and its result adapted back to a solution for \( P \) through \( A_O \). The quality of the solutions for the whole dataset is measured against the ground truth \( g_i \) according to a metric \( m \).

Let \( X \) and \( Y \) be algorithms respectively solving problems \( A \) and \( B \), and let \( D_P \) be a dataset for problem \( P \). We can compare the approximate evaluations of algorithms \( X \) and \( Y \) on dataset \( D_P \) if \( P \trianglerighteq A \land P \trianglerighteq B \). This way two algorithms can be compared to each other even though they do not natively solve the same problem. Since relation \( \trianglerighteq \) is reflexive, two algorithms can be obviously compared if they both solve problem \( P \) and there is a dataset for such a problem. Moreover, since \( \trianglerighteq \) is a preorder, two algorithms can be compared if they solve respectively \( A \) and \( B \), \( A \trianglerighteq B \), and there is a dataset for \( A \).

Let us give an example of this last statement. Say we want to compare the output quality of two algorithms \( X \) and \( Y \), respectively solving \( A2KB \) and \( C2KB \), with respect to a benchmark dataset \( D \) for problem \( C2KB \). Document \( d \), provided as instance by \( D \), does not need any adaptation to fit as input to problem \( A2KB \), since both problems have text documents as instances (see Table 4.1). Once document \( d \) is fed as input to the two algorithms, they will respectively produce solutions \( S^A2KB_X \) (which is a set of annotations in document \( d \)) and \( S^C2KB_Y \) (which is a...
4.3 Evaluation metrics for entity linking

<table>
<thead>
<tr>
<th>Adaptation</th>
<th>Instance adaptation</th>
<th>Solution adaptation $A_O$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2KB ≼ Sa2KB</td>
<td>Identity.</td>
<td>Discard the scores, keep only annotations with a score higher than a given threshold.</td>
</tr>
<tr>
<td>C2KB ≼ Sc2KB</td>
<td>Identity.</td>
<td>Discard the scores, take only tags with a score higher than a given threshold.</td>
</tr>
<tr>
<td>D2KB ≼ A2KB</td>
<td>Keep document, discard the mentions.</td>
<td>Let $M$ be the set of mentions to disambiguate, part of the instance. Take only the annotations $\langle m, c \rangle$ of the solution such that $m \in M$.</td>
</tr>
<tr>
<td>Sc2KB ≼ Sa2KB</td>
<td>Identity.</td>
<td>Discard the mentions, keep the entities and their score and transform them in a scored tag. In case more than one annotation points to the same entity, keep only the one with highest score.</td>
</tr>
<tr>
<td>C2KB ≼ A2KB</td>
<td>Identity.</td>
<td>Discard the mentions, keep only the set of entities.</td>
</tr>
</tbody>
</table>

Table 4.2: Adaptations between problems.

The adaptations we propose are presented in Table 4.2. Proof of their monotonicity is trivial.

Keeping in mind that the adaptations between problems is transitive and reflexive, this leads to a hierarchy of problems illustrated by the graph in Figure 4.3. The most general problem is Sa2KB, which is adaptable to all other problems. Problems D2KB and C2KB are the most specific, meaning that an algorithm natively solving either D2KB or C2KB can only be evaluated with respect to its ability to solve it.

4.3 Evaluation metrics for entity linking

The issue of establishing common metrics, shared by the community, to evaluate the performance of entity-linking algorithms against a benchmark dataset is of crucial importance to guide the research in the development of new algorithms. In this chapter, metrics for the evaluation of correctness are proposed. The aim is to establish a set of experiments that fairly evaluate the performance
of an entity-linking algorithm and evaluate how similar the solutions found by two algorithms are (Section 4.5).

The more a problem is related to culture and human knowledge, the more its complexity makes it impossible for algorithms (and even for humans) to find an exact solution. All algorithms that solve NLP problems such as entity linking provide approximate solutions. In this chapter, we formalize measures to benchmark the quality of these solutions and find how distant the approximation is from a ground truth defined by humans.

In (Cornolti et al., 2013), we proposed a set of metrics to measure the performance of a generic entity linking system, using a variant of the classic IR metrics to compare a system output against a dataset. With performance we refer, throughout this thesis, to the quality of the approximated result with respect to the ground truth.

As seen in Section 4.1.3, the basic elements forming a solution are either tags (entities associated to a document) or annotations (entities associated to a mention), depending on the problem at stake. For an instance (a document), an algorithm provides as solution a collection (either in a set or in a ranked list) of tags/annotations.

Approximate evaluations (Definition 4.9) are performed checking the approximate solution of an entity linking algorithm against the ground truth given by a dataset. For each problem, a different set of metrics has to be defined. First we will define basic metrics that account for how many tags/annotations are correct, and
how many are missing. On top of these basic metrics, we will define document-wise metrics that aggregate basic metrics in values comparable among documents. In other words, document-wise metrics account for the quality of a solution for a single instance. Lastly, since benchmarking datasets are formed by a collection of instances, we will define dataset-wise metrics, that aggregate measures for each document of the dataset.

Proposed metrics are the standard in Information Retrieval (Manning et al., 2008):

**Basic metrics:** true positives, false positives, false negatives

**Document-wise metrics:** Precision, Recall, $F_1$

**Dataset-wise metrics:** Micro-/Macro- Precision, Recall, $F_1$

but in our context they were generalized with respect to a *match relation*, a binary relation that captures whether or not an annotation or tag is correct, according to specific features. By generalizing with match relations, we open to the possibility of considering an annotation (or tag) part of a solution to be correct even though it is not equal to an annotation (or tag) in the ground truth, or vice-versa. Let’s first define a *match relation*:

**Definition 4.10.** A match relation $M$ on set $X$ is a relation that is symmetric and reflexive, i.e.

$$M(y, x) \iff M(x, y)$$

$$\forall x \in X. M(x, x)$$

In our applications, $X$ is either the set of tags or the set of annotations.

### 4.3.1 Cross-knowledge-base entity equivalence

There are cases in which annotator and dataset refer to distinct knowledge bases, for example an annotator might return links to Wikipedia whereas a dataset might contain references to DBpedia. Thankfully, knowledge bases are generally connected to each other, i.e. for each entity in a KB there exists an explicit link to the equivalent entity in other KBs, if such an entity exists.

In order to being able to test an annotator that links towards a KB against a dataset that links to another KB, we define an operator of cross-knowledge base entity equivalence:

**Definition 4.11.** Let $e_1$ and $e_2$ be two entities part of knowledge bases $kb_1$ and $kb_2$, respectively. The cross-knowledge base entity equivalence relation (in short, entity equivalence) is a binary relation among entities that is verified between $e_1$ and $e_2$ if and
4.3. Evaluation metrics for entity linking

only if $kb_1$ and/or $kb_2$ states an equivalence relationship between $e_1$ and $e_2$. The relation is represented with the symbol $\equiv$.

Relation $\equiv$ is reflexive, symmetric, and transitive.

In case $kb_1 = kb_2$, according to the definition, the relation is verified simply if the entity is the same (we assume that entity identity is reflexive!) or the knowledge base indicates an internal equivalence link.

Let us clarify with an example. The following four entities are referred to by their whole URI, that includes the knowledge base they are part of:

$$
e_1 = \text{en.wikipedia.org/wiki/Barack_Obama}$$
$$
e_2 = \text{en.wikipedia.org/wiki/Barack_Hussein_Obama}$$
$$
e_3 = \text{en.dbpedia.org/page/Barack_Obama}$$
$$
e_4 = \text{fr.dbpedia.org/page/Barack_Obama}$$

Each pair of entities $e_1, e_2, e_3, e_4$ verifies the entity equivalence, i.e.

$$e_1 \equiv e_2 \equiv e_3 \equiv e_4$$

Let us see why. Entity $e_1$ is the English Wikipedia entry about Barack Obama, and $e_2$ is another entry in the English Wikipedia that redirects to $e_1$. Wikipedia contains the information about the redirection, and we interpret it as an equivalence between $e_1$ and $e_2$.

Entity $e_3$ is the entry about Barack Obama in the English version of DBpedia. It has a property foaf:primaryTopic that cross-links it to $e_1$. We interpret this property as an equivalence relation between $e_3$ and $e_1$.

Entity $e_4$ is the entry about Barack Obama in the French version of DBpedia. It has a property owl:sameAs that links it to $e_3$. We interpret this property as an equivalence relation between $e_4$ and $e_3$.

Throughout this thesis, we assume that for each pair of knowledge bases, their entities are linked (this is verified for all knowledge bases mentioned in this thesis).

4.3.2 Basic metrics

Here follows the definition of the generalized basic metrics:

**Definition 4.12.** Let $s$ be the approximate solution for an instance, $g$ be the ground truth given by the dataset for that instance, and $M$ be a match relation. The following higher-order functions are defined:
4.3. Evaluation metrics for entity linking

true positives \( tp(s, g, M) = \{ x \in g \mid \exists x' \in s : M(x', x) \} \)
false positives \( fp(s, g, M) = \{ x \in s \mid \nexists x' \in g : M(x', x) \} \)
false negatives \( fn(s, g, M) = \{ x \in g \mid \nexists x' \in s : M(x', x) \} \)

Intuitively, true positives (TPs) are elements of the ground truth that have correctly been found by the algorithm; false positives (FPs) are incorrect elements that have been found by the algorithm, but should have not; false negatives (FNs) are missing elements that should have been found by the algorithm, but were not. All this, relatively to the aspects on which match relation \( M \) focuses.

Note that by using the equality as match relation, we obtain the classical definitions of true positives, false positives and false negatives.

A note on true positives

It is worthy to spend a few words on the definition of true positives. As per its Definition 4.12, true positives are “elements in the ground truth that are in a match relation with at least one element of the solution”, and not vice-versa. Let’s see why. First of all, it is important to consider that if the match relation, which is symmetric per definition, is also injective, i.e.

\[ M(a, x) \land M(x, b) \implies a = b \]

then

\[ \{ x \in g \mid \exists x' \in s : M(x', x) \} = \{ x' \in s \mid \exists x \in g : M(x', x) \} \]

hence the definition of true positives is equivalent to “elements in the solution that are in a match relation with at least one element of the ground truth”.

Anyway, match relations are not injective in general. We will see examples of non-injective match relations, such as the Weak annotation match in Section 4.4. In such cases, there might be more than one element of the solution that matches with a single element of the ground truth. Since we consider the ground truth to be the ideal solution, we count all elements of the solution that match with the same element of the ground truth as one. This way, in case an algorithm generates a high number of elements that match with the same element in the ground truth, its count of TPs is not affected. It follows that, in case of non-bijective match relations, the count of TPs plus the count of FPs is not necessarily the size of the solution.

An example of match relation

Suppose we want to measure the performance of a system solving A2KB on a dataset that provides a solution for the A2KB problem, thus \( s \) and \( g \) are sets of
4.3. Evaluation metrics for entity linking

annotations, but we are only interested in measuring the goodness of the retrieved entities, and not that of the spotted mentions. In this case, we define a match relation $M_e$ that only focuses on entities, and such that two annotations $a_1$ and $a_2$ match if and only if their entity is the same. We define $M_e$ as:

**Definition 4.13.** Let $a_1 = \langle m_1, e_1 \rangle$ and $a_2 = \langle m_2, e_2 \rangle$ be two annotations, the Entity annotation match $M_e$ is defined as:

$$M_e(a_1, a_2) \iff e_1 = e_2$$

Since $=$ is symmetric and reflexive, so is $M_e$.

An example of basic metrics built on top of $M_e$ is given in Figure 4.4. The document at issue is $d =$The novel begins in the Shire, where the Hobbit Frodo Baggins inherits the Ring from Bilbo. As Fantasy junkies certainly know, the context is that of The Lord of the Rings, a novel by J. R. R. Tolkien. Given the context, the ground truth for this document has been defined as a set of five annotations: the first one links the term *novel* to entity *novel*, a category-entity for all fictitious prose narratives; the two terms *the Shire* are linked to the fictional region of the Middle Earth (a named entity); term *Hobbit* is linked to the fictional race of half-men; etc.

In the example, an entity linking algorithm have generated an approximate solution for such a document. It consists of four annotations. Let’s see how the set of true positives, false positives and false negatives for match relation $M_e$ is defined. To find the true positives, we scan the set of annotations in the ground truth and search for matching annotations in the solution. The first annotation *novel* $\mapsto$ *novel* is identical to an annotation in the ground truth. Since match relations are reflexive, it is a true positive: the algorithm has found a correct annotation. The second annotation (*the Shire*) is in a match relation with the second annotation of the solution: the fact that their mention is different is not taken into consideration by $M_e$, as long as the entity is the same. The same applies to the fourth annotation *Frodo Baggins*. The third annotation has entity *hobbit*, but there is no annotation in the solution with such an entity, therefore it is a false negative: the algorithm has missed something. The same applies to the fifth annotation in the ground truth: despite having the same mention with an annotation in the solution, the entity differs. We have counted three true positives and two false negatives. To find the set of false positives, we need to scan the solution. The first three annotations have a match against one annotation in the ground truth (we already checked that, since match relations are symmetric). The last annotation in the solution has no matching annotation in the ground truth. We have counted one false positive.
4.3. Evaluation metrics for entity linking

Ground Truth

The \textit{novel} begins in the \textit{Shire}, where the \textit{Hobbit} Frodo Baggins inherits the \textit{Ring} from Bilbo

Output

Figure 4.4: Example showing matches according to $M_e$, that only takes entities into account, discarding mentions. In the example, there are three false positives, two false negatives, and one false positive. The precision is $\frac{3}{7}$, while recall is $\frac{3}{5}$.

4.3.3 Document-wise metrics

Based on the definition of TPs, FP and FNs, we can define document-wise metrics. They are the classic IR metrics of precision, recall and $F_1$ (Manning et al. 2008), generalized by a match relation:

\textbf{Definition 4.14.} Let $s$ be the approximate solution for an instance, $g$ be the ground truth given by the dataset for that instance, and $M$ be a match relation. Precision, recall, and $F_1$ are respectively defined as:

$$P(s, g, M) = \begin{cases} \frac{|tp(s, g, M)|}{|tp(s, g, M)| + |fp(s, g, M)|} & \text{if } |tp(s, g, M)| + |fp(s, g, M)| > 0 \\ 1.0 & \text{otherwise} \end{cases}$$

$$R(s, g, M) = \begin{cases} \frac{|tp(s, g, M)|}{|tp(s, g, M)| + |fn(s, g, M)|} & \text{if } |tp(s, g, M)| + |fn(s, g, M)| > 0 \\ 1.0 & \text{otherwise} \end{cases}$$

$$F_1(s, g, M) = \frac{2 \cdot P(s, g, M) \cdot R(s, g, M)}{P(s, g, M) + R(s, g, M)}$$

Values of precision, recall and $F_1$ are in $[0, 1]$. Intuitively, precision is the ratio of correct elements with respect to all elements provided by a solution; recall is the ratio of elements of the ground truth that have been found in the solution; and $F_1$ is the harmonic mean of precision and recall.
As seen in the definitions, there are special cases that must be handled separately. In the first scenario, \(|tp(s, g, M)| + |fp(s, g, M)| = 0\), meaning that the solution is empty. In this limit case we consider precision to be 1.0 (the highest), since no incorrect elements were found. In the second scenario, we have \(|tp(s, g, M)| + |fn(s, g, M)| = 0\), meaning that the ground truth is empty. In this case, we consider recall to be 1.0 (the highest), since no elements were left behind.

Note that, in case one of these two limit cases is met, the only scenario in which the value of \(F_1\) is non-zero, is if both conditions are met, in other words, if both the solution and the ground truth are empty. In this case, the ideal solution is found, and \(F_1 = 1.0\). In all other cases in which one of the two limit cases is met, we have \(F_1 = 0.0\), as either the solution is empty and the ground truth is non-empty (leading to \(P = 1.0, R = 0.0, F_1 = 0.0\)) or the solution is non-empty and the ground truth is empty (leading to \(P = 0.0, R = 1.0, F_1 = 0.0\)).

This limit cases are of particular importance for the evaluation of entity linking on queries (Chapter 6), in which documents are made of a few keywords and both solutions and ground truths are frequently empty.

In the previous example (Figure 4.4), precision is \(P(s, g, M_e) = 3/(3 + 1) = 0.75\), recall is \(R(s, g, M_e) = 3/(3 + 2) = 0.6\), and \(F_1(s, g, M_e) = (2 \cdot 3/4 \cdot 3/5)/(3/4 + 3/5) = 0.643\).

### 4.3.4 Dataset-wise metrics

We have everything in place to define dataset-wise Metrics as defined in Definition 4.7 that let us assess the performance of an entity linker over a whole dataset, featuring multiple instances. The following dataset-wise metrics serve the purpose.

**Macro-metrics**

**Definition 4.15.** Let \(D\) be a dataset, \(G\) be the list of ground truths for each document in \(D\), and \(S\) be the list of approximate solutions found by an algorithm, such that for document \(i = 1, \ldots, |D|\), the ground truth is \(g_i\) and the approximate solution is \(s_i\). We
4.3. Evaluation metrics for entity linking

Define macro-precision, macro-recall, macro-$F_1$ respectively as:

\[
P_{mac}(S, G, M) = \frac{\sum_{i=1}^{|D|} P(s_i, g_i, M)}{|D|}
\]

\[
R_{mac}(S, G, M) = \frac{\sum_{i=1}^{|D|} R(s_i, g_i, M)}{|D|}
\]

\[
F_{1mac}(S, G, M) = \frac{\sum_{i=1}^{|D|} F_1(s_i, g_i, M)}{|D|}
\]

Each macro-metric is simply the arithmetic average of a document-wise metric over the dataset. Intuitively, they give us an idea of what to expect when running the same algorithm on a new document, never seen before, similar to those in the dataset.

A note on the definition of macro-$F_1$

In some of the previous works, macro-$F_1$ has been defined as the harmonic mean of macro-precision and macro-recall, mocking the definition of $F_1$. We discourage such a definition. Though in some cases the value would be similar, in other cases it fails in the purpose of estimating the average $F_1$ of a document. This happens in case per-document results are heavily polarized towards precision (to the detriment of recall) or recall (to the detriment of precision). This scenario is frequent for datasets with short documents, for example queries. As an example, consider a dataset of 100 documents. For 50 of them, the algorithm has achieved $P = 1.0$ and $R = 0.0$, resulting in $F_1 = 0.0$. For the other 50, it has achieved $R = 1.0$ and $P = 0.0$, again resulting in $F_1 = 0.0$. Both macro-recall and macro-precision are 0.5, so their average is 0.5, but the average, expected, per-document $F_1$ is 0.0. The macro-$F_1$ as defined above would instead reflect this.

Micro-metrics

Macro-metrics refer to a whole dataset, but give expected values for the performance on a single document. In some applications instead, it is of greater importance to measure the performance on a dataset globally rather than on a single document. An example of such an application is the annotation of all tweets generated in the last 24 hours to see what entities have been mentioned the most. In this case, we are interested in the evaluation of dataset annotations, no matter what document provides them. To this end, micro-metrics are defined:

Definition 4.16. Let $D$ be a dataset, $G$ be the list of ground truths for each document in $D$, and $S$ be the list of approximate solutions found by a algorithm, such that for document
4.4 Problem-specific match relations

\[ i = 1, \ldots, |D|, \text{ the ground truth is } g_i \text{ and the approximate solution is } s_i. \] We define 

\[ P_{\text{mic}}(S, G, M) = \frac{\sum_{i=1}^{|D|} |tp(s_i, g_i, M)|}{\sum_{i=1}^{|D|} (|tp(s_i, g_i, M)| + |fp(s_i, g_i, M)|)} \]

\[ R_{\text{mic}}(S, G, M) = \frac{\sum_{i=1}^{|D|} |tp(s_i, g_i, M)|}{\sum_{i=1}^{|D|} (|tp(s_i, g_i, M)| + |fn(s_i, g_i, M)|)} \]

\[ F_{1\text{mic}}(S, G, M) = \frac{2 \cdot P_{\text{mic}}(R, G, M) \cdot R_{\text{mic}}(S, G, M)}{P_{\text{mic}}(S, G, M) + R_{\text{mic}}(S, G, M)} \]

Intuitively, micro-metrics consider the dataset as one big document, and counts the number of true positives, false positives and false negatives globally, discarding the reference to the document they belong to.

4.4 Problem-specific match relations

In this section, we will show how the metrics described in previous sections can be instantiated for specific problems and to separately capture specific aspects of an algorithm performance. We begin by outlining these aspects we want to focus on.

1. The ability of the algorithm in recognizing mentions (for problems Sa2KB and A2KB).

2. The ability of the algorithm in assigning an entity to each mention (for problems Sa2KB, A2KB and D2KB)

3. The ability of the algorithm in assigning tags to a document (Sc2KB, C2KB);

4.4.1 Metrics for the C2KB problem

For the C2KB problem, the match relation to use is quite straightforward. The output of a C2KB algorithm is a set of tags. Keeping in mind that a tag is codified as the entity it refers to, the following match relations are given:

**Definition 4.17.** Let \( T \) be the set of tags. Tag match is a match relation \( M_t \) on \( T \) between two tags \( t_1 \) and \( t_2 \). It is defined as

\[ M_t(t_1, t_2) \iff t_1 \equiv t_2 \]
4.4. Problem-specific match relations

The Tag match relation $M_t$ is the equality relation defined on tags. As such, it is reflexive and symmetric.

To measure the performance for the C2KB problem, metrics defined in definitions 4.16 (micro-measures) and 4.15 (macro-measures) can be instantiated with match relation $M_t$.

4.4.2 Metrics for the D2KB problem

D2KB output consists of a list of annotations. To compare two annotations, the following match relation is given.

**Definition 4.18.** Let $A$ be the set of annotations. Strong annotation match is a match relation $M_a$ on $A$ between two annotations $a_1 = \langle (p_1, l_1), e_1 \rangle$ and $a_2 = \langle (p_2, l_2), e_2 \rangle$. It is defined as

$$M_a(a_1, a_2) \iff \begin{cases} p_1 = p_2 \\ l_1 = l_2 \\ e_1 = e_2 \end{cases}$$

Just like the Tag match, this match relation is the equality relation defined on annotations, hence it is reflexive and symmetric.

To measure the performance for the D2KB problem, metrics defined in definitions 4.16 (micro-measures) and 4.15 (macro-measures) can be instantiated with match relation $M_a$.

It is worthy to note that, in case a D2KB algorithm returns no annotation for a mention received as input, recall is negatively affected (as one FN is counted), but in case a mention is associated the wrong entity, both precision and recall are negatively affected (as one FN and one FP is counted). In other words, the $F_1$ measures on D2KB penalizes over-confident algorithms that return incorrect entities with respect to conservative ones, that return nothing. In case we don’t want to penalize over-confident D2KB algorithms (intuitively, if we consider a wrong annotation as bad as no annotation at all), we should focus on recall measures.

4.4.3 Metrics for the A2KB problem

As in D2KB, the output of a A2KB problem is a set of annotations. The difference between these two problems is that in A2KB the mentions are not given as input and must be found by the algorithm.

A2KB can be measured by the same metrics we just defined for D2KB, based on the Strong annotation match $M_a$ (Definition 4.18).
By the definition, this match relation is verified only if the mention matches exactly. This approach leaves aside some cases of matches that should still be considered as right. Let us see an example of such a case. Suppose an A2KB algorithm is given as input the sentence *The New Testament is the basis of Christianity*. A correct annotation returned by the annotation system could be \( \langle \langle 4, 13 \rangle, \text{New Testament} \rangle \) (correctly mapping the mention *New Testament* to entity *New Testament*). But suppose the ground truth given by the dataset was another similar and correct annotation \( \langle \langle 0, 17 \rangle, \text{New Testament} \rangle \) (mapping the mention *The New Testament* to the same entity). Since the mentions differ, a metric based on the *Strong annotation match* would count one false positive and one false negative, whereas a relaxed evaluation should count only one true positive.

To treat these cases, Definition 4.18 can be relaxed as described in Definition 4.19 to match annotations with overlapping mentions and same entity:

**Definition 4.19.** Let \( A \) be the set of annotations. Weak annotation match is a match relation \( M_w \) on \( A \) between two annotations \( a_1 = \langle \langle b_1, l_1 \rangle, e_1 \rangle \) and \( a_2 = \langle \langle b_2, l_2 \rangle, e_2 \rangle \). It is defined as

\[
M_w(a_1, a_2) \iff \left\{ \begin{array}{l}
b_1 \leq b_2 \leq l_1 \lor b_1 \leq l_2 \leq l_1 \\
\lor b_2 \leq b_1 \leq l_2 \lor l_1 \leq b_2 \leq l_2 \\
e_1 = e_2
\end{array} \right.
\]

Note that \( M_a(a_1, a_2) \implies M_w(a_1, a_2) \). Intuitively, a Weak annotation match is verified if a Strong annotation match is verified (annotations have equal mentions) or, more generally, if the mentions overlap. Both are verified only if the entity of the annotations is the same. Relation \( M_w \) is reflexive and symmetric, but is not injective. It follows that there may be multiple annotations in the ground truth that match one single annotation in the ground truth, and vice versa (see [A note on true positives](Page 49)).

To measure the performance for the A2KB problem, metrics defined in definitions 4.16 (micro-measures) and 4.15 (macro-measures) can be instantiated with match relation \( M = M_w \) (for Weak annotation match) or \( M = M_a \) (for Strong annotation match).

### 4.4.4 Metrics for the Sc2KB and Sa2KB problems

As their non-scored version, Sc2KB and Sa2KB return respectively a set of annotations and a set of tags, with the addition of a likelihood score for each annotation/tag. In practice, the output of such systems is never compared against a ground truth of the same kind (in a ground truth, it’s a nonsense to assign a “likelihood score” to the annotations/tags). Hence, to measure the performance...
4.4. Problem-specific match relations

on the output of a Sc2KB and Sa2KB algorithm, it must be adapted (see the Se-
cion 4.2 about problem adaptations) to the problem the ground truth provides a
solution for, by discarding the score. The score can anyway be used to decide
what annotations/tags to keep and what to discard. Metrics presented in previ-
sous sections, with $M = M_t$ for Sc2KB and $M \in \{M_w, M_a\}$ for Sa2KB, can be used
for values of the threshold ranging in $[0, 1]$. In this range, we select the threshold
that maximizes $F_{1mac}$.

4.4.5 Metrics for Entity Recognition (for Sa2KB and A2KB)

Metrics defined for Sa2KB and A2KB instantiated with annotation match rela-
tions $M_w$ (weak) and $M_a$ (strong) measure the ability of an algorithm to find the
correct annotation, which involves, for each annotation, finding both the correct
mention (entity recognition) and the correct entity (entity disambiguation). As
explained at the beginning of Section 4.4, for a developer of entity linking algo-
rithms battling for overcoming her algorithm’s limitations, it is often more useful
to focus measures on one of these two aspects. The second aspect (entity disam-
biguation) is captured by the Entity annotation match $M_e$. In this section, we will
present a match relation that focuses on the aspect of entity recognition in Sa2KB
and A2KB, enabling us to answering the question: How much of the error in a
(S)A2KB algorithm is determined by the lack of mention recognition?

To answer this question, we can use a match relation that only checks the over-
ap of mentions, ignoring the entity:

Let $A$ be the set of annotations. Weak annotation match is a match relation $M_w$
on $A$ between two annotations $a_1 = (\langle b_1, l_1 \rangle, e_1)$ and $a_2 = (\langle b_2, l_2 \rangle, e_2)$. It is defined as

\[
M_w(a_1, a_2) \iff b_1 \leq b_2 \leq l_1 \lor b_1 \leq l_2 \leq l_1 \lor b_2 \leq l_2 \leq l_1 \lor b_2 \leq l_1 \leq l_2
\]

In other words, the match relation is verified if the mentions of two annota-
tions overlap. The entity is ignored.

To measure the performance on the problem of entity recognition, metrics
defined in definitions 4.16 (micro-measures) and 4.15 (macro-measures) can be
instantiated with match relation $M_m$. 

Definition 4.20. Let $A$ be the set of annotations. Mention annotation match is a match
relation $M_m$ on $A$ between two annotations $a_1 = (\langle b_1, l_1 \rangle, e_1)$ and $a_2 = (\langle b_2, l_2 \rangle, e_2)$. It
is defined as

\[
M_m(a_1, a_2) \iff b_1 \leq b_2 \leq l_1 \lor b_1 \leq l_2 \leq l_1 \lor b_2 \leq b_1 \leq l_2 \lor b_2 \leq l_1 \leq l_2
\]
4.5 Similarity between systems

Apart from the performance of an entity linking algorithm on a dataset, when developing an algorithm it can be useful to compare how similar it is to other algorithms.

The similarity between systems can be measured considering how similar their output are for the text documents contained in a dataset. In this section, the following definitions hold: let $D = [d_1, d_2, \ldots, d_n]$ be a dataset that contains $n$ documents, given as input to two systems $t_1$ and $t_2$ that solve problem $P \in \{\text{Sa2KB, Sc2KB, C2KB, A2KB, D2KB}\}$. Let $A = [a_1, a_2, \ldots, a_n]$ and $B = [b_1, b_2, \ldots, b_n]$ be respectively the output of $t_1$ and $t_2$, so that $a_i$ and $b_i$ are the solutions found respectively by $t_1$ and $t_2$ for document $d_i$.

4.5.1 Document-wise similarity metric

We propose a metric to check the similarity of the two sets of annotations $a$ and $b$ inspired by the Jaccard similarity coefficient (Jaccard, 1901), generalized to take into account the possibility that two annotations match according to a match relation, such as those presented in Definitions 4.17, 4.18, 4.19 and 4.20, even though not being equal. The following metric is proposed:

**Definition 4.21.** Let $a \subseteq X$ and $b \subseteq X$ be two sets, and $M$ be a match relation on set $X$. Similarity metric $Sim$ is defined as:

$$Sim(a, b, M) = \frac{|\{x \in a \mid \exists y \in b : M(x, y)\}| + |\{x \in b \mid \exists y \in a : M(x, y)\}|}{|a| + |b|}$$

Note that function $Sim$ is symmetric (in $a$ and $b$) and its value ranges in $[0, 1]$. Important features of $Sim$ are that $Sim(a, b) = 1$ if and only if, for all elements in $a$, there is a matching element in $b$, and vice-versa; $Sim(a, b) = 0$ if and only if there is not one single element in $a$ that matches with an element in $b$, and vice-versa. Unlike the Jaccard distance, $Sim$ is not a distance function since $Sim(a, b) = 1$ does not imply $a = b$, and does not verify the triangle inequality.

Note that we can instantiate $Sim$, with any of Tag match (for Sc2KB and C2KB systems whose output is a set of tags), Weak annotation match, Strong annotation match and Mention annotation match (for Sa2KB, A2KB, D2KB systems whose output is a set of annotations). As discussed later, the choice of match relation depends on what aspect we need to measure the similarity according to.

4.5.2 Dataset-wise similarity metric

The similarity of two lists of sets $A$ and $B$ can be defined as the arithmetic average of $Sim$ on the sets of the lists, giving the same importance to all the sets contained.
in the lists regardless of their size ($S_{\text{macro}}$) or as the overall “intersection” divided by the overall size, which gives more importance to bigger sets ($S_{\text{micro}}$). Formally:

**Definition 4.22.** Let $A$ and $B$ be two lists of elements $a_i, b_i \subseteq X$ and let $M$ be a match relation on $X$. The following definitions are given:

$$S_{\text{macro}}(A, B, M) = \frac{1}{n} \cdot \sum_{i=1}^{n} S'(a_i, b_i, M)$$

$$S_{\text{micro}}(A, B, M) = \frac{\sum_{i=1}^{n} (|\{x \in a_i \mid \exists y \in b_i : M(x, y)\}| + |\{x \in b_i \mid \exists y \in a_i : M(x, y)\}|)}{\sum_{i=1}^{n} (|a_i| + |b_i|)}$$

$S_{\text{macro}}$ and $S_{\text{micro}}$ share the same properties as $S'$: they range in $[0, 1]$, their value is 0 if and only if, for each $i \in [1, \cdots, n]$, there is not one single element in $a_i$ that matches with an element in $b_i$ and vice-versa, and their value is 1 if and only if, for each $i \in [1, \cdots, n]$ and for all elements in $a_i$, there is a matching element in $b_i$, and vice-versa. If $A = B$, then $S_{\text{macro}}(A, B, M) = S_{\text{micro}}(A, B, M) = 1$.

**4.5.3 Instantiating $S_{\text{micro}}$ and $S_{\text{macro}}$**

Let $S$ be any of $S_{\text{micro}}$ or $S_{\text{macro}}$. The meaning of the value given by this similarity metric depends on the match relation $M$ it is instantiated with. For C2KB and Sc2KB, in which the solution is a set of tags, the only defined matching function is $M_t$. In this case, $S$ is the fraction of how many of the entities found by both $t_1$ and $t_2$ are in common.

For all problems whose solution is a set of annotations (Sa2KB, A2KB, D2KB), any match relation $M \in \{M_a, M_w, M_m\}$ can be used, with the following meaning:

- $S(A, B, M_a)$ gives the fraction of common annotations (linking the same entity and having same mention).

- $S(A, B, M_w)$ gives the fraction of annotations appearing in both sets linking the same entity and having overlapping mention as an annotation in the other set.

- $S(A, B, M_m)$ gives the fraction of annotations appearing in both sets having overlapping mention with an annotation in the other set, no matter what is the entity.

- $S(A, B, M_e)$ gives the fraction of annotations appearing in both sets linking the same entity, ignoring the mentions.
4.5.4 Measuring true positives and true negatives similarity in detail

S-metrics can be used not only to check the whole output of two systems. The focus can instead be put on measuring how many of the true positives and true negatives two systems have in common, to see whether their correct spots and mistakes are similar or not. To do this, we can simply consider the value of $Sim$ computed on the subset of true positives or false negatives of $A$ and $B$.

**Definition 4.23.** Let $G = [g_1, \cdots, g_n]$ be the ground truth for a dataset, $g_i \subseteq X$ being the ground truth for instance $I_i$. Let $O = [o_1, \cdots, o_n]$ be the solution provided by an algorithm, $o_i \subseteq X$ being the solution for instance $I_i$. Let $M$ be a match relation on $X$. The following definitions are given:

$$T(O, G, M) = [tp(o_1, g_1, M), \cdots, tp(o_n, g_n, M)]$$

$$F(O, G, M) = [fp(o_1, g_1, M), \cdots, fp(o_n, g_n, M)]$$

Where $tp$ and $fp$ are the true positives and the false positives functions defined in Definition 4.12.

$T(O, G, M)$ and $F(O, G, M)$ are lists containing, for each instance $I_i$, respectively the true positives and the false positives contained in the output $o_i$ according to the match relation $M$ and the ground truth $g_i$.

The fraction of common true positives between outputs $A$ and $B$ is hence given by $S(T(A, G, M), T(B, G, M), M)$ whereas the fraction of common false negatives is given by $S(F(A, G, M), F(B, G, M), M)$, where $S$ can be either $S_{micro}$ or $S_{macro}$.

4.6 GERBIL: the BAT-Framework made easy

The BAT-framework has been implemented as a Java library providing interfaces for annotators and datasets, that a developer has to implement in order to run the benchmark. Later, we proposed GERBIL, an evaluation framework for entity linking built on top of the BAT-Framework. It uses the formal framework and the metrics defined in the previous sections of this chapter and, on the software side, it calls the BAT-Framework Java library to perform measures. GERBIL has been proposed in (Usbeck et al., 2015).

This software provides developers, end users and researchers with a friendly web interface that allows for the agile, fine-grained and uniform evaluation of entity linkers on multiple datasets. It is designed to be used by both developers,
who generally need to run a high number of experiments while testing the annotator under development, and users of entity annotators, who are interested in finding out what annotator better fulfill their needs.

GERBIL evaluates the performance of entity linkers treating them as black boxes, in facts it indirectly analyzes them by comparing the solution found by them against the ground truth. This means that GERBIL is agnostic with respect to possible training procedures. Nonetheless, it provides ways to indirectly verify the performance of single components of an entity linker.

For developers, GERBIL provides results that can be compared against the state of the art in entity linking, so that strengths and weaknesses can easily be spotted. This lets the developer come up with an informed agenda of where to focus her development.

A central feature of GERBIL is that of providing a persistent URL that describes a particular experiment. This is central for providing permanent, third-party, independent results. Through the experiment URLs, anyone can access experiment settings and results. The settings will include information about the annotator, the datasets it has been tested with, and the type of experiments that have been run. Results will include the outcome of the measures for that experiment. This includes the full, unambiguous setting of an experiment, and (as long as the annotator is deterministic), constitutes a method to provide full reproducibility of experiments.

The results of an experiment can also be downloaded in machine-processable format, allowing for the efficient querying and post-processing of evaluation results. Downloaded data employs DataID and DataCube to denote tools and datasets, and this ensures that results can be easily combined and queried while the exact configuration of the experiments remains uniquely reconstructible.

Experiments are archived in a central repository, and this allows end users to gather all pieces of information required to choose entity linkers for practical applications.

Once an annotator is plugged into GERBIL, it is very easy to evaluate it against all datasets GERBIL provides. Datasets mainly differ from each other for the domain from which documents are taken: queries, tweets, articles and books have different degrees of ambiguity, grammaticality, length, registry, and vocabulary. Testing an annotator on all these domains gives us an idea of the robustness of our annotator. On the other hand, a user of entity annotation systems, if interested in doing entity linking on a specific domain, would choose the annotator that best performs in that domain.
4.6.1 Extensibility

In the latest version (1.2.2), GERBIL supports 14 annotators, 19 datasets and seven predefined experiment types. Most importantly, GERBIL is open-source and extensible with new datasets and new annotators, and its architecture reflects this. Since the field of research of entity linking is continuously changing, we need to provide developers with the possibility to add new experiments, new measures and new problems to be evaluated. To address this, the following design choices were made:

**Extend with new entity linkers** Integration of new annotation algorithms is made easy and requires little time: GERBIL provides a wrapping interface that allows annotators to be evaluated via their REST interface.

**Extend with new datasets** Also datasets are easily integrated: GERBIL provides means to gather datasets for evaluation directly from data services such as DataHub.

**Extend with new measures** The evaluation measures used by GERBIL are implemented as interfaces. Thus, the framework can be easily extended with novel measures devised by the annotation community.

4.6.2 GERBIL architecture overview

GERBIL complies with a service-oriented architecture driven by the model-view-controller pattern (see Figure 4.5). Entity annotation systems, datasets and configurations like experiment type, matching relations and metrics are implemented as controller interfaces which are easily pluggable to the core controller. The output of experiments as well as descriptions of the various components are stored in a server-less database for fast deployment. Finally, the view component displays through a web interface the configuration options and renders experiment results delivered by the main controller communicating with the annotator and the database.

Experiments run in GERBIL can be configured in several manners. We present some of the most important parameters of experiments available in GERBIL.

4.6.3 Experiment types

GERBIL offers three types of experiments, one for each entity linking problem D2KB, A2KB and C2KB. An experiment on a problem measures the ability of an annotator to solve that problem. GERBIL uses the problems defined by the BAT-Framework. We remind the reader that, as by Definition 4.9 (Page 44), in
order to perform an experiment for problem $P$, we need a dataset for a problem $P'$ and an annotator that solves a problem $P''$ such that $P \preceq P'$ and $P \preceq P''$. An example configuration to perform a C2KB experiment would be to measure the performance of a Sa2KB annotator on a A2KB dataset. On such an annotator-dataset pair, also experiment A2KB could be performed.

One major formal update of the measures in GERBIL is that it also contemplates NIL annotations, i.e. special annotations having NIL as an entity. This kind of annotations represent the fact that a sequence of terms has been identified as a mention of an entity, but the entity is not present in the knowledge base. This has particular importance for finding emerging entities, which are entities that appear in some text but has not yet been included in the knowledge base. Though GERBIL accepts NIL annotations, they are currently not considered in evaluations. GERBIL just makes it possible to define new measures that take them into account.

Here follows the complete list of experiments, along with the algorithms that they execute in order to achieve the approximate solution for each instance, that is later compared against the ground truth.

**D2KB Experiment** (given a text document $d$ and a set of mentions $M$, find associated entities in $M \times K$, where $K$ is the set of entities in the knowledge base).

- In case the annotator natively solves D2KB, text $d$ and set of mentions $M$ are fed to the annotator, its result is considered as is;
In case the annotator natively solves **A2KB**, text \( d \) is fed to the annotator, that returns annotations \( A \). This result gets adapted to a solution for the D2KB problem by discarding annotations whose mention is not in \( M \);

In case the annotator natively solves **Sa2KB**, a per-dataset threshold \( t \) on the annotation score is set so to optimize the micro-\( F_1 \) score over the dataset, and the solution gets adapted in the same way as A2KB.

**A2KB Experiment** (given a text document \( d \), find the set of annotations \( A \)).

- in case the annotator natively solves **A2KB**, text \( d \) is fed to the annotator, its result is considered as is;
- In case the annotator natively solves **Sa2KB**, a per-dataset threshold \( t \) on the annotation score is set so to optimize the micro-\( F_1 \) score over the dataset.

**C2KB Experiment** (given a text document \( d \), find the set of entities \( E \) mentioned in it).

- in case the annotator natively solves **C2KB**, text \( d \) is fed to the annotator, its result is considered as is;
- in case the annotator natively solves **A2KB**, for each instance of the dataset, text \( d \) is fed to the annotator, that returns annotations \( A \). This solution is adapted by keeping all entities that appear in \( A \), and discarding the mentions;
- In case the annotator natively solves **Sa2KB**, a per-dataset threshold \( t \) on the score is set so to optimize the micro-\( F_1 \) score over the dataset, and the solution gets adapted in the same way as A2KB.

**Match relations and metrics**

As seen in Chapter 4, a match relation defines which conditions an element forming a solution has to fulfill to be considered correct. GERBIL experiments can be configured to measure metrics instantiated with any match relation.

GERBIL measures performance according to the same set of metrics and match relations defined for the BAT-Framework, namely precision, recall and \( F_1 \) in both their micro and macro versions (definitions 4.15 and 4.16). These metrics are offered as result of an experiment.

**Runtime performance**

In many scenarios, for example when processing big data, runtime is as important as the quality of the solution. GERBIL provides methods to measure the runtime
4.6. GERBIL: the BAT-Framework made easy

Results are provided also in tabular form.

Interlinked Output

The output of a GERBIL experiment consists in a table representing the values of the chosen metrics, computed on an annotator’s solution with respect to the ground truth provided by the dataset. Figure 4.6 shows an example web page showing the result of a GERBIL experiment. All this data is accessible through a permanent, time-stamped, URL. This way, GERBIL provides comprehensive, reproducible and publishable experiment results. The page shown at this URL provides a detailed description of each component of an experiment, together with sophisticated visualizations that allow for a quick comparison of tools and datasets on recently run experiments, without additional computational effort.

At the same page, GERBIL also provides machine-readable, inter-linkable results following the 5-star Linked Data principles. These statistical results are offered as JSON-LD28 RDF data using the RDF DataCube vocabulary.
The Data Cube vocabulary enables multi-dimensional, statistical information to be represented using the W3C RDF (Resource Description Framework) standard and published following the principles of linked data. This standard is compatible with the Linked SDMX standard for easy sharing of statistical results across organizations. Every GERBIL experiment is modeled as a qb:Dataset (according to the Data Cube vocabulary, a collection of observation having a common dimensional structure) containing the individual runs of the annotators on each benchmarking dataset as qb:Observation’s (according to Data Cube, a single observation with associated measured values). An observation provides both the experiment configuration and results.

Each observation features five qb:Dimension’s: experiment type, match relation, annotator, benchmark dataset and the time at which the experiment occurred. The six evaluation measures offered by GERBIL (micro and macro Precision, Recall and $F_1$) as well as the error count are expressed as qb:Measures.

To describe datasets, GERBIL makes use of DataID, a meta-data system developed specifically for describing properties of datasets and their physical manifestation. For example, they provide a way to describe what people and organizations have rights and responsibilities with respect to a dataset (e.g. by giving a way to declare a dataset license). Metadata include title, description and authors. The property dcat:Distributions gives a way to access the physical manifestation of a dataset, i.e. access to the benchmark dataset. Licenses are linked via the property dc:license as ODRL license specifications. Property dc:rights also include citations of the relevant publications.

To describe annotators in a similar fashion, we extended DataID for services. The class Service, to be described with the same basic properties as dataset, was introduced. Property datid:distribution provides the URL at which the service can be queried. Moreover, properties datid:Parameters and Configurations can be used to describe specific parameters of an annotator.

By providing well-structured configuration and results for our experiments, we enable third-party applications to keep track of advancements in entity linking.

### 4.7 GERBIL and the BAT-Framework: a preliminary evaluation

How do we evaluate a framework for the evaluation of entity linking algorithms? One of the aspects we decided to investigate is the practicability and convenience of the GERBIL framework. To do so, we investigated the effort needed by a user to use GERBIL for the evaluation of novel annotators. We asked entity linking sys-
tems developers to plug a novel annotator into GERBIL, and compared the time needed to do so against the time needed to implement the measures themselves.

Five developers with expert-level programming skills in Java participated to the survey. Each developer was asked to evaluate how much time she needed to write the code necessary to evaluate her annotator on a new dataset and how much time she needed to integrate it into GERBIL. Overall, the developers reported that they needed 1 to 4 hours to write ad-hoc evaluation code (it took 1-2 hours to four people, 3-4 hours for one person). But most importantly, all developers reported that they needed either the same or less time to integrate their annotator into GERBIL.

This result in itself is of high practical significance as it means that by using GERBIL, developers can evaluate on all datasets integrated into GERBIL (currently, 19) with the same effort they needed for one. In addition to that, results are computed in a standard, stable way, as experiments are performed in a way on which the community has achieved consensus, making measures comparable with other computed with GERBIL.

When asked how easy it was for them to integrate their systems into GERBIL, all developers reported they felt comfortable (an average of 4.0 on a 5-point Likert scale).

Though the small numbers do not confer this evaluation great statistical value, they still suggest that using GERBIL does not lead to any overhead, instead it significantly improves the quality and uniformity of results.

### 4.7.1 Adoption of the BAT-Framework and GERBIL

It is also interesting to measure the impact of the BAT-Framework and GERBIL on the entity linking research community. It is obviously hard to objectively measure it, and we will limit our analysis to giving a few numbers about the adoptions of the metrics we defined by works appearing in scientific publications. Despite being partial, this evaluation still gives interesting results.

To our knowledge, there is a total of 40 scientific publications proposing new entity linking methods and using the metrics defined by the BAT-Frameworks for evaluating them.

Concerning the adoption of GERBIL, maintainers of the website report that in the first four months of its public release (October 17th, 2014 to February 15th, 2015), a total of 1,824 experiments have been run on the platform, evaluating a total of 12,466 annotator-dataset pairs. These figures indicate a good

---

11 The number has been found by searching with Google Scholar papers citing the BAT-Framework and proposing a new entity linking algorithm, as of October 2016.
4.7. GERBIL and the BAT-Framework: a preliminary evaluation

adoption by the research community of our proposals in terms of formalization of entity linking.
5

Evaluation of Entity Linkers

In this chapter, we provide the reader with an overview of the performance of entity linking systems on text. We evaluate them using the GERBIL software and the metrics defined in the BAT-Framework. This way, we underline the strengths and weaknesses of each system. We will focus on documents of three domains: news, web pages, and short text, and for each domain, we will discuss results. Each of these three domains offer peculiar features and challenges. In this chapter we test entity linkers on natural language documents that feature, at least to some level, a grammatical structure. In Chapter 8.5 we will present results of entity linkers on search engine queries.

We perform our analysis on all entity linkers and all available benchmarking datasets, though for clarity we will report only results for the best performing ones. For a list and short description of entity linkers, see Section 3.1.1, and for a list of datasets and their characteristics, see Section 3.1.2. A quick reference to systems and datasets is given respectively in Tables 3.1 (page 19) and 3.2 (page 27). We encourage the reader to have a quick look at these tables.

5.1 Experimental setting

To ease the reading of this thesis and give a clear view of the results, we decided to focus only on the micro- \( F_1 \) measure. Macro- \( F_1 \) provided very similar results and identical ranking among systems. We also decided not to explicitly report measures of precision and recall, since for our purpose of ranking the annotators, \( F_1 \), which is a combination of precision and recall, is enough. Moreover, to limit the amount of information provided to the reader, we concentrated our analysis on the eight best-performing entity linkers, and did not report results for the others.

Concerning the match relation, we decided to report results based on the strong annotation match \( M_a \) for problem D2KB, the tag match \( M_e \) for problem C2KB, and the weak annotation match \( M_w \) for A2KB. The first two are obvious
5.1. Experimental setting

choices: in D2KB mentions are given as input, so there is no need to relax the mention match, and in C2KB we simply check if the entity is the same. For A2KB instead, mentions are not given as input, hence results better resembles a system performance if we consider as true positives annotations that link the same entity but may have different, overlapping mentions. This is guaranteed by the weak annotation match, that relaxes the mention match while keeping the entity match strong. See Section 4.4.3 for a more detailed discussion on metrics for the A2KB problem.

The datasets provide a ground truth for either A2KB or C2KB problems. For example, AIDA-CONLL is a dataset of type A2KB (it provides sets of annotations as ground truth, i.e sequences of terms linked to entities), while Meij dataset is of type C2KB (it provides sets of tags, i.e. entities linked to the whole text). Moreover, entity linkers natively solve Sa2KB, D2KB, or both. For example, AIDA natively solves both Sa2KB (finds mentions and link them to entities, with a score) and D2KB (given the mentions, links them to entities), while DoSeR only solves D2KB.

For each combination of problem/system/dataset we have one micro-$F_1$ value. Of course, not all combinations are possible: the Meij dataset, that provides a ground truth for C2KB, can only be used to test Sa2KB systems, and D2KB annotators can be tested only for the D2KB problem, but not A2KB or C2KB (for example, PBoH cannot be tested for A2KB).

Nonetheless, thanks to the hierarchy of problem adaptability introduced in Section 4.2 and the approximate evaluation defined in Definition 4.9, we can perform the following tests:

**C2KB problem (finding tags):** annotators solving Sa2KB (but not those solving D2KB) can be tested on all datasets;

**A2KB problem (finding annotations)** annotators solving Sa2KB (but not those solving D2KB) can be tested on all A2KB datasets (but not those providing C2KB ground truth);

**D2KB problem (linking pre-defined mentions)** annotators solving either Sa2KB or D2KB can be tested on all A2KB datasets (but not those providing C2KB ground truth).

5.1.1 A note on the fairness of D2KB tests

It is worthy to note an important characteristic about the fairness of experiments of type D2KB. As explained in Chapter 4, an annotator natively solving Sa2KB (find mentions and entities in a text) can also solve D2KB (given the mentions,
5.1. Experimental setting

link them to entities). In this scenario, the approximate adaptation consists in feeding the text to the Sa2KB annotator and retain only the annotations that overlap with a mention from the ground truth. This way, annotations in excess (that would count as false positives) are not counted, while wrong and good overlapping annotations are counted for what they are. Comparing results obtained with such an adaptation against results obtained with annotators that natively solve D2KB might not be completely fair, as the two systems are not fed with the same information: a native D2KB annotator is programmed to take into account the mentions as input of the problem, while a native Sa2KB annotator ignores them. In case a D2KB system is programmed to focus on the given mentions, e.g. with a mapping between mention and candidate entities broader than a dictionary match, this could give it an advantage with respect to agnostic Sa2KB systems. Unfortunately, the one we presented is the only way for guaranteeing a comprehensive comparison among Sa2KB and D2KB systems for what concerns the sub-problem of entity disambiguation. In any case, we argue that D2KB systems may obtain a higher score than their Sa2KB counterparts for this reason.

In the following experiments, for the D2KB tests we will use the D2KB interface for systems that provide it (namely AGDISTIS, PBoH, DoSeR, WAT 2), and apply the aforementioned adaptation for all other annotators (namely Babelfy, DBpedia Spotlight, TagMe 2, and Wikipedia Miner).

5.1.2 Employed software and APIs

The benchmark has been done using the most up-to-date APIs or software versions made available by their authors. Tests have been performed with GERBIL. For entity linkers that cannot be queried through an API nor can be locally deployed, we report results from their papers, evaluated with the same metrics defined in this thesis.

5.1.3 Performed experiments

We set up three experiments, each exploring the ability of the systems on a specific domain. Configurations for the experiments are summarized in Table 5.1.

Experiment 1: News

In Experiment 1, we analyze the performance of all entity linkers in news datasets, namely ACE2004, AIDA/CO-NLL, AQUAINT, MSNBC, N3-Reuters-128.

Each document in those datasets is a newspaper article and generally talks about topics of a single semantic field, related to a single event. Typical article length is between 1000-2000 characters. News are well-formed text, and this
5.1. Experimental setting

<table>
<thead>
<tr>
<th>Domain</th>
<th>Systems</th>
<th>Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp. 1 News</td>
<td>all*</td>
<td>ACE2004, AIDA/CO-NLL, AQUAINT, MSNBC, N3-Reuters-128</td>
</tr>
<tr>
<td>Exp. 2 Web pages</td>
<td>all*</td>
<td>IITB</td>
</tr>
<tr>
<td>Exp. 3 Tweets/short text</td>
<td>all*</td>
<td>Meij+, Kore50, Microposts2014, N3-RSS-500</td>
</tr>
</tbody>
</table>

Table 5.1: Configuration for the experiments, each performed on a set of datasets of a specific domain.

* Evaluations have been done by testing all feasible system-problem-dataset combinations.

+ The Meij dataset provides a ground truth of type C2KB, hence it can only be used for testing the C2KB problem. All other datasets provide a ground truth of type A2KB, hence they can be used for testing any of C2KB, A2KB, D2KB problems.

makes mention recognition relatively easy, as names and other concepts are usually referred to in a clean, extended, form. They provide a rich and uniform context that can be used to assist the disambiguation phase, giving a strong signal about which is the entity mentioned in the text.

Experiment 2: Web pages

In Experiment 2, we analyze the performance of all entity linkers in the only available dataset featuring web pages, namely IITB.

Compared to news, web pages are typically longer (4000 characters). They are generally well-formed text. Their context is more noisy, as it generally contains a lower degree of relatedness among mentioned entities. That is because a single web page may contain text about different, unrelated topics, and also presentational text such as menus or comments.

Experiment 3: Short text

In Experiment 3, we analyze the performance of all entity linkers on datasets of tweets (Meij, Microposts-2014) or short sentences (Kore50, N3-RSS-500).

Tweets are very short (typically around 80 characters) and provide little context, which makes disambiguation harder. Mention recognition is also harder compared to other domains, because tweets are not always grammatical, and feature many abbreviations, slang and colloquial references to people and concepts.

Short sentences offer similar challenges, but unlike tweets, they are well formed and grammatical. Entity mentions are extremely ambiguous, also because of
the lack of context, and can only disambiguated by heavily exploiting knowledge bases. For example, in the one-sentence document (from the Kore50 dataset):

```
Eric preferred to play Blues instead of Rock, so he joined Mayall’s band
```

the mention Eric has to be linked to Eric Clapton and Mayall has to be linked to John Mayall, and this has to be done exploiting the little amount of context the sentence offers.

## 5.2 Results of evaluation

### 5.2.1 Results for Experiment 1: News

Results for news datasets are reported in the first 8 columns of tables 5.2 (A2KB test), 5.3 (C2KB test), and 5.4 (D2KB test) at pages 78, 79 and 80.

**A2KB test on news**

Results for A2KB measure both the capacity of detecting mentions and linking them to the right entity. The most representative dataset for this experiment is AIDA/CoNLL, that features the highest number of documents with well-curated annotations. On the complete dataset, WAT2 is the best performing annotator, with a micro-$F_1$ score of 60.5%. The second-best annotator is Babelfy, with a score 5% lower than WAT 2. The third annotator is Wikipedia Miner, with a score 22% lower than WAT 2. DBpedia Spotlight and TagMe 2 reach poorer performances of 26% and 30%, respectively. Scores across the portions of AIDA/CoNLL are very similar, with a deviation lower than 2% for all systems, and this indicates the stability of results.

The ranking of systems is the same across other news datasets: WAT 2 is always the best-performing annotator, followed by Babelfy and Wikipedia Miner. The only exception to that is the AQUAIINT dataset, on which the ranking of WAT 2 and Wikipedia Miner are swapped.

**C2KB test on news**

In the C2KB test, mentions are not considered, and only entities are. As explained in Section 4.1.3 this problem is of fundamental importance for all applications involving representation of text as a set of topics (e.g. clustering and classification), where we need entities mentioned by a document but ignore where exactly they are mentioned. Obviously, the performance of a system on this test is heavily dependent on the performance of A2KB, and is typically higher than it (it is easier
5.2. Results of evaluation

to find that entity is mentioned somewhere in the document than to find the exact words that mention it).

Results for the C2KB test on news datasets are reported in the first eight columns of Table 5.3. Both the ranking of annotators and the values of micro-$F_1$ are very similar to the A2KB test. WAT 2 is the best-performing entity linker across all news datasets, followed by Babelfy and Wikipedia Miner.

In some cases, the micro-$F_1$ score on C2KB is higher than that of A2KB by a wide margin. This happens for the AQUAINT dataset, where all systems have a jump of at least 7% (13% in case of WAT 2) from A2KB to C2KB. This result underlines that in the AQUAINT dataset mentions are harder to find with respect to other datasets.

On the opposite, there are datasets, such as N3-Reuters-128, in which annotators get a higher score in A2KB compared to C2KB. This is only possible if entities mentioned multiple times in the same document are correctly spotted, while entities mentioned fewer times are not. In facts, the A2KB test counts each annotation as either a positive or a negative, while the C2KB test counts each tag only once per document.

D2KB test on news

The D2KB test lets us include a wider number of annotators with respect to previous comparisons, namely those that solve D2KB only: AGDISTIS, PBoH and DoSeR. This test focuses on the disambiguation algorithm of the annotator, which is the component on which most effort has been dedicated by researchers in recent years.

We note that the performance on D2KB is always higher than on A2KB. This is motivated by the fact that in D2KB, algorithms do not have to challenge with the problem of finding mentions, which are given as input, and the evaluation ignores surplus annotations found by the annotator whose mention is not included in the instance.

On this task it is harder to draw a straight conclusion about which disambiguation algorithm performs best. The three best performing systems are alternatively PBoH, DoSeR, and WAT 2, depending on the dataset, but their micro-$F_1$ value is very similar across datasets, with a difference smaller than 2% in most cases.

It is also worthy to note the performance of Babelfy, that, despite not being designed for D2KB and thus being penalized in this test, achieves the third position on AIDA/CoNLL, despite its micro-$F_1$ being significantly lower than other systems.

PBoH and WAT 2 have very good stability across datasets, with micro-$F_1$ ranging between 87% - 89% (PBoH) and 84% - 90% (WAT 2). The only exception to this
is the N3-Reuters-128 dataset, where their performance is respectively 76% and 79% (this is in line with previous experiments: this dataset seems harder than other news datasets). DoSeR has a less stable performance, ranging from 78% (AIDA) to 91% (MSNBC).

**News: conclusions**

“To treat news, what off-the-shelf annotator better fits my needs?” the reader might ask. Thankfully, the answer to that question is pretty straightforward: for problems A2KB and C2KB, WAT 2 is the best-performing annotator on news datasets, by a significant margin. It has very stable results, at least 5% higher than Babelfy, the second-best annotator, an all news datasets.

D2KB test has a more limited audience: it tells us what disambiguation approach works best, and what lines of research are better being pursued. Of course, D2KB systems can be plugged on top of an entity recognizer to build a A2KB annotator, and this may provide interesting results in many cases. WAT 2’s disambiguation algorithm has very good and stable performance, always reaching the second place. PBoH has slightly better results on AIDA and AQUAINT, but worse results on N3-Reuters-128 and ACE2004. DoSeR seems to feature the best disambiguation algorithm, reaching the best performance on three datasets out of five, but similarly to PBoH it has a lower stability than WAT 2. In conclusion, WAT 2, DoSeR and PBoH are the best-performing algorithms and represent the state of the art when it comes to disambiguation, even though there is no clear evidence of any of them clearly outperforming the others.

### 5.2.2 Results for Experiment 2: Web Pages

This experiment is performed on IITB, the only dataset featuring web pages. As the reader can see by looking at tables 5.2 (A2KB test), 5.3 (C2KB test), and 5.4 (D2KB test), results heavily differ from the previous experiment. WAT 2, the best annotator for news, reaches a poor performance on web pages, as does Babelfy (second best annotator on news). This is motivated by the fact that both systems aim at keeping a document-wise coherence, which does not exist in web pages, that may feature multiple semantic fields. The most robust annotator in case of the absence of global coherence is Wikipedia Miner, that reaches the micro-$F_1$ of 49%. Also DBpedia Spotlight reaches a relatively high performance, 5% lower than Wikipedia Miner, followed by TagMe 2.

Results for the C2KB test are very similar. Like on some news datasets (e.g. N3-Reuters-128), they are lower with respect to A2KB, and this indicates a high number of false positives or false negatives referring to the same entity.
5.2. Results of evaluation

Things change with the D2KB test, where mentions are given as input and the aforementioned issue does not apply. When it comes to benchmarking the disambiguation step on web pages, results are in line with news datasets: DoSeR is the best disambiguator, reaching comparable, but smaller performance with respect to news datasets. The second-best disambiguation algorithm is WAT 2, that has a similar decrease in performance with respect to news datasets. PBoH reaches a poorer performance and is overtaken by DBpedia Spotlight.

In conclusion, we argue that annotating web pages offers unique challenges, as the presence of distinct semantic fields needs to be modeled explicitly. Off-the-shelf, Wikipedia Miner is the best performing annotator, though it reaches limited performance. DoSeR seems the most promising disambiguation algorithm, reaching decent performance.

5.2.3 Results for Experiment 3: Short Text and tweets

Results for short text and single-sentence datasets are reported in the four leftmost columns (five for C2KB) of tables 5.2 (A2KB test), 5.3 (C2KB test), and 5.4 (D2KB test). We remind the reader that KORE50 and N3-RSS-500 are A2KB datasets consisting of one-sentence documents excerpted from their context, while Microposts2014 and Meij are datasets of tweets. Microposts2014 is of type A2KB while Meij is of type C2KB. For this reason, we will only be able to perform C2KB experiments on the Meij dataset.

Concerning single-sentence datasets (KORE50 and N3-RSS-500), WAT 2 is the best performing annotator when tested for the A2KB and C2KB problems. Babelfy has very similar performance, while other annotators achieve a significantly worse performance. When it comes to testing the disambiguation component (D2KB test), WAT 2 still reaches the most solid performance, granting it the second place on both datasets. Two annotators, namely Babelfy and TagMe 2, reach the first place respectively on KORE50 and N3-RSS-500, but significantly worse performance on the other dataset. PBoH also reaches interesting performance, though 10%/6% worse than WAT 2.

In conclusion, for situations where the context is lost, WAT 2 seems the most stable annotator, and this suggests that the idea of modeling candidate entities through their embedding helps, at least to some degree, to reconstruct the lost context.

Concerning tweet datasets (Meij and Microposts2014), results are less straightforward. Wikipedia Miner, Babelfy and DBpedia Spotlight alternatively gain the best performance, though there is a lack of uniformity, suggesting that each of them is over-fit on a specific dataset. WAT 2 performs really bad, achieving a micro-\(F_1\) up to 24% lower than the best annotator. Results are much higher on
the D2KB test, where the mention recognition phase is ruled out. Here, PBoH achieves the best performance, followed by WAT 2 (lower by 5-7%) and TagME 2.

In conclusion, concerning tweet annotation, we argue that the difference in numbers between A2KB/C2KB tests and the D2KB test indicates that the main challenge that comes with annotating tweets is mention detection, which is reasonable given that these kinds of messages are generally not well formed. This indicates the need to design mention recognizers specific for the domain of tweets. Off-the-shelf, Babelfy and Wikipedia Miner seem the best choices for annotating tweets.
### Table 5.2: Micro-$F_1$ scores (percentage) reported by GERBIL for the A2KB problem, using the weak annotation match $M_w$ as match relation. Each annotator has been tested on datasets of various domains: news, web pages and short text. For each dataset and each metric, we highlight in green, yellow and red respectively the first, second and third best system.
### 5.2. Results of evaluation

<table>
<thead>
<tr>
<th></th>
<th>ACE2004</th>
<th>AIDA/CoNLL-Complete</th>
<th>AIDA/CoNLL-TestA</th>
<th>AIDA/CoNLL-TestB</th>
<th>AIDA/CoNLL-Training</th>
<th>AQUAINT</th>
<th>MSNBC</th>
<th>N3-Reuters-128</th>
<th>IITB</th>
<th>KORE50</th>
<th>Meij</th>
<th>Microposts2014-Train</th>
<th>N3-RSS-500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Babelry</td>
<td>16.9</td>
<td>53.2</td>
<td>52.7</td>
<td>54.2</td>
<td>53.1</td>
<td>35.2</td>
<td>41.4</td>
<td>32.7</td>
<td>17.6</td>
<td>53.4</td>
<td>23.5</td>
<td>26.0</td>
<td>31.1</td>
</tr>
<tr>
<td>DBpedia Spotlight</td>
<td>7.9</td>
<td>25.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>29.9</td>
<td>19.8</td>
<td>11.4</td>
<td>41.8</td>
<td>27.0</td>
<td>30.1</td>
<td>18.7</td>
<td>10.6</td>
</tr>
<tr>
<td>TagMe 2</td>
<td>6.5</td>
<td>27.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>28.9</td>
<td>16.2</td>
<td>11.4</td>
<td>31.2</td>
<td>38.4</td>
<td>26.9</td>
<td>7.7</td>
<td>8.9</td>
</tr>
<tr>
<td>WAT 2</td>
<td>24.4</td>
<td>57.4</td>
<td>58.2</td>
<td>55.0</td>
<td>57.8</td>
<td>46.1</td>
<td>55.6</td>
<td>38.7</td>
<td>26.9</td>
<td>62.3</td>
<td>19.5</td>
<td>9.0</td>
<td>15.1</td>
</tr>
<tr>
<td>Wikipedia Miner</td>
<td>13.0</td>
<td>35.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>45.3</td>
<td>30.8</td>
<td>19.2</td>
<td>46.9</td>
<td>32.1</td>
<td>43.1</td>
<td>15.5</td>
<td>15.5</td>
</tr>
</tbody>
</table>

Table 5.3: Micro-$F_1$ scores (percentage) reported by GERBIL for the C2KB problem, using the tag match $M_e$ as match relation. Each annotator has been tested on datasets of various domains: news, web pages and short text. For each dataset and each metric, we highlight in **green**, **yellow** and **red** respectively the first, second and third best system.
<table>
<thead>
<tr>
<th>System</th>
<th>ACE2004</th>
<th>AIDA/CoNLL-Cm</th>
<th>AIDA/CoNLL-Ta</th>
<th>AIDA/CoNLL-Tb</th>
<th>AIDA/CoNLL-Tr</th>
<th>AQUAINT</th>
<th>MSNBC</th>
<th>N3-Reuters-128</th>
<th>ITTB</th>
<th>MSNC</th>
<th>N3-RSS 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGDISTIS</td>
<td>65.8</td>
<td>60.3</td>
<td>59.1</td>
<td>58.3</td>
<td>61.0</td>
<td>60.1</td>
<td>75.4</td>
<td>67.9</td>
<td>41.2</td>
<td>34.2</td>
<td>50.4</td>
</tr>
<tr>
<td>Babelfy*</td>
<td>63.2</td>
<td>78.0</td>
<td>75.8</td>
<td>80.4</td>
<td>78.0</td>
<td>72.3</td>
<td>78.2</td>
<td>58.6</td>
<td>57.1</td>
<td>73.1</td>
<td>50.6</td>
</tr>
<tr>
<td>DBpedia Spotlight*</td>
<td>70.4</td>
<td>58.8</td>
<td>54.9</td>
<td>57.7</td>
<td>60.0</td>
<td>74.0</td>
<td>69.3</td>
<td>56.4</td>
<td>65.4</td>
<td>37.6</td>
<td>56.4</td>
</tr>
<tr>
<td>TagMe 2*</td>
<td>81.9</td>
<td>72.1</td>
<td>69.1</td>
<td>70.6</td>
<td>73.2</td>
<td>76.3</td>
<td>76.0</td>
<td>59.3</td>
<td>57.2</td>
<td>57.3</td>
<td>59.1</td>
</tr>
<tr>
<td>Wikipedia Miner*</td>
<td>77.1</td>
<td>64.7</td>
<td>61.6</td>
<td>60.7</td>
<td>66.5</td>
<td>76.0</td>
<td>64.2</td>
<td>60.0</td>
<td>58.6</td>
<td>41.6</td>
<td>54.9</td>
</tr>
<tr>
<td>PBoH</td>
<td>87.2</td>
<td>86.7</td>
<td>86.6</td>
<td>87.4</td>
<td>86.6</td>
<td>89.5</td>
<td>76.5</td>
<td>62.5</td>
<td>61.7</td>
<td>74.2</td>
<td>73.1</td>
</tr>
<tr>
<td>DoSeR</td>
<td>90.7</td>
<td>-</td>
<td>78.4</td>
<td>-</td>
<td>84.2</td>
<td>91.1</td>
<td>85.0</td>
<td>74.1</td>
<td>-</td>
<td>-</td>
<td>75.1</td>
</tr>
<tr>
<td>WAT 2</td>
<td>89.5</td>
<td>86.0</td>
<td>84.1</td>
<td>85.2</td>
<td>86.7</td>
<td>85.2</td>
<td>89.9</td>
<td>79.4</td>
<td>69.1</td>
<td>71.5</td>
<td>68.5</td>
</tr>
</tbody>
</table>

Table 5.4: Micro-$F_1$ scores (percentage) reported by GERBIL for the D2KB problem, using the strong annotation match $M_s$ as match relation. Each annotator has been tested on datasets of various domains: news, web pages and short text. For each dataset and each metric, we highlight in green, yellow and red respectively the first, second and third best system. * these systems do not natively solve D2KB and can only be tested by applying the Sa2KB to D2KB problem adaptation. See Section 5.1.1.
Part II

Entity linking on queries
In this chapter, we introduce the problem of entity linking on web search engine queries, that is quickly emerging as a novel algorithmic challenge (Carmel et al., 2014). The problem is formally the same as that of entity linking on text documents (a query is a very short text document), but on this domain it presents particular challenges and rewards.

A critical component of a search engine is the one responsible for processing a query and guessing a user’s actual information need. For doing so, a search engine has limited per-query information (the few keywords of the query, the query session, the geo-location of the user), enormous amounts of contextual information (the indexed web, knowledge bases, query-logs and click-through logs), and very limited time to return the information or web documents most relevant to the query. The user expects nothing less than to be understood by the search engine, regardless of the validity of the clues given in her query. To meet this need, search engines nowadays rely on predictors that help them interpret, correct, classify and reformulate every submitted query in a split second before the actual document retrieval or question answering begins.

An important aspect of query understanding is entity linking, as defined in previous chapters. For example, answering the question How tall is the Tour Eiffel? depends on recognizing that words Tour Eiffel are a mention of the entity Tour Eiffel, having certain properties, including its height. The quality of this process is a crucial first step towards the development of reliable and robust algorithms for dialogue understanding and processing (see e.g. (Dalton et al., 2014; Manshadi and Li, 2009; Meij et al., 2014; Suchanek and Weikum, 2013; Yin and Shah, 2010)).

According to (Guo et al., 2009; Yin and Shah, 2010), over 70% of web search queries contain named entities (NEs). Entity analysis is becoming an important component of web search technology as search engines are trying to tighten the gap between the user intent and the response provided to her. This has the final aim of increasing precision, contextualization, and personalization of search re-
results. In particular, commercial search engines use named entities to determine the content of their result page, often directly showing information about named entities included in queries.

Linking mentions to entities in queries presents particular challenges with respect to doing so on longer text. The main reasons for this are that (1) queries are not in natural language, as they often provide no grammatical structure; (2) they can be misspelled; (3) they feature unreliable capitalization, word order and tokenization; (4) they are often inherently ambiguous; (5) they are short (usually two or three keywords), and thus provide no context that may assist disambiguation.

Noisy language makes it harder, with respect to well-formed text, to do entity recognition (i.e. map subsequences of terms against a set of candidate entities), which is a primary step for many traditional entity linking algorithms; and poor context makes it harder to resolve ambiguity for a specific subsequence of terms by choosing the right entity from a set of candidates.

Traditional entity linking on longer text is built on top of a set of assumptions that do not hold for queries, e.g. that the document’s grammatical structure can be parsed, or that words are written in the same form as they appear in a dictionary. For this reason, algorithms (e.g. Ferragina and Scaiella (2010); Yosef et al. (2011); Ratinov et al. (2011)) that achieve good results on long, well-formed text, achieve poor results on queries, as we will observe in Section 8.5. This motivates the development of ad-hoc algorithms for entity linking in queries.

6.1 The ERD Challenge

In March-June 2014, the SIGIR conference hosted ERD’14, the Entity Recognition and Disambiguation Challenge (Carmel et al., 2014), a workshop for pushing the research in entity linking. The challenge had two tracks: one for long documents (long track), and one for web queries (short track). In this chapter, we will focus on the short track. Participants were asked to build an entity linking system for queries. Those systems had to be queryable through a REST API, and this was intended to push into the development of systems that could be queried and experimented even after the end of the challenge. Participants were not provided with a benchmark dataset for training their systems, but they could build one on their own. During the development phase, challenging systems could be registered to a website that would test them against a dataset that served as development set; the website would return the macro-$F_1$ reached by the system for this dataset. This way, participants could have an idea of how their system performed. This stage lasted from March 25th to June 10th, 2014. After this period, the final test took place until June 20th. In this phase, systems were tested against the test
set, disjoint from the development set. This evaluation was the one employed to determine the final ranking of participants. Organizers decided not to publish the development set nor the test set, even after the end of the challenge, to avoid overfitting.

This challenge turned out to be an important turning point especially for entity linking on queries, that was never studied before.

A total of 18 teams coming from both industry and academy worldwide, participated to the final evaluation of the short-track challenge.

In collaboration with researchers at Google and at the Ludwig-Maximilian University of Munich, we participated to the challenge with our SMAPH System for Query Entity Linking, obtaining the highest $F_1$ score in the final test, and winning the challenge. The SMAPH system and its developments are presented in Chapter 8.

6.1.1 Query ambiguity interpretation and metrics

Challenge organizers observed that a fraction of queries were inherently ambiguous, thus can have multiple interpretations. An interpretation is a set of entities that describe a coherent meaning for the query.

For example, query `armstrong moon landing`, has only one possible interpretation: the user is searching for information about astronaut Neil Armstrong landing on the Moon, hence the only valid interpretation consists of two annotations: `armstrong` $\mapsto$ Neil Armstrong and `moon landing` $\mapsto$ Moon Landing.

Instead, query `armstrong moon` has at least two possible interpretations. The first valid interpretation is that the user is searching for information about when and how Neil Armstrong was on the Moon, the second is that she is searching for the song “Moon River” by Louis Armstrong. The corresponding interpretations are:

$$I_1 = \{\text{armstrong} \mapsto \text{Neil Armstrong}, \text{moon} \mapsto \text{Moon}\};$$

$$I_2 = \{\text{armstrong} \mapsto \text{Louis Armstrong}, \text{moon} \mapsto \text{Moon River}\};$$

To manage interpretations, ERD’14 organizers proposed a variant of the C2KB problem in which, for a single query, the algorithm has to return a set for each interpretation (thus, the returned value for a single query is a set of sets of entities).

The evaluation of the systems, the following measure was used. Let $\hat{A} = \{\hat{E}_1, \cdots, \hat{E}_n\}$ be the ground truth for a query ($\hat{E}_i$ are interpretations, i.e. set of entities), and let $A = \{E_1, \cdots, E_m\}$ be the algorithm response, then:

$$P = |\hat{A} \cap A| / |A|$$
\[ R = |\hat{A} \cap A|/|\hat{A}| \]
\[ F_1 = \frac{2 \cdot P \cdot R}{P + R} \]

The metric used for the final ranking is the macro-$F_1$ over the dataset.

Note that an interpretation returned by the algorithm is considered correct if and only if all its entities are exactly the same as those in an interpretation in the ground truth.

Despite the formulation of the entity linking problem with multiple interpretations, most participants (including us) decided to focus on the most common interpretation, and return a single set of entities for each query. Note that, in case of algorithms that return exactly one (possibly empty) interpretation, precision and recall can assume only two sets of values: if the returned interpretation is included in the ground truth, we have \( P = 1.0, R = 1.0/|\hat{A}|, F_1 = 2.0/(1.0 + |\hat{A}|) \); otherwise we have \( P = R = F_1 = 0.0 \).

### 6.1.2 Construction of the ERD dataset

Guidelines for building the dataset included three constraints:

1. **Use the longest mention for an entity**, e.g. for query “I live in redmond, WA”, the correct mention pointing to the city of Redmond is “redmond, WA” and not “redmond”;
2. **Annotate named entities only** (see Section 4.1.1 for a root-level categorization of entities);
3. **No overlapping annotations are allowed**;

As a method for building both development and test benchmark datasets, organizers of the challenge proposed a pooling technique: annotations derived from the output of the challenging systems were grouped into a pool of annotations, and crowdsourced judges were employed to decide whether an annotation was correct or not. Each annotation was judged three times. The use of judges from crowdsourcing guarantees a non-biased third-party evaluation. Conflicting annotations (groups of overlapping annotations that have been accepted by the judges) were manually adjusted by experts.

During the development phase, the pooling was done periodically: as more systems were tested, more annotations were added to the pool and their correctness had to be verified. After an update in the development set, all systems had to be tested against the new ground truth.

The pooling method is opposed to building an \textit{a priori} ground truth, and has one main drawback: it does not guarantee on the coverage of annotations. In facts,
if a correct annotation has not been found by any of the tested systems, it will not appear in the ground truth. While this does not affect the quality of measured precision (the whole system output is checked by humans), it has an impact on recall (as only a limited portion of the ground truth is available to check its presence in the result). Recall is not computed with respect to a ground truth, but with respect to a virtual super-system that merges the output of all systems participating to the challenge. This still guarantees a fair evaluation among systems, but does not say anything about the recall of the system with respect to the ideal output. Of course, this problem is mitigated by the number of participants: it is likely that the aggregated output of the 19 independently-developed systems participating to the challenge leaves behind only a negligible fractions of the correct annotations.

In Chapter 7 we will describe a method for building a benchmark dataset that overcomes this limitation, by employing human judges to build the pool of candidate annotations too.

### 6.1.3 Systems participating the challenge

Among the 19 systems (including ours) that participated to the challenge for the short track (entity linking on queries) we feature the three that achieved the highest score. Though it achieved the highest score among participants, we do not include in this brief list our system SMAPH, that will be covered in detail in Chapter 8.

**NTNU-UiS** ([Hasibi et al., 2014](#)) (Norwegian University of Science and Technology) uses a multi-stage framework, first recognizing entity mentions, next scoring candidate entities using a learning-to-rank method, finally, using a greedy algorithm to find all valid interpretation sets for the query.

**NTUNLP** ([Chiu et al., 2014](#)) (National Taiwan University) searches the query trying to match freebase surface forms with the longest-match strategy. The disambiguation step is built on top of TagME and Wikipedia.

**Seznam** ([Eckhardt et al., 2014](#)) (Seznam.cz Research, Czech Republic) uses Wikipedia and DBpedia to generate candidate annotations, than builds a graph of mentioned entities exploiting the link structure of Wikipedia. The disambiguation step is based on PageRank over this graph that assigns a score to each entity.
6.2 Entity linking on queries: state of the art

There is little prior work on entity linking for queries. Most research somehow concerning query understanding is aimed at Named Entity Recognition (Pasca, 2007) (the task of finding what keywords are mentions of named entities, without linking them to the entity), possibly associated to query intent discovery (Li, 2010) or query classification into pre-defined classes (Manshadi and Li, 2009; Guo et al., 2009; Eiselt and Figueroa, 2013). Some work has also been done with the aim of building a kind of grammar for queries, for example by POS tagging keywords or tagging them with a limited number of classes and other linguistic structures (Alasiry et al., 2012; Bendersky et al., 2011), or assigning a coarse-grained purpose to each segment (Joshi et al., 2014). Wei et al. (2008) present a method to do abbreviation disambiguation in queries.

Some of these works operate just on the query text (see e.g., Jain and Pennacchiotti (2011)), others use query logs (Alasiry et al., 2012), click through information (Li et al., 2011), search sessions (Du et al., 2010), top-k snippets and web phrase DBs (Bendersky et al., 2011; Hagen et al., 2012), and large manually annotated collections of open-domain queries to extract robust frequency or mutual-information features and contexts (Eiselt and Figueroa, 2013).

Another interesting line of research is that of query segmentation, having the goal of identifying important phrases and compound concepts as indivisible segments of a query. The authors of these works show that a search engine can exploit such hints to increase result precision, since documents that do not contain a segment’s words in proximity or even in the exact same order can be discarded (see e.g. Risvik et al. (2003); Jones et al. (2006); Tan and Peng (2008)). More recently, a series of papers by Hagen et al. (see e.g. Hagen et al. (2012)) proposed simple and effective scoring functions (for assigning scores to segments) using a weighted sum of normalized web-phrase frequencies (taken from the Google n-gram corpus). In particular, (Hagen et al., 2012) showed via a large experimental test that Wikipedia-page titles are very effective in query segmentation.

In literature there are also works that explicitly treat entity linking in queries, proposing algorithms to solve the problem. Blanco et al. (2015) design a fast and space-efficient entity-linking method leveraging information from query logs and anchor texts. This method aims at solving a ranking version of the A2KB problem where entity-mention pairs are scored and ranked. The system was evaluated on the Webscope L24 (Yahoo Search Query Log To Entities) dataset by concentrating on entities only, and not considering mentions (i.e., the C2KB problem). This way the authors did not determine the final annotation for a query but, rather, a ranked list of (possibly many) entities which was then evaluated by means of typical ranking metrics. The authors did not evaluate their system on the ERD’14
platform nor against the annotators participating in the ERD’14 Challenge.

Another interesting work is that presented by Hasibi et al. (2015) (a follow-up
of the NTNU-UiS system presented at the ERD’14 Challenge), where authors
describe a method to solve the problem of finding query interpretations (the prob-
lem defined for the ERD’14 Challenge). The method is based on three phases: (i)
candidate annotations are generated aiming at maximum recall, by exploiting two
sources: DBPedia and Google’s Freebase Annotations of the ClueWeb Corpora
(FACC); (ii) candidate annotations are assigned a score by using models for tradi-
tional document retrieval, in particular the Mixture of Language Models (MLM),
its score is combined with the commonness in a generative model; (iii) interpre-
tations are iteratively generated by picking non-overlapping annotations, starting
from those with higher score. The system is evaluated on a cleaned version of the
Webscope L24 dataset, where only explicitly mentioned, named entities are kept
(i.e., general concepts and implicit mentions are removed). The system achieves
a gain of around 5% in $F_1$ with respect to TagMe, which is used as a baseline. In
addition to the proposed system, Hasibi et al. (2015) have the merit of framing the
problems related to entity linking on queries, proposing the problem of semantic
mapping, i.e. finding the ranked list of entities that, even though not necessarily
having an explicit mention in the query, are related to it, for example the entity
Ann Dunham in the query obama mother.

In Chapter 8 we will present our proposal for joint mention-entity linking in
queries, namely we do not treat entity recognition and entity disambiguation as
two separate problems. This has been proposed by other authors for longer doc-
ments. Sil and Yates (2013) generate candidates with independent NER and
entity linking base systems, then re-rank joint candidates with a linear maximum
entropy model trained to optimize the L2-regularized conditional log likelihood
on a training set of documents. Guo et al. (2013) perform joint mention detection
and entity linking on tweets. They generate candidates on n-grams with a base
linking model and represent with a binary vector the entity mentions predicted.
Learning is cast as a structured prediction task, via SVM, using the Hamming
distance between the predicted and gold vector as the loss function.

6.3 Entity linking on queries: our contribution

In the next chapters, we will present our contribution to the task of entity linking
on queries. This can be divided into four main contributions:

- For the first time, we investigate the problem of identifying all entities (not
  only named entities) and their mentions in a query. This is the A2KB problem
  applied on open-domain web queries, with the whole Wikipedia as knowl-
edge base. This investigation is aimed at building a deeper understanding of the query compared to the approach of the ERD Challenge, as non-named entities in queries provides precious information.

- We build and release to the public the GERDAQ benchmark dataset of type A2KB and having queries as domain. GERDAQ features 1000 well-curated queries that have been labeled via a two-phase crowdsourcing process. GERDAQ meets the highest standards in terms of precision and recall even though A2KB on queries is a complex task even for humans. In Chapter 7 we thoroughly describe the process we followed to build the dataset.

- We design a state-of-the-art query annotator, SMAPH, for the A2KB problem on web queries. SMAPH uses a learning-to-rank model that jointly predicts the best complete A2KB annotation for the input query. The ranking crucially relies on a novel set of features we have designed that model the probability that a candidate entity is actually mentioned in the query, and thus should be “linked back” to it. In contrast to prior work, learning involves direct optimization of $F_1$, the measure of evaluation. SMAPH and its successors are described in Chapter 8.

- We present an extensive set of experiments that evaluate SMAPH and its updates on two benchmarks, the ERD’14 benchmark and our novel dataset GERDAQ, and show that it achieves state-of-the-art performance on both. The evaluation is presented in Section 8.5.
In this chapter, we present the methodology we employed to build the GERDAQ (General Entity Recognition, Disambiguation and Annotation in Queries) benchmark dataset. GERDAQ is a training and benchmarking dataset providing ground truth annotations for 1000 queries. Like all datasets, it can be either used for supervised machine learning or to evaluate query annotators.

GERDAQ is the result of a cooperation between the University of Pisa, University of Munich, and Google. The cost of creating this dataset was roughly $2000, funded by a Google Research award 2013. Queries are derived from the KDD Cup 2005 dataset (Li et al., 2005) and have been annotated by workers on Crowdflower, an on-line meta-crowdsourcing engine. The dataset has been released for free to the community, under a Creative Commons license, to assist the development of other systems and the research in query entity linking.

Even for experts on the subject, it is often hard to correctly annotate a query. The most complicated aspect of annotating a query is that it requires workers to think about what the user had in mind when she typed the query. The worker then has to spot the mention of an entity and pick the referenced entity from a knowledge base. Of course, no single worker has knowledge of the whole catalog of entities provided by a knowledge base and, even more so, no worker can interpret queries typed by any user, each with different background and culture.

In order to get as close as possible to this goal, we performed the annotation involving a high number of human raters. After selecting the queries to include in GERDAQ, the first phase was aimed at maximizing coverage, the second at refining precision.

www.crowdflower.com
7.1 Dataset construction workflow

7.1.1 Phase 0: Query selection

The queries forming GERDAQ instances have been sampled from the KDD-Cup 2005 competition dataset, which consists of 800,000 queries. These queries are taken from MSN search logs with some preliminary filtering.

First we cleaned the dataset by discarding the queries that looked like web addresses (i.e., those containing `www` or `http`), then we randomly sampled 1000 queries following a distribution that reflected their original word-count length. Figure 7.1 presents example queries from the random sample.

7.1.2 Phase 1: maximizing coverage

This phase aimed at maximizing the coverage of annotations, without considering their precision.

We set up a job on Crowdflower and, for each query, asked workers to spot annotations in queries, namely a mention and the entity referenced by it. Workers were instructed to make sure that the concept they were thinking of for a mention was actually the one described by the Wikipedia page they were going to link. Workers were asked for each query to spot as many mentions of Wikipedia entities as they could and link them to Wikipedia URLs. The job was set up so to accept only mentions that were actual substrings of the query and URLs that were existing English Wikipedia articles.

Quality of workers' contribution was covertly tested during the job execution in the following way. Queries of the GERDAQ dataset to be annotated were issued to the workers, but, among them, we inserted a set of 70 additional quality-control queries, in a way that workers could not distinguish them. Quality-control queries were chosen so to be of simple interpretation and not ambiguous. For those queries, we also built a ground truth. A worker response for a quality-control query was considered acceptable if the worker spotted at least one an-
7.1. Dataset construction workflow

<table>
<thead>
<tr>
<th>Query: south st philly stores</th>
</tr>
</thead>
<tbody>
<tr>
<td>stores ↦ Retail</td>
</tr>
<tr>
<td>south st philly ↦ South Street (Philadelphia)</td>
</tr>
<tr>
<td>philly ↦ Philadelphia</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query: cooking school &quot;mont st. michel</th>
</tr>
</thead>
<tbody>
<tr>
<td>cooking school ↦ Cooking school</td>
</tr>
<tr>
<td>mont st. michel ↦ Mont Saint-Michel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query: photos of stary night</th>
</tr>
</thead>
<tbody>
<tr>
<td>photos ↦ Photograph</td>
</tr>
<tr>
<td>stary ↦ Star</td>
</tr>
<tr>
<td>night ↦ Night</td>
</tr>
<tr>
<td>stary night ↦ The Starry Night</td>
</tr>
</tbody>
</table>

Table 7.1: Annotations proposed by workers in Phase 1 for three queries. Right column indicates how many workers spotted an annotation.

notation of the ground truth built by us, and any number of annotations not included in the ground truth. The form let workers input up to 10 annotations per query. This way, we accepted partial knowledge but require the workers to have non-empty knowledge on easy queries. When workers issued wrong responses for quality-control queries, they were prompted with an error message explaining the correct annotation process, but workers who persisted in failing were permanently excluded from the job, and their previous responses ignored.

Since no worker has full knowledge of all domains, high coverage could only be reached if this job employed as many workers as possible, hopefully with different background and culture. For this reason, we collected at least 10 (in some cases, 11) responses for each query, and then aggregated it. The job completed in a few hours and collected a total of 10,038 responses (not counting those given by unreliable workers and those given for quality-control queries). A response includes a set of annotations for a single query. The workers found a total of 3,197 distinct annotations (3.2 per query). A total of 271 workers took part in the job; they processed 37 queries each on average.

Table 7.1 shows the aggregated output from Phase 1 for three queries. Each annotation is associated with the number of workers that found it. As figures from a later step of refinement show (see Tables 7.2 and 7.3), the fact that an annotation was found only by a few workers does not determine that it is wrong. For example, in query south st philly stores, the user intent is to get information about stores in South Street, a street in Philadelphia that is one of the city’s
7.1. Dataset construction workflow

Table 7.2: Distribution of judgments over the 3,197 distinct annotations. Read the first column as “1,048 annotations were found by a single worker”

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>≥10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1048</td>
<td>384</td>
<td>291</td>
<td>229</td>
<td>215</td>
<td>190</td>
<td>189</td>
<td>234</td>
<td>218</td>
<td>199</td>
</tr>
</tbody>
</table>

largest tourist attractions. 6 out of 11 workers correctly recognized that st philly refers to that street, but only three found that the keyword store refers to retail stores. Nonetheless, the annotation is correct. On the other hand, in query photos of stary night, the user intent is to find pictures of the night view of the sky in which stars are visible. Entities Photograph, Star, Night are correctly found, but 6 out of 11 workers identified stary night as being a mention of Van Gogh’s famous painting The Starry Night. Both are reasonable interpretations, and a later phase of refinement will have to choose which is the most reasonable according to common sense.

Table 7.2 shows the distribution of how many workers spotted the same annotation: about half of the annotations were found by one or two workers but, as we will see later, most of them were nonetheless judged correct in Phase 2.

7.1.3 Phase 2: refining precision

In Phase 2, we discarded the portion of bad annotations found by workers in Phase 1, keeping only the good ones. We created a second job on CrowdFlower, asking workers to judge, in a scale from 1 to 10, the likelihood that a certain annotation found in Phase 1 was correct. Workers were prompted with questions like: In the query armstrong moon, how likely does armstrong refer to the entity Neil Armstrong? Workers were also provided with an abstract of the candidate Wikipedia page (e.g. Neil Alden Armstrong was an American astronaut...), to better tell appropriate entities from wrong ones.

Similarly to Phase 1, also Phase 2 featured covert quality-control. For the same set of 70 queries used for quality-control in Phase 1, we manually generated 76 correct and 69 wrong annotations. Similarly to Phase 1, we chose simple, unambiguous cases. These annotations were covertly provided to workers that had to judge them. To have their responses on quality-control instances considered acceptable, workers had to assign a score between 1 and 4 to wrong annotations, and a score between 7 and 10 to correct annotations. Workers failing in recognizing multiple quality-control annotations have been excluded from the job and their contribution not taken into account.

Each query was processed by at least 3 workers, for a total of 390 workers who
7.1. Dataset construction workflow

Table 7.3: Distribution of average annotation scores. Example: 81 annotations received an average score in \([0.1, 0.2]\). 375 annotations were given the maximum score by all workers (last column).

```
<table>
<thead>
<tr>
<th>Score</th>
<th>0</th>
<th>.1</th>
<th>.2</th>
<th>.3</th>
<th>.4</th>
<th>.5</th>
<th>.6</th>
<th>.7</th>
<th>.8</th>
<th>.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>53</td>
<td>81</td>
<td>94</td>
<td>134</td>
<td>169</td>
<td>251</td>
<td>325</td>
<td>369</td>
<td>633</td>
<td>712</td>
<td>375</td>
</tr>
</tbody>
</table>
```

Table 7.3: Distribution of average annotation scores. Example: 81 annotations received an average score in \([0.1, 0.2]\). 375 annotations were given the maximum score by all workers (last column).

<table>
<thead>
<tr>
<th>Query: south st philly stores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
</tr>
<tr>
<td>stores ⟷ Retail</td>
</tr>
<tr>
<td>philly ⟷ Philadelphia</td>
</tr>
<tr>
<td>south st philly ⟷ South Street (Philadelphia)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query: cooking school &quot;mont st. michel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
</tr>
<tr>
<td>cooking school ⟷ Cooking school</td>
</tr>
<tr>
<td>mont st. michel ⟷ Mont Saint-Michel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query: photos of stary night</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
</tr>
<tr>
<td>photos ⟷ Photograph</td>
</tr>
<tr>
<td>star ⟷ Star</td>
</tr>
<tr>
<td>night ⟷ Night</td>
</tr>
<tr>
<td>stary night ⟷ The Starry Night</td>
</tr>
</tbody>
</table>

Table 7.4: Score assigned by workers to annotations in Phase 2. Central column indicates how many workers spotted an annotation, right column is the average of the scores assigned by Phase 2 workers.

processed 26 queries each on average, and collected a total of 9,612 annotation scores. The distribution of average scores assigned to annotations in Phase 2 is shown in Table 7.3. As numbers clearly state, workers of Phase 2 considered correct a big fraction of annotations, even among those that were found by a limited number of workers in Phase 1. Sampled output from Phase 2 is shown in Table 7.4.

7.1.4 Phase 3: manual refinement by experts

A non-trivial issue was that of defining a threshold on the average score to discard bad annotations. We decided to leave this job to an expert human judgement. By randomly sampling annotations, we observed that all annotations with a score smaller than 0.58 were wrong, and thus to be discarded. Similarly, all annotations
with a score above 0.65 were correct. Annotations in range 0.58 – 0.65 (a few dozens) were manually double checked for correctness until complete agreement between two members of the research team developing GERDAQ was reached.

Of course, there can be queries with two or more conflicting (i.e. overlapping) annotations, all with scores high enough to be accepted. This happened for 90 mentions over a total of 2043. This can be due to three factors:

1. **Actual ambiguity of the query**, that brings uncertainty on what entity to link. For example, in query *armstrong moon*, mention *armstrong* can either be interpreted as mentioning *Neil Armstrong* or *Louis Armstrong*. Since we are building a dataset that labels the most common interpretation, we solve these cases by keeping the annotation with highest score.

2. **Unclear mention but unambiguous entity**, that brings uncertainty on what mention to link to the entity. We solve these cases by choosing the annotation with highest score among those linking the same entity.

3. **Subsuming entities**, for example in query *president of u.s. 2006*, mention *president of u.s.* mentions the institution *President of the United States* but an incorrect annotation might link mention *u.s.* to *United States*. We solve these cases by keeping the annotation whose mention subsumes the others.

In case of actual query ambiguity, the GERDAQ dataset also features secondary interpretations found by the workers, which is available for future work, though not considered in this thesis.

After the refinement, the dataset consists of 2043 distinct annotations (an average of 2.0 annotations per query). This constitutes the final GERDAQ dataset (see Table 7.5 for basic statistics).

We randomly split GERDAQ into *training set* (500 queries), *development set* (250 queries), and *test set* (250 queries). We encourage researchers to train and tune their systems on the first two portions and keep the test set for the final evaluation, on which to compare query entity linkers. We released the dataset under a Creative Commons license.

---

2Dataset at [http://acube.di.unipi.it/datasets/](http://acube.di.unipi.it/datasets/)
### Dataset construction workflow

#### Table 7.5: GERDAQ dataset statistics for each portion.

<table>
<thead>
<tr>
<th>Portion</th>
<th>Queries</th>
<th>Q. with ( \geq 1 ) anns</th>
<th>Avg. anns per non-empty q.</th>
<th>Avg. anns per query</th>
<th>Avg. query length (chars)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>500</td>
<td>446</td>
<td>2.10</td>
<td>1.86</td>
<td>25</td>
</tr>
<tr>
<td>Development</td>
<td>250</td>
<td>221</td>
<td>2.05</td>
<td>1.81</td>
<td>22</td>
</tr>
<tr>
<td>Test</td>
<td>250</td>
<td>222</td>
<td>1.95</td>
<td>1.73</td>
<td>23</td>
</tr>
</tbody>
</table>

The second column indicates the amount of queries in a portion; the third column indicates the number of queries having at least one annotation (non-empty queries); the fourth column indicates the average number of annotations among non-empty queries while the fifth indicates the same quantity among all queries; the last column indicates the average query length in characters.
SMAPH: a System for Entity Linking on Queries

SMAPH is our proposal for entity linking on web search engine queries. In this chapter, we describe it in three variants. As observed in previous chapters, and as proved by the evaluations performed in Section 8.5, traditional entity linking algorithms developed for long texts work poorly on queries, and this motivates the development of entity linkers specific for the domain of queries. The SMAPH system is currently the state of the art in terms of quality of results, as tests among all publicly available datasets show.

The most challenging aspect of understanding queries is their limited size, their lack of context, and the variety of styles in which people write them, that makes it hard to discover a grammar or any other structure.

Search engines have the impressive duty of organizing the information provided by the Web and point users to the portion of knowledge they expressed through their query.

Given a query, search engines provide the list of Web documents that are pertinent to it, but they also provide extremely important accessory information: for each retrieved Web document, they include a snippet of text extracted from those documents. In snippets, query keywords appear (usually shown in bold form) as they are written in well-formed web pages. This way, snippets provide a context for the query, locating query keywords in portions of the web page text. We can leverage on this context to disambiguate query mentions: query keywords, often misspelled or chosen by people with different backgrounds, are resolved into general, clean forms. These snippets are ranked by relevance, with respect to the query, by means of sophisticated algorithms that leverage huge indexed document collections (the whole web), link graphs and query log analysis.

The approach used in SMAPH is built on top of this information provided by search engines. In facts, we propose to deal with the challenges posed by queries by piggybacking on web search engines. Piggyback on search engines
was first introduced by Rüd et al. (2011), where authors claim search engines are the available resources that most closely cover world knowledge. For a high-level overview of search engines and the Piggyback approach, we point the reader to Section 2.3.

Search engines tend to be robust to the issues raised by queries because they have been designed to deal effectively with surface problems like misspellings, tokenization, capitalization and word order.

In this chapter we start by describing SMAPH-1, the first version of the SMAPH system, winner of the web queries track at the ERD 2014 Challenge (Cornolti et al., 2014). SMAPH-1 solves the C2KB problem, i.e. it finds entities mentioned by a query, without finding their mentions. Though the system achieved the Top-Scoring Award at the ERD Challenge, we sought space for improvement. We perform a thorough error analysis of SMAPH-1, which let us identify, among others, a limitation related to the abundance of false negative entities, leading to low recall. These are entities that, despite being explicitly mentioned by the query, are associated to poor signals, and thus discarded by SMAPH-1. To overcome this issue, we had to make sure that entities found by the system get boosted in case they are bound to query keywords. We decided to overcome this limitation by explicitly modeling this problem as an A2KB task, and present two upgrades of the SMAPH-1 system: SMAPH-S, that deals with individual annotations, and SMAPH-2, that performs a collective disambiguation, considering groups of annotations as candidate solutions.

Furthermore, we consider not only named entities (NEs) but also general concepts, which has been recognized as a key feature of modern query-annotation tools (Dalton et al., 2014).

We evaluate our systems on two datasets: GERDAQ, the benchmarking dataset described in Chapter 7 and the dataset employed for the ERD’14 challenge. On both datasets, we show that our proposed system, SMAPH-2, achieves effective results on the A2KB problem with respect to strong baselines, and that it is the state of the art, since it outperforms the winner of ERD’14 and all other annotators that were proposed and tested on ERD’14 after the end of the challenge (i.e. since August 2014).

All SMAPH systems share the first step of candidate entities generation, based on the piggyback approach, namely extracting candidate entities from the snippets of search results returned by a search engine for a query. Snippets are annotated with a robust text annotator designed for short texts (TagME-WAT by Piccinno (2016)). This alleviates the language noise and brevity of queries by offering a longer and cleaner context. SMAPH-S and SMAPH-2 adopt a link-back approach that prunes the set of candidate entities and keeps only those more likely to ac-
8.1 Candidate entity generation

The component of candidate entity generation is shared among all variants of SMAPH. Its purpose is that of finding a set of candidate entities for input query $q$, on which the solution will be built. The reader might find it useful to follow the flow chart in Figure 8.1, where candidate entity generation is shown in the upper part.

For all versions of SMAPH, this is the only phase in which new entities come into play, even though some of them will be discarded in later steps. Given that, this phase aims at maximizing the recall of the candidate set, which affects the potential recall of the eventual solution for $q$.

Candidate entity generation piggybacks on the results returned by the public API of Bing, and works in two phases.

8.1.1 Phase 1: Fetching

First, query $q$ is issued to Bing, enabling the spelling correction feature. This way results are not affected by spelling errors possibly present in the query. The first 25 results returned by Bing, in particular their URLs and snippets, are taken into consideration for later processing.

Secondly, a new query $q^w$, composed by $q$ concatenated with the word wikipedia, is issued to Bing and the top 10 results are taken into consideration for later processing. This query boosts results from Wikipedia. Note that search engines also support domain-restricted queries that could be used to search among Wikipedia pages only. We decided not to use this type of search because it tends to return articles loosely related to the actual query. Instead, by simply appending word wikipedia, we give the search engine a soft suggestion about the kind
8.1. Candidate entity generation

Figure 8.1: Overview of the SMAPH systems. Continuous line indicates workflow; dashed lines indicate generated data. The first phase, *candidate entity generation*, is shared among the three systems. Solution generated by SMAPH-1 consists of the pruning of candidate entities. SMAPH-S and SMAPH-2 share the step of generation of candidate annotations: SMAPH-S ranks them and builds a greedy solution, while SMAPH-2 generates bindings, rank them, and returns the highest-rank binding as solution.
of pages we are interested in, but also obtain, as their rank, a signal of their pertinence to the query.

The output of this phase is a list of 25 snippets and 25 URLs from the first search, and 10 URLs from the second search.

### 8.1.2 Phase 2: Candidate-entity generation

Entities are drawn from three sources.

**Source 1.** Among the top-25 URLs returned by the search of \( q \), we find those that point to Wikipedia pages. Corresponding entities form the set \( E_1 \).

**Source 2.** The same is done with the top-10 URLs returned by the search of \( q'' \). These entities form the set \( E_2 \).

**Source 3.** Snippets of the top-25 results of the first search are, independently from each other, annotated with the text annotator WAT\(^1\) (see Section 3.1.1 for a description). As seen in Section 5.2.3, WAT has very good performance on annotating sentences excerpted from longer documents, and snippets are indeed well-formed excerpts from web pages of a few dozen terms. WAT finds mentions in the snippets and links them to Wikipedia entities, by exploiting the context provided by the snippet. For each snippet, WAT returns a set of annotations. We only keep the ones that overlap with a bold-highlighted substring of the snippet, as those substrings are the way in which query terms appear in web pages.

While Source 1 and 2 are straightforward to understand, explanation of Source 3 may benefit from an example. Say \( q = \text{armstrong mon lading} \), a query presenting a misspelling in the writing of the term moon and landing, and an incomplete, ambiguous reference to Neil Armstrong. Figure 8.2 shows the top 5 results returned by Bing when searching \( q \). Consider as an example the third snippet:

> Video embedded – Armstrong was a NASA astronaut and the first man on the moon or, more accurately, the first man to set foot on the moon. He...

Note that term mon have been corrected to moon. In a way, query terms have been “translated” into natural language, since they are shown as they appear in an existing web page written by a human. They are also put into a context that assists disambiguation.

\(^1\)During the development of SMAPH, we tried several other annotators other than WAT, but they yielded worse performance when annotating snippets.
8.1 Candidate entity generation

Figure 8.2: Top 5 results returned by Bing for query armstrong mon lading

When the snippet is fed to WAT, it searches the text for potential mentions, finding four sequences of tokens: Armstrong, NASA, astronaut, and man on the moon. The first mention, Armstrong, is ambiguous in that by itself it could refer to Neil Armstrong, Louis Armstrong, Armstrong County, Pennsylvania or other entities, but it is located in the same context (the same web page) with other terms such as NASA and astronaut, and WAT uses this context to disambiguate it into Neil Armstrong. This is repeated for all other mentions. WAT also makes mistakes, for example, it would link man on the moon to the 1999 movie starring Jim Carrey. In total, WAT would find four annotations:

Armstrong $\rightarrow$ Neil Armstrong
NASA $\rightarrow$ National Aeronautics and Space Administration
astronaut $\rightarrow$ Astronaut
man on the moon $\rightarrow$ Man on the Moon (film)

Despite being used by WAT in the disambiguation process, mention astronaut does not overlap with any bold portion of the snippet, hence it is discarded. We remind the reader that text in the snippet rendered in bold form is a re-writing
of a sequence of query terms, as opposed to other text drawn from the web page but not present in the query. The lack of overlap between mention and bold text indicates that, though entity Astronaut is probably related to the query, it is not explicitly mentioned by it, hence it is discarded.

The analysis of this snippet contributes to the construction of set $E_3$ by adding entities Neil Armstrong, National Aeronautics and Space Administration, Man on the Moon (film). More entities will be added by analyzing the other snippets.

### 8.1.3 Coverage of the three sources

As said, the objective of the candidate entity generation was to reach a high coverage, at the cost of a low precision, as in this step we define the upper bound on the final recall achieved by SMAPH systems. In Section 8.5.3 we will show that the coverage of the union $E_1 \cup E_2 \cup E_3$ is 87.6% for all entities and 94.4% for named entities.

### 8.2 SMAPH-1: Individual entity pruning

After the set of candidate entities $E_1 \cup E_2 \cup E_3$ is generated, SMAPH-1 chooses a subset of them, and returns this subset as a result. The choice of the subset is implemented with a binary classifier that judges each candidate entity independently and decides whether it should be included in the result or not.

#### 8.2.1 Entity pruning via SVM binary classification

As binary classifier we used a Support Vector Machine (SVM) non-linear classifier with an RBF kernel. We recommend the reader to refer to Section A.1 for a high-level overview of SVM binary classification. In particular, we used the implementation LibSVM by Chang and Lin (2011), a library for training and testing SVM models.

In our application, SVM objects are candidate entities in $E_1 \cup E_2 \cup E_3$. Each candidate entity is associated a feature vector, as detailed later. Training examples were gathered by running the candidate entity generation step for all queries included in the training portion of GERDAQ, and labeling candidate entities included in the ground truth as positive examples, the others as negative.

The model was trained with SVM parameter $C$, RBF parameter $\gamma$, category weights $p_-, p_+$, chosen by means of a simple grid search. Also feature selection has been performed by ablation, as described in Section A.1.5. Both parameter optimization and feature selection has been done keeping as a model selection objective function the maximization of macro-$F_1$ of SMAPH-1 on the development
portion of GERDAQ. We chose a portion of the dataset disjoint from the training set to avoid over-fitting.

During the development of SMAPH-1, we also built models using kernel functions other than RBF, including linear SVM classifiers, but they achieved worse results. This shows that the feature space of entities is not linearly separable between correct and wrong entities, that features are inter-dependent, and that the projection in the infinite-dimensions space implicitly done by the RBF function is more easily linearly separable.

The entities of $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ surviving the classification process are returned as the result for $q$.

### 8.2.2 Entity features

The classifier is built on top of a set of features that, for each entity, take into account the coherence and robustness of the annotation process, the ranking and composition of snippets, the syntactic similarity between $q$ and the snippets’ bold text portions, the syntactic similarity between $q$ and the title of the candidate entity. Before exploring the features presented in Table 8.1, we need a few definitions:

- $U(q)$ is the list of URLs returned by Bing for query $q$. First URL is that of the highest-rank result;
- $C(q)$ is the list of snippets returned by Bing for $q$ (a sort of context of query terms). First snippet is that of the highest-rank result;
- $B(q)$ is the multi-set of bold portions of all snippets returned by Bing for $q$;
- $W(q)$ is the total number of web pages found by Bing for query $q$;
- $T(e)$ is the Wikipedia-page title of entity $e$;
- $T^*(e)$ is $T(e)$ excluding the final parenthetical-string, if any. E.g.

$$T^*(\text{ER (TV series)}) = \text{ER}$$

- $A(s)$ is the set of annotations (mention-entity pairs) found by our auxiliary annotator WAT in snippet $s$ overlapping with a bold portion of the snippet;
- $\rho(s, m, e)$ is the $\rho$-score indicating the confidence of the annotation $(m, e)$ in snippet $s$ (Piccinno, 2016);
- $lp(m)$ is the link probability of mention $m$ (Milne and Witten, 2008b), computed as the ratio between the number of times $m$ is an anchor in Wikipedia divided by the number of all its occurrences in the Wikipedia pages;
8.2. SMAPH-1: Individual entity pruning

- \text{comm}(m,e)\) is the \textit{commonness} of the annotation \((m,e)\) [Milne and Witten, 2008b], computed as the number of links in Wikipedia having \(m\) as anchor and pointing to the page of \(e\), divided by the number of times anchor \(m\) appears in Wikipedia as a link to any page.

- \text{amb}(m)\) stands for \textit{ambiguity} and is the number of distinct Wikipedia pages that a mention \(m\) points to in the whole Wikipedia;

- \(ED(x,y)\) is the Levenshtein distance between strings \(x\) and \(y\), normalized by \(\max(|x|_c,|y|_c)\), where \(|x|_c\) stands for the number of characters of string \(x\).

- \(MinED(a,b)\) is an asymmetric measure of distance of string \(a\) towards string \(b\), defined as follows\[2\] let \(a_t\) and \(b_t\) be the set of terms in strings \(a\) and \(b\), then

\[
MinED(a,b) = \text{avg}_{t_a \in a_t} \left( \min_{t_b \in b_t} (ED(t_a,t_b)) \right)
\]

In other words, for each term in \(a\), we find the closest term in \(b\); \(MinED(a,b)\) is the average distance between them.

- \(q^w\) is the query formulated by juxtaposing \(q\) and the term \textit{wikipedia}.

The feature vector for each entity is composed by features in Table 8.1 (hence, the dimension of the original feature space is \(p = 24\)). Since features are the very core of our classifier, we describe the rationale behind them. First of all it is very important to note that by using non-linear kernels, SVM captures, to some degree, inter-dependence of features. This means that, despite a feature not being by itself discriminative of an entity being pertinent or not to a query, they still provide an important signal when combined with other features.

Features 1 and 2 are relative to entities drawn from any source. Feature \textit{webTotal} is the number of documents in the web that matched query \(q\). A high value may indicate that it is more likely for a query to be well formed, thus containing entities. Named entities typically have a lower number of synonyms compared to general entities, hence if they are found, they are usually correct. Moreover, named entities are typically mentioned with first capital letter. To capture this, we included features \textit{isNE} and \textit{captBolds}.

We also defined features 3–8, relative to sources 1 and 2 (Wikipedia pages as web results of \(q\) and \(q^w\)). Feature \textit{rank} is rather obvious: if an entity has appeared in higher positions among web result, it is more likely to be pertinent to the query. The minimum edit distance between the title of the entity and the query, and between the bold portions of the snippets (mentions of the entity as they appear

\[2\]We use \(\text{avg}_{x \in X} f(x)\) to indicate the arithmetic average of the results of function \(f\) applied to all elements of \(X\), and \(\text{avg}(X)\) to indicate the arithmetic average of the elements of \(X\).
in web pages) and the query is a strong indicator of the entity being cited by the query. This motivates edit distance based features 4, 5 and 6. The average number of terms in bold snippets may also be an indicator of the strength of an entity.

Entities drawn from Source 3, our main source, have a number of features (9–24) relative to the process of snippet annotation performed by WAT. Feature $freq$ (how many snippets mention the entity) is an obvious indicator of an entity’s correctness. Similarly, feature $avgRank$ captures where, in the list of results, the entity is mentioned: if it is mentioned in higher-ranked snippets, it is more likely to be correct. The annotation process comes with important signals: $\rho$ is the confidence assigned by WAT to that annotation; $lp$ is the prior probability that the mention that refers to any entity; $comm$ is the prior probability that the mention refers to that particular entities, among the candidates, and not taking context into account. For all these measures, we take the minimum, maximum, and average value among the snippets. Higher values indicate strength of the entity. In contrast, $ambig$ is the number of senses a mention may have, and lower values suggest higher confidence in the annotation. Finally, similarly to edit-distance measures previously described, we capture a signal that suggests how much the mentions that were linked to the entity are contained in the query with features 23 and 24.

8.3 SMAPH-S: Local entity link-back

As we will see in the experimental section, most errors made by SMAPH-1 are false negative entities that appear as candidates, are explicitly mentioned in the query, but are assigned a low score due to their bad features and, thus, discarded by the SVM binary classifier. For example, an entity actually mentioned by the query may appear with low frequency in snippets (Feature 9). This reflects into a result with high precision but low recall: intuitively, for SMAPH-1’s binary classifier it is easier to detect bad entities than good ones, hence, to optimize $F_1$, it sacrifices recall in favor of a higher precision. To overcome this limitation, we decided to enforce the bond between candidate entities and the query terms that cite them by explicitly modeling it, in order to come up with features that boost the score of entities cited by the query.

With this motivation, we decided to move our focus to the A2KB problem, which forces us to model features not only relative to candidate entities, but also relative to the mentions (terms of the query) that link those entities. Our first step has been to design a variant of SMAPH-1, called SMAPH-S, that enforces this by a process we call link-back. The goal of this step is to match the candidate entities of the set $E_1 \cup E_2 \cup E_3$ with the most appropriate mentions present in the input query. This way, we can exploit features that better model how likely an
### 8.3. SMAPH-S: Local entity link-back

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>webTotal</td>
<td>$W(q)$</td>
</tr>
<tr>
<td>2</td>
<td>isNE</td>
<td>1 if $e$ is a named entity, 0 otherwise. Based on Freebase as detailed in (Carmel et al., 2014)</td>
</tr>
</tbody>
</table>

### Drawn From Sources $\mathcal{E}_1$ and $\mathcal{E}_2$

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>rank</td>
<td>position of $e$’s URL in $U(q^*)$</td>
</tr>
<tr>
<td>4</td>
<td>EDTtitle</td>
<td>$\text{MinED}(T(e), q^*)$</td>
</tr>
<tr>
<td>5</td>
<td>EDTiNP</td>
<td>$\text{MinED}(T^<em>(e), q^</em>)$</td>
</tr>
<tr>
<td>6</td>
<td>minEDBolds</td>
<td>$\min{\text{MinED}(b, q^<em>) : b \in B(q^</em>)}$</td>
</tr>
<tr>
<td>7</td>
<td>captBolds</td>
<td>number of capitalized strings in $B(q^*)$</td>
</tr>
<tr>
<td>8</td>
<td>boldTerms</td>
<td>$(1/</td>
</tr>
</tbody>
</table>

### Drawn From Source $\mathcal{E}_3$

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>freq</td>
<td>$(</td>
</tr>
<tr>
<td>10</td>
<td>avgRank</td>
<td>$(\sum_{i \in [0,25]} p_i)/25$ where $p_i = \begin{cases} i &amp; \text{if} (., e) \in A(C(q)_i) \ 25 &amp; \text{otherwise} \end{cases}$</td>
</tr>
<tr>
<td>11</td>
<td>pageRank</td>
<td>PageRank of $e$ in Wikipedia Graph $P := {\rho(s, m, e) : (m, s) \in X(q)}$; $\rho_{\text{min}}(P)$, $\rho_{\text{max}}(P)$, $\rho_{\text{avg}}(P)$</td>
</tr>
<tr>
<td>12</td>
<td>Lpmin</td>
<td>min($\mathcal{L}$)</td>
</tr>
<tr>
<td>13</td>
<td>Lpmax</td>
<td>max($\mathcal{L}$)</td>
</tr>
<tr>
<td>14</td>
<td>commmin</td>
<td>$\text{min}(\text{comm}(m, e) : (m, s) \in X(q))$</td>
</tr>
<tr>
<td>15</td>
<td>commmax</td>
<td>$\text{max}(\text{comm}(m, e) : (m, s) \in X(q))$</td>
</tr>
<tr>
<td>16</td>
<td>commavg</td>
<td>$\text{avg}(\text{comm}(m, e) : (m, s) \in X(q))$</td>
</tr>
<tr>
<td>17</td>
<td>ambigmin</td>
<td>$\text{min}(\text{amb}(m) : (m, s) \in X(q))$</td>
</tr>
<tr>
<td>18</td>
<td>ambigmax</td>
<td>$\text{max}(\text{amb}(m) : (m, s) \in X(q))$</td>
</tr>
<tr>
<td>19</td>
<td>ambigavg</td>
<td>$\text{avg}(\text{amb}(m) : (m, s) \in X(q))$</td>
</tr>
<tr>
<td>20</td>
<td>mentMEDmin</td>
<td>$\text{min}(\text{MinED}(m, q) : (m, s) \in X(q))$</td>
</tr>
<tr>
<td>21</td>
<td>mentMEDmax</td>
<td>$\text{max}(\text{MinED}(m, q) : (m, s) \in X(q))$</td>
</tr>
</tbody>
</table>

Table 8.1: Features of a candidate entity $e$ (used by SMAPH-1, SMAPH-S and SMAPH-2) for query $q$. Let $X(q) := \{(m, s) : s \in C(q) \land (m, e) \in A(s)\}$. 
entity is being cited by a sequence of query keywords. In this link-back process, some entities will be discarded because they cannot be linked to any mention in \( q \), while others, despite having bad feature values, will be boosted by a bond with a mention. The final result is a set of full annotations (i.e., mention-entity pairs) of the input query.

### 8.3.1 SMAPH-S algorithm

Let’s see in more detail how entity link-back is performed in SMAPH-S to build the solution for query \( q \). Let us denote with \( \text{Seg}(q) \) the set of all possible segments in \( q \) (a segment is an n-gram of any length) and with \( \mathcal{E}_q = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3 \) the set of candidate entities for \( q \). The set of candidate annotations for \( q \) is simply the Cartesian product \( \text{Seg}(q) \times \mathcal{E}_q \).

The SMAPH-S algorithm consists of three steps:

**Candidate annotation generation** The set of candidate annotations \( \text{Seg}(q) \times \mathcal{E}_q \) is generated. This set contains a high number of annotations, and only a few of them are correct. There is also a high number of annotations conflicting with other annotations, i.e., having the overlapping (possibly equal) mentions. Two conflicting annotations cannot both end up in the solution. This will be implemented by the last step.

**Candidate annotation ranking** Annotations in \( \text{Seg}(q) \times \mathcal{E}_q \) are ranked by the likelihood that they are correct, according to the value assigned to them by a regressor \( R : \text{Seg}(q) \times \mathcal{E}_q \mapsto \mathbb{R} \). The higher the value assigned to annotation \( a_i \), the more likely it is considered correct by the later step.

**Greedy solution generator** The sorted list of candidate annotations is scanned from most to least likely. Annotation \( a_i = (m_i, e_i) \) is added to the solution set only if \( m_i \) does not overlap with any previously added annotation, stopping whenever \( R(a_i) \) is lower than a threshold. The threshold is chosen in order to maximize the macro-\( F_1 \) on GERDAQ dev. This way, in case of conflicting annotations, the one with highest rank is chosen.

The choice of building such a regressor, as opposed to building a classifier like we did for SMAPH-1, comes from the need by the last step of the algorithm to choose between two conflicting annotations. \( R \) is a key component of the algorithm, and its accuracy in assigning likelihood scores is central, as it defines a policy to resolve conflicts among annotations and a criteria for stopping the annotation process, discarding low-ranked annotations. Let’s see how it is trained.
8.3.2 Prediction of likelihood for candidate annotations via SVR

Given an annotation \( a = (m, e) \), function \( R \) is learned to predict the likelihood (a real value in \([0, 1]\)) that mention \( m \) occurring in query \( q \) refers to entity \( e \). The training process defines a function \( R : Seg(q) \times E_q \mapsto \mathbb{R} \). The algorithm we use for regression (i.e., the process of finding \( R \)) is \( \epsilon \)-SVR, that defines \( R \) by means of a subset of support vectors, chosen as a subset of the training examples. We encourage the reader to refer to Section A.1.4 for a high-level description of Support Vector Regression and \( \epsilon \)-SVR.

Like for SMAPH-1, the regressor \( R \) is trained on training portion of GERDAQ. Training examples are candidate annotations \( Seg(q) \times E_q \) for all queries \( q \) in the dataset, generated as shown above. A candidate annotation is considered a positive example iff it appears in the ground truth; negative otherwise. Positive examples are assigned the label \( 1.0 \), negative examples are assigned \(-1.0\).

As already observed, training examples are heavily unbalanced towards negative ones. This is not a problem as SVR supports unbalanced training sets. We generate a feature vector for each annotation \( a = (m, e) \) as detailed later.

Similarly to SMAPH-1, training parameters and features of regressor \( R \) have been chosen with the objective function of maximizing macro-\( F_1 \) on the development set.

8.3.3 Annotation features

The set of features that constitute feature vectors associated to annotations \((m_i, e_i) \in Seg(q) \times E_q\)

are (i) those used by SMAPH-1, modeling properties relative to the entity \( e_i \) only (Features 1–24 in Table 8.1), concatenated with (ii) a set of five new features that capture aspects of the binding between mention \( m_i \) and entity \( e_i \) (Features 25–29 in Table 8.2). We point out that the features listed in both tables are the result of a feature selection process from a larger set of features involving annotations, bold parts of snippets and entities. The description of features in Table 8.2 uses, in addition to the definitions introduced in Section 8.2, the following definitions:

- \( F(e, s) \) is the number of times (frequency) that entity \( e \) has been linked in Wikipedia pages by anchor \( s \).
- \( G(e) \) is the set of anchors used in Wikipedia to link \( e \).

Let’s discuss the rationale between these features. Feature 25 (anchorsAvgED) is key: it is the average edit distance between mention \( m_i \) and all anchors that
### Table 8.2: Features of a candidate annotation \((m_i, e_i)\) (used by SMAPH-S and SMAPH-2), where \(m_i\) is the mention (list of query terms) and \(e_i\) is the entity.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>anchorsAvgED</td>
<td>(\frac{\sum_{s \in G(e_i)} \left( \sqrt{F(e_i, s)} \cdot ED(s, m_i) \right)}{\sum_{s \in G(e_i)} \sqrt{F(e_i, s)}})</td>
</tr>
<tr>
<td>26</td>
<td>minEdTitle</td>
<td>(\text{MinED}(m_1, T(e_1)))</td>
</tr>
<tr>
<td>27</td>
<td>EdTitle</td>
<td>(\text{ED}(m_1, T(e_1)))</td>
</tr>
<tr>
<td>28</td>
<td>commonness</td>
<td>(\text{comm}(m_1, e_1))</td>
</tr>
<tr>
<td>29</td>
<td>lp</td>
<td>(\text{lp}(m_1))</td>
</tr>
</tbody>
</table>

point to \(e_i\) in Wikipedia, which are possible ways to reference \(e_i\). Edit distances are weighted with respect to the number of times \(e_i\) is pointed to by an anchor (the square root mitigates the effect of high-frequency anchors). The more times \(e_i\) has been referenced by anchors similar to \(m_i\), the higher \(\text{anchorsAvgED}\) will be, hence a high value of this feature suggests that mention \(m_i\) refers to \(e_i\). Features 26 and 27 aim at measuring the syntactic similarity between mention and entity title: \(\text{edTitle}\) is simply their edit distance, while \(\text{minEdTitle}\) is the minimum word-to-word edit distance as defined above. Lower values of features 26 and 27 indicate a higher similarity between mention \(m_i\) and title of entity \(e_i\), and thus a stronger bond between the two. Feature 28 (\(\text{commonness}\)) is another measure or the mention-entity bond strength: the more frequently \(m_i\) refers to \(e_i\) (as opposed to other entities) in Wikipedia, the higher is the bond. Feature 29 (\(\text{lp}\)) is instead a measure of how likely the mention \(m\) is actually a mention of anything. A higher value of \(\text{lp}\) indicates that the whole annotation is more likely to be correct.

The experimental section will evaluate the performance of SMAPH-S on GERDAQ and will show that relying on individual mention-entity pairs one by one to select the binding for the query \(q\) is too restrictive and results in low \(F_1\). This motivated the design of a collective disambiguation process that is the novel core of SMAPH-2, the best performing annotator we propose in this thesis.

#### 8.3.4 Notable excluded features

The set of features described in this chapter and eventually employed by the SMAPH systems are a subset defined via a feature selection process by ablation of a wider number of features from which we discarded those that proved not to bring any advantage in terms of macro-\(F_1\) on the development set.

Among the excluded features we mention the score provided by the models that constitute the basis of (Blanco et al., 2015), based on word embeddings (generated with \texttt{word2vec}). These models measure the similarity of the word
embeddings of the query terms to the word embeddings of the first paragraph of
the Wikipedia page that describes e. In a way, they model the context of a query
mention as the word embeddings of surrounding keywords and exploit that con-
text for disambiguation. Though this score works well if used alone (see Blanco
et al. (2015)), it does not offer to our model any additional information to decide
whether an entity is mentioned by a query or not. A reason for that might be
that we already exploit a larger (and, experimentally, more precise) query con-
text given by the snippets in which the term appears. This context is then used
by the text-annotator WAT to assist entity disambiguation.

8.4 SMAPH-2: Joint entity link-back

Let’s recap. SMAPH-1 judges candidate entities individually, and aims at model-
ing the bond between an entity and the whole query; SMAPH-S judges candidate
annotations individually, and enriches SMAPH-1’s classifier by also modeling the
bond between a mention (keywords of the query) and the entity. In both systems,
the decision about whether or not to assign an entity (or annotation) to a query is
taken independently from other entities/annotations. For this reason, both sys-
tems fail in modeling the interdependency of entities and the quality of a solution
as a whole.

SMAPH-2 meets this goal focusing on bindings:

Definition 8.1. A binding is the set of all annotations of a document, i.e. the set of
all mentions into a document and the entities they refer to. It can be codified as a set of
annotations, i.e. a set \{⟨m₁, e₁⟩, · · · , ⟨mₙ, eₙ⟩\}.

By focusing on bindings, we define a joint annotation prediction model that
aims at ranking candidate bindings by their \(F₁\) score, taking into account features
about the complete set of annotations for a single query. In facts, considering
the whole set of annotations, in one binding, we can design features that cap-
ture properties of the relation between multiple annotations and the input query
(e.g., how many query terms are covered by the binding) and among annotations
themselves (e.g., the semantic relatedness among their entities). Those features
are described in Table 8.3. In SMAPH-2 we use also the features regarding single
entities and annotations contained in a candidate binding (Tables 8.1 and 8.2).

The core component of SMAPH-2 is the ranker that searches, among a set of
candidate bindings, the binding that maximizes \(F₁\), the primary evaluation mea-
sure of our task. The better a binding is (in terms of \(F₁\)) the higher its predicted
rank should be. SMAPH-2 shares the Candidate entity generation and the Candi-
date annotation generation steps with SMAPH-S (see Figure 8.1). After candidate
Candidate binding generation  We first enumerate all possible segmentations of \( q \) using a BIO encoding, namely sequences of the symbols B-I-O of length \(|q|\) (number of keywords in the query), which respectively denote the beginning, continuation and absence of a segment. These are \( o(3^{|q|}) \), though not all possible BIO sequences correspond to valid segmentations (a label I can only follow I or B, but not O). Note that the query length \(|q|\) is typically short, so the number of segmentation is treatable. The set of segmentations generated by the BIO sequences of \( q \) is called \( BIO_q \). For an example of segments for query \( q = \text{armstrong mon lading} \) (\(|q| = 3\)), see Table 8.3.

Candidate bindings are then generated for query \( q \) by taking each segmentation \( G \in BIO_q \) and assigning to each segment in it any possible entity in \( E_q = E_1 \cup E_2 \cup E_3 \) (with repetitions). This generates a total of \( O(|BIO_q| \cdot |q| \cdot |E_q|) \) training examples for each query \( q \), which are heavily unbalanced towards negative ones. We adopted a simple pruning heuristic to significantly confine the size of this set, so that the model can be trained in around 30 minutes. The heuristic is based on the following algorithm. It assigns, for each segmentation \( G \in BIO_q \), candidate entities drawn from \( E_q \) to segments \( s_i \in G \) (in the algorithm, this set is \( C_{s_i} \)):

1. For each pair \((s, e) \in G \times E_q\) if \( MinED(s, \mathcal{T}(e)) \leq 0.7\), add \((s, e)\) to \( C_s\);
2. For each entity \( e \in E_q \) that has not been added in the previous step, add \((s, e)\) to \( C_s\) for all \( s \in G\).
3. The set of candidate bindings generated for segmentation \( G \) is the Cartesian product \( C_{s_1} \times \cdots \times C_{s_{|G|}}\).

The final set of candidate bindings \( A_q \) for query \( q \) is the union of those Cartesian products for all \( G \in BIO_q \), formally:

\[
A_q = \bigcup_{G \in BIO_q} C_{s_1} \times \cdots \times C_{s_{|G|}} \text{ where } G = s_1, \ldots, s_{|G|}
\]

In other words, the heuristic consists in first assigning as candidates to segment \( s \) entities whose title is similar to \( s \), then assigning remaining, unassigned entities to all segments. An example of \( A_q \) is given in Figure 8.4.

This simple heuristic reduces the number of generated examples to an average of 495 bindings per query on GERDAQ train, and it keeps, for most queries, the candidates with highest \( F_1 \).
8.4. SMAPH-2: Joint entity link-back

<table>
<thead>
<tr>
<th>Segmentation</th>
<th>Size</th>
<th>BIO seq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>armstrong</td>
<td>mon</td>
<td>lading</td>
</tr>
<tr>
<td>armstrong mon</td>
<td>lading</td>
<td>2</td>
</tr>
<tr>
<td>armstrong</td>
<td>mon lading</td>
<td>2</td>
</tr>
<tr>
<td>armstrong mon lading</td>
<td>1</td>
<td>BII</td>
</tr>
<tr>
<td>armstrong</td>
<td>mon</td>
<td>2</td>
</tr>
<tr>
<td>armstrong mon</td>
<td>1</td>
<td>BIO</td>
</tr>
<tr>
<td>armstrong</td>
<td>lading</td>
<td>2</td>
</tr>
<tr>
<td>mon</td>
<td>lading</td>
<td>2</td>
</tr>
<tr>
<td>mon lading</td>
<td>1</td>
<td>OBI</td>
</tr>
<tr>
<td>armstrong</td>
<td>1</td>
<td>BOO</td>
</tr>
<tr>
<td>mon</td>
<td>1</td>
<td>OBO</td>
</tr>
<tr>
<td>lading</td>
<td>1</td>
<td>OOB</td>
</tr>
<tr>
<td>(empty set)</td>
<td>0</td>
<td>OOO</td>
</tr>
</tbody>
</table>

Figure 8.3: Example of segmentations set \(BIO_q\) for query \(armstrong mon lading\) of length 3. Vertical bar “|” indicates segment truncation. Middle column shows the number of segments for a segmentation. Right column shows corresponding B-I-O sequence for each segment.

Candidate binding ranking and best binding prediction We rank the bindings in \(A_q\) by means of a ranker \(R'\). This ranker is trained to rank bindings by their \(F_1\). SMAPH-2 returns as solution the binding \(A_q^*\) with the highest rank. The rest of this section will give details on how the ranker is trained and how it works.

8.4.1 Ranking of candidate bindings via LambdaMART

Ranker \(R'\), that is responsible for ranking candidate bindings for a query by their \(F_1\), is implemented as an ensemble of decision trees built with LambdaMART. We recommend the reader to refer to Section A.2 for a high-level overview of LambdaMART. We used the implementation RankLib\(^3\) by Van Dang, a library for training and testing ranking models, that provides nine learning-to-rank algorithms, including LambdaMART.

Given a set of candidate bindings \(A_q\) for query \(q\), ranker \(R'\) is trained to predict the correct ranking of those bindings according to the \(F_1\) measure they would achieve if they were given as a solution for \(q\). The training process defines a model for building a bijective mapping \(R' : A_q \mapsto [1, \cdots, |A_q|]\). In other words, prediction consists in assigning to each binding a unique position in the final ranking.

\(^3\)https://sourceforge.net/p/lemur/wiki/RankLib/
**Segmentation: armstrong|mon|lading**

\[ b_1 \in \mathcal{A}_q \quad \text{armstrong} \mapsto \underline{Neil \ Armstrong} \quad \text{mon} \mapsto \underline{Moon} \quad \text{lading} \mapsto \underline{Emergency \ Landing} \]

\[
\vdots
\]

**Segmentation: armstrong|mon lading**

\[ b_2 \in \mathcal{A}_q \quad \text{armstrong} \mapsto \underline{Neil \ Armstrong} \quad \text{mon lading} \mapsto \underline{Moon \ Landing} \]

\[ b_3 \in \mathcal{A}_q \quad \text{armstrong} \mapsto \underline{Lance \ Armstrong} \quad \text{mon lading} \mapsto \underline{Moon \ Landing} \]

\[ b_4 \in \mathcal{A}_q \quad \text{armstrong} \mapsto \underline{Louis \ Armstrong} \quad \text{mon lading} \mapsto \underline{Emergency \ Landing} \]

\[
\vdots
\]

**Segmentation: armstrong**

\[ b_5 \in \mathcal{A}_q \quad \text{armstrong} \mapsto \underline{Neil \ Armstrong} \]

\[ b_6 \in \mathcal{A}_q \quad \text{armstrong} \mapsto \underline{Lance \ Armstrong} \]

\[ b_7 \in \mathcal{A}_q \quad \text{armstrong} \mapsto \underline{Louis \ Armstrong} \]

\[
\vdots
\]

**Segmentation: (empty segmentation)**

\[ b_8 \in \mathcal{A}_q \quad \text{(empty binding)} \]

\[
\vdots
\]

Figure 8.4: An example of set \( \mathcal{A}_q \) for query armstrong mon lading. Only 8 bindings and four segmentations are shown. In the example, the set of segmentations \( BIO_q \) is that of Figure 8.3, while the set of candidate entities \( \mathcal{E}_q \) includes entities Neil Armstrong, Lance Armstrong, Louis Armstrong, Moon and Emergency Landing. For each segmentation, a set of bindings are generated. All bindings generated for segmentation \( G \) have size \( |G| \). For the empty segmentation, the only generated binding is the empty binding. In cases in which the query contains no entity mentions, this is the right solution. At training time, each binding is associated to the \( F_1 \) score it would achieve if it was given as solution. In the example, binding \( b_2 \) is the correct solution achieving \( F_1 = 1.0 \).
Similarly to SMAPH-1 and SMAPH-S, also ranker $\mathcal{R}'$ is trained on the training portion of GERDAQ. Training examples are candidate bindings in $A_q$ for each query $q$ of the dataset (see above for a description of how the $A_q$ set is generated). We generate a feature vector for each binding $b_i \in A_q$ as detailed later, and we compute the $F_1$ measure obtained by $b_i$ with respect to the ground truth for query $q$. The $F_1$ is instantiated with the Strong annotation match (Definition 4.18), meaning that in order to be considered correct, an annotation in the binding must have equal mention and entity with an annotation in the ground truth. The label associated to a candidate binding is the order of its $F_1$ value. For example, if $A_q = \{b_1, b_2, b_3, b_4\}$, and $F_1$ values associated to those bindings are respectively $l'_1 = 1.0$, $l'_2 = l'_3 = 0.5$, $l'_4 = 0.0$, then the label respectively assigned to them will be $l_1 = 3$, $l_2 = 2$, $l_3 = 2$, $l_4 = 1$, where a higher label indicates that the binding should be assigned a higher rank (because it would lead to a higher $F_1$).

Note that, at training time, the same label can be assigned to more than one binding. This is important because at training time LambdaMART aims at optimizing NDCG, which is a metric that measures the distance between the predicted ranking and the optimal ranking (see Section A.2.1). By assigning the same label to bindings with same $F_1$ we are ignoring the order of bindings with same $F_1$ (in the example, swapping the order of $b_2$ and $b_3$ does not change the value of the objective function). In other words, training only considers the order of training examples with different $F_1$. Intuitively, we are telling LambdaMART that in case two bindings have the same $F_1$, we don’t care which is assigned the highest rank. Also note that for each query, there is either one binding that achieves $F_1 = 1.0$ or none. In case there is one, that will be the only binding that is assigned the highest rank at training time.

Training examples are heavily unbalanced towards negative ones. This is not a problem as LambdaMART supports unbalanced training sets.

The reader might wonder why we chose to build a ranker to predict the order of bindings (and take as solution the highest-ranked binding), rather than a regressor to estimate the $F_1$ measure of a binding (and take as solution the binding with highest estimated $F_1$). The reason why we made this choice is the computational intractability of the second approach. While in SMAPH-1, for each query, we had $|E_q| \approx 10^4$ training examples (each with 24 features) and in SMAPH-S we had $|E_q| \cdot |\text{Seg}(q)| \approx 10^2$ training examples (each with 29 features), in SMAPH-2 we have $|BIO_q| \cdot |q| \cdot |E_q| \approx 10^3$ training examples, each with 96 features. Given that the computational complexity of training an SVR is $O(S^2 \cdot F)$ (Platt 1998), where $S$ is the number of training examples and $F$ is the number of features, training a SVR for SMAPH-2 is unfeasible. Training a LambdaMART model instead has a complexity of $O(S \cdot F)$ (Bekkerman et al. 2011).
8.4.2 Binding Features

Each candidate binding
\[ b = (m_1, e_1), (m_2, e_2), \ldots, (m_{\mid b \mid}, e_{\mid b \mid}) \]
is associated with a feature vector \( F(b) \) that can be logically divided in three groups of features:

**Entity features.** Features of Table 8.1, relative to entities, are computed for entities \( e_1, \ldots, e_{\mid b \mid} \). For each of those entity features, we add to \( F(b) \) its maximum, minimum and average value. This forms the first group of \( 3 \cdot 24 = 72 \) features. For example, for Feature \( \text{freq} \) (Feature 9, the ratio of snippets of search results that mentioned an entity), we consider its value for entity \( e_h \) that has highest frequency, for the entity \( e_l \) that has lowest frequency, and the average frequency of entities \( e_1, \ldots, e_{\mid b \mid} \). The rationale behind taking the maximum, minimum and average value of features is that they capture different aspects of the quality of the binding. For example, higher values of feature \( \text{freq} \) for entity \( e \) are indicators that \( e \) is probably mentioned by the query. Hence, if the minimum of feature \( \text{freq} \) across entities is high, that is an indicator that all entities contained in the binding are good, hence the binding is good. On the other side, if the maximum of feature \( \text{freq} \) across entities is low, that is an indicator that all entities are bad, hence the binding is bad. Average \( \text{freq} \) mitigates the effect of single high- or low-frequency entities and considers all entities with equal weight.

**Annotation features.** Features of Table 8.2 are computed over \( b \)'s annotations and their maximum, minimum and average values for each annotation are added to \( F(b) \). These form the second group of \( 3 \cdot 5 = 15 \) features that model, for each annotation \( (m_i, e_i) \), the probability that mention \( m_i \) refers to entity \( e_i \). The rationale is similar to that we described for entity features.

**Binding features.** Finally, features of Table 8.3, relative to the candidate binding as a whole, are computed over \( b \). These constitute the third group of 9 features. With these features, we can finally achieve the goal of capturing attributes of a binding as a whole. Let us show the rationale between each of them, but first note that also LambdaMART, similarly to SVR, captures, at least to some level, interdependence of features, whose power must thus often be considered in combination with others. An important aspect of bindings is how much its entities are related to each other. In facts, a high relatedness might indicate a good binding. To model this, we compute for each pair of entities their relatedness and then take the minimum and maximum of them as features 30 and 31. The relatedness is computed as the
### 8.5 Evaluation of SMAPH

In this section we provide an extensive evaluation of the performance of SMAPH systems and a comparison against the state of the art.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>(rel_{\text{min}})</td>
<td>(\min(R))</td>
</tr>
<tr>
<td>31</td>
<td>(rel_{\text{max}})</td>
<td>(\max(R))</td>
</tr>
<tr>
<td>32</td>
<td>(nTokens)</td>
<td>(</td>
</tr>
<tr>
<td>33</td>
<td>(covg)</td>
<td>(\sum_{(m,e) \in b_q} (</td>
</tr>
<tr>
<td>34</td>
<td>(sumSegLp)</td>
<td>(\sum_{s \in \text{Seg}(q)} lp(s))</td>
</tr>
<tr>
<td>35</td>
<td>(avgSegLp)</td>
<td>(\sum_{s \in \text{Seg}(q)} lp(s)/</td>
</tr>
<tr>
<td>36</td>
<td>(nBolds)</td>
<td>(</td>
</tr>
<tr>
<td>37</td>
<td>(nDisBolds)</td>
<td>(</td>
</tr>
<tr>
<td>38</td>
<td>(minEdBlds)</td>
<td>(\text{avg}_{b \in B(q)} \text{MinED}(b,q))</td>
</tr>
</tbody>
</table>

Table 8.3: Features of a candidate binding \(b_q\) for query \(q\) (used by SMAPH-2). 

\(E\) is a function that maps a binding to the entities it contains. \(rel(e_1, e_2)\) is the relatedness among entities \(e_1\) and \(e_2\), measured by the Jaccard similarity of the sets of incoming links for \(e_1\) and \(e_2\). Recall also that \(\text{Seg}(q)\) is the set of all possible segments of a query \(q\) (see Section 8.3.1).

Jaccard similarity of the sets of incoming links for the two entities in a pair. Features 32 and 33 aim at modeling the quality of the binding’s annotations coverage. This is of particular importance for estimating the recall of a binding: if a very long query (Feature 32 – \(nTokens\)) has very few keywords covered by an annotation (Feature 33 – \(covg\)), then the annotation probably lacks recall. Features 34 and 35 serve the same purpose: if a query contains mentions with high link probability but the coverage (Feature 33) is small, then the binding probably lacks recall; if it is the other way around, then the binding probably lacks precision. Also if the number of portions of snippets in bold form, counted as total (Feature 36) or distinct (Feature 37) is high, or if the edit distance of the bold portions of snippets and the query is small, then we expect the query to contain many annotations.

The final length of the feature vector \(F(b)\) is three times the number of features in Tables 8.1 and 8.2 (because each original feature is represented as its minimum, maximum and average across entities/annotations) plus the number of features in Table 8.3 for a total of 96 features.
It is important to point out that the performance of SMAPH heavily depends on the quality of results delivered by the underlying search engines, which depend on many aspects of the structure of a search engine, including the user data they possess (query logs, click logs), external resources (the Web, knowledge bases), their variation over time, and, last but not least, the algorithm used for ranking web results. It is virtually impossible to isolate the contribute of each of these components to the quality of SMAPH results, and we are constrained into treating the whole search engine as a black box.

### 8.5.1 Evaluation metrics

As evaluation metrics we use those proposed in Section 4.3. Algorithms for entity linking on queries may either solve C2KB or A2KB. We evaluate them respectively with metrics defined in Sections 4.4.1 and 4.4.3 that are general enough to work properly on query documents. The only consideration that needs to be done regards match relation for annotations, thus for the A2KB problem only. In long documents it makes sense to relax the annotation match to consider as correct annotations that link the right entity but whose mention only overlaps with that in the ground truth. This is needed, for example, in cases in which an article or a preposition is included in a mention of the ground truth but not in the solution. In these cases, it is reasonable to ignore “low-information” words. Queries instead typically have no articles and all keywords have a dense meaning, hence relaxing the mention match would consider as correct many annotations that link unrelated words. For this reason, we will measure the performance for the A2KB problem by instantiating the measures of precision, recall and $F_1$ only with the *Strong annotation match* $M_a$ (Definition 4.18 at page 55). For C2KB, we instantiate the metrics with the *Strong tag match* $M_e$ (Definition 4.17).

We recall the basics of those metrics through an example of the computation of the metrics for a single query. Let $q = \text{armstrong mon lading}$, and let the corresponding ground truth binding $b^* = \{a_1^*, a_2^*\}$ for the A2KB problem be composed of two annotations:

$$a_1^* = \text{armstrong} \mapsto \text{Neil Armstrong}$$
$$a_2^* = \text{mon lading} \mapsto \text{Moon Landing}$$

thus, the first term is a mention of the astronaut, while the second and third form a single mention of the “landing on the Moon” event. Let $\bar{b} = \{\bar{a}_1, \bar{a}_2, \bar{a}_3\}$ be the approximate solution given by a system:

$$\bar{a}_1 = \text{armstrong} \mapsto \text{Neil Armstrong}$$
$$\bar{a}_2 = \text{mon} \mapsto \text{Moon}$$
$$\bar{a}_3 = \text{lading} \mapsto \text{Moon Landing}$$
Annotation $a_1$ is a true positive (TP); $\bar{a}_2$ and $\bar{a}_3$ are false positives (FPs); $a_2^*$ is a false negative (FN). Counting them, we have

$$|tp(b^*, \bar{b}, M_a)| = 1$$
$$|fp(b^*, b, M_a)| = 2$$
$$|fn(b^*, \bar{b}, M_a)| = 1$$

yielding document-wise metrics:

$$P(b^*, \bar{b}, M_a) = 1/3$$
$$R(b^*, b, M_a) = 1/2$$
$$F_1(b^*, \bar{b}, M_a) = 2/5$$

Performance on a dataset is obtained by computing dataset-wise metrics micro- and macro- $P$, $R$, and $F_1$ (Section 4.3.4).

It is important to consider that queries sometimes contain no entities (in particular, named entities). The way these cases are accounted for can play a significant role in the final evaluation metrics. The ability of a system to not annotate entities where there are none is crucial. This aspect is better captured by macro measures, while micro measures focus on the quality of retrieved entities/annotations. For this reason, we will report both micro and macro measures.

### 8.5.2 Experimental setting

**The experimented annotators**

In our experiments we tested the following annotators (for algorithmic details see Chapter 3):

**WAT** is the improved version of TagME introduced in (Piccinno and Ferragina, 2014) for the A2KB task (annotation detection). As relatedness function in the disambiguation process we used the Jaccard similarity among in-links, because it performed best on GERDAQ.

**AIDA** is the A2KB annotator introduced in (Hoffart et al., 2011b), we downloaded the code from the official web site. AIDA offers several disambiguation methods, we tested all of them and found that they offer almost the same performance on GERDAQ, so we only report the best number.

---

4Note that we have also employed WAT as snippet annotator in SMAPH’s candidate entity generation phase (Section 8.1). Here instead, we are testing how it performs directly on queries.

5http://www.mpi-inf.mpg.de/yago-naga/aida/
8.5. Evaluation of SMAPH

**NTNU-UiS** is a query annotator for the C2KB task (entity only detection), introduced in (Hasibi et al., 2014) that uses a multi-stage framework, first recognizing entity mentions, next scoring candidate entities using a learning-to-rank method, finally, using a greedy algorithm to find all valid interpretation sets for the query.

**NTUNLP** introduced for the C2KB task in (Chiu et al., 2014) searches the query trying to match freebase surface forms with the longest-match strategy. The disambiguation step is built on top of TagME and Wikipedia.

**Seznam** introduced for the C2KB task in (Eckhardt et al., 2014) uses Wikipedia and DBpedia to generate candidate annotations, then builds a graph of mentioned entities exploiting the link structure of Wikipedia. The disambiguation step is based on PageRank over this graph that assigns a score to each entity.

**SMAPH-1** (Section 8.2) deals with the C2KB task.

**SMAPH-S** (Section 8.3) is our first proposal for the A2KB problem, derived from SMAPH-1, evaluates each mention-entity pair individually.

**SMAPH-2** (Section 8.4) is our final annotator that deals with the A2KB problem by evaluating annotation sets collectively.

The first two annotators (AIDA and WAT) are the baselines for the C2KB and A2KB problems, while SMAPH-S serves as a first step towards our best proposal for A2KB, namely SMAPH-2. Other annotators employed here are the top-ranking annotators of the ERD Challenge.

**Evaluation datasets**

Our experiments have been conducted on two datasets:

**ERD** The dataset used in the ERD Challenge to test the annotators solving the C2KB problem (Cornolti et al., 2014). The entity knowledge base is a subset of Freebase, namely only its named entities. It consists of 500 queries fully annotated with NEs drawn from Freebase. Details of its creation process are given in Section 6.1.2. The ERD Challenge dataset is not available off-line. Systems can be tested by sending queries to the ERD Challenge platform to be annotated. The query’s ground truth remains unknown, so it does not let any error analysis be carried out. This makes the evaluation against this dataset a real third-party check of the robustness of annotators. For detailed information about the creation of the ERD dataset, see 6.1.2.

---

6For an overview of the distinction between kinds of entities, see Section 2.1
8.5. Evaluation of SMAPH

GERDAQ This is the novel dataset we have built via CrowdFlower. The entity knowledge base is the whole Wikipedia, hence it includes all kinds of entities (not only named entities). Details of its creation process are given in Chapter 7. GERDAQ has been designed to train and test the query annotators on both the C2KB and the A2KB problem, including all entities of Wikipedia. Throughout this chapter, we run out evaluation limited on the test portion of the dataset.

8.5.3 Coverage of entity sources

The first evaluation we perform is about the coverage reached by the Candidate entity generation phase. As explained in Section 8.1, this component, that is shared among SMAPH systems, is the only one that aims at discovering entities that may be related to the query. After candidate entities are generated, some of them can be discarded, but none can be added. For this reason, its performance defines an upper bound on the recall of our system.

We evaluated the coverage of the three entity sources. We remind the reader the main idea behind each of them:

- \( \mathcal{E}_1 \) are Wikipedia pages appearing as results when searching \( q \);
- \( \mathcal{E}_2 \) are Wikipedia pages appearing as results when searching \( q + \) wikipedia;
- \( \mathcal{E}_3 \) are entities found by annotating snippets given as result to query \( q \).

SMAPH systems employ the union of these entity sources \( \mathcal{E}_q = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3 \). We also remind the reader that each entity source comes with a set of features associated with an entity (see Table 8.1 at page 109) that assist later steps of the algorithms.

Table 8.4 reports the coverage and precision of each entity source. Coverage and precision is given with respect to the set of all entities \( (C_E, P_E) \) and the set of named entities only \( (C_{NE}, P_{NE}) \). They were respectively computed as the macro-recall and macro-precision of an hypothetical system that returns the whole set \( \mathcal{E}_q \) as solution. We can note a few facts:

- Source \( \mathcal{E}_3 \) is the largest single source of entities, though having small precision.
- Entities in \( \mathcal{E}_1 \) are included in \( \mathcal{E}_2 \) (formally, \( \mathcal{E}_1 \subseteq \mathcal{E}_2 \)). This is motivated by the fact that Wikipedia pages that appear as result when searching \( q \) also appears when searching \( q + \) wikipedia. This may wrongly suggest to the
8.5. Evaluation of SMAPH

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{E}_1$</th>
<th>$\mathcal{E}_2$</th>
<th>$\mathcal{E}_3$</th>
<th>$\mathcal{E}_3 \cup \mathcal{E}_1$</th>
<th>$\mathcal{E}_3 \cup \mathcal{E}_2$</th>
<th>$\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_E$</td>
<td>14.8</td>
<td>28.7</td>
<td>84.9</td>
<td>86.1</td>
<td>87.6</td>
<td>87.6</td>
</tr>
<tr>
<td>$P_E$</td>
<td>35.3</td>
<td>21.5</td>
<td>23.4</td>
<td>22.5</td>
<td>19.6</td>
<td>19.5</td>
</tr>
<tr>
<td>$C_{NE}$</td>
<td>26.6</td>
<td>41.9</td>
<td>92.7</td>
<td>93.5</td>
<td>94.4</td>
<td>94.4</td>
</tr>
<tr>
<td>$P_{NE}$</td>
<td>44.6</td>
<td>24.2</td>
<td>23.2</td>
<td>22.4</td>
<td>19.2</td>
<td>19.1</td>
</tr>
</tbody>
</table>

Table 8.4: Coverage ($C$) and precision ($P$) of the entity sources $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ on GERDAQ test portion. Top two rows report coverage and precision about all entities, bottom two rows are limited to Named Entities. $C$ stands for coverage (how many entities of the ground truth are found), $P$ stands for precision (how many entities found are in the ground truth). Read the leftmost column as “entity source $\mathcal{E}_1$ finds 14.8% of all ground truth entities and 26.6% of the ground truth named entities, with a precision of 35.5% among all entities and 44.6% among named entities”.

reader that $\mathcal{E}_1$ can be discarded, because it does not extend the coverage. On the contrary, we observe that $\mathcal{E}_1$ has higher precision than $\mathcal{E}_2$, hence the fact that $e \in \mathcal{E}_1 \land e \in \mathcal{E}_2$ gives a stronger signal about the correctness of $e$ compared to the case in which $e \notin \mathcal{E}_1 \land e \in \mathcal{E}_2$.

- Merging all sources together adds to plain $\mathcal{E}_3$ a +2.7% coverage on all entities and +0.9% coverage of named entities. But this is not the only advantage to using $\mathcal{E}_1$ and $\mathcal{E}_2$: again, the membership of a candidate entity $e$ in $\mathcal{E}_1$ and/or $\mathcal{E}_2$ gives a stronger signal of its correctness than just being in $\mathcal{E}_3$.

- The coverage reached by the union $\mathcal{E}_q$ is rather good for both the set of all entities (87.6%) and the set of named entities (94.4%). An ideal system built on top of these sources, i.e. a system that keeps all correct candidate entities and discards all wrong ones, would achieve an impressive $F_1$ score of around (93%) when considering all entities.

8.5.4 Experiment results

Experiment #1: Named Entity-only detection (C2KB-NE task)

In this section, we measure the ability of our competing systems in finding named entities mentioned by queries of the ERD and GERDAQ datasets. Some of the annotators we experiment with are designed to detect all entities and, in some cases, their mentions; however, in this experiment we evaluate their ability to spot named entities only, without considering the corresponding mentions. In other words, we test the systems’ ability to solve C2KB, using as Knowledge Base the subset of Wikipedia entities that are named entities.
Table 8.5: C2KB over ERD dataset (macro-$F_1$).

<table>
<thead>
<tr>
<th>System</th>
<th>$F_{1mac}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIDA</td>
<td>22.1</td>
</tr>
<tr>
<td>WAT</td>
<td>58.6</td>
</tr>
<tr>
<td>Seznam</td>
<td>66.9</td>
</tr>
<tr>
<td>SMAPH-S</td>
<td>67.0</td>
</tr>
<tr>
<td>NTU</td>
<td>68.0</td>
</tr>
<tr>
<td>SMAPH-1</td>
<td>68.8</td>
</tr>
<tr>
<td>NTNU-UiS</td>
<td>69.9</td>
</tr>
<tr>
<td>SMAPH-2</td>
<td>70.8</td>
</tr>
</tbody>
</table>

Table 8.5 reports the macro-$F_1$ computed by the ERD online evaluation infrastructure. The table shows that WAT is superior to AIDA over the annotation of queries, but it is up to 10% worse than the winner of the ERD’14 Challenge (SMAPH-1), which is in turn superseded by our new proposal SMAPH-2 by another +2% (absolute) in macro-$F_1$. We notice that after the completion of the ERD’14 Challenge, other systems have been proposed and tested on the ERD platform. Systems such as NTNU-UiS and NTU have scored better results than the original SMAPH-1. Nonetheless, SMAPH-2 obtains again the top spot.

As a first result, these figures show that queries are difficult to annotate by traditional means of entity linking. The $F_1$-measure over queries obtained by WAT and AIDA is significantly lower, $-16\%$ absolute, than their $F_1$ achievements on short texts (see Section 5.2.3). This motivates the design of dedicated annotators for web queries, that are crucial in order to reach reasonable performance. Another interesting result is that linking-back entities to mentions seems not useful if not properly implemented; e.g., by a joint full-query prediction approach. In fact, SMAPH-S performs worse even than SMAPH-1.

We also tested annotators on the test portion of the GERDAQ dataset, restricting the evaluation to named entities only (i.e., non-named entities, both in the solution provided by the system and in the ground truth provided by the dataset, were not considered in the evaluation). Table 8.6 confirms results of the previous experiment: SMAPH-2 improves SMAPH-1 by about 3% (macro-$F_1$) and 5% (micro-$F_1$).

Unlike the ERD dataset, the ground truth of the GERDAQ dataset is known at evaluation time, hence we can perform a more in-depth analysis, also measuring precision and recall, both in micro- and macro- weighting. Table 8.6 shows, among other things, that WAT has higher entity recall than AIDA. This is mo-
tivated by the fact that AIDA expects text in natural language as input, while queries are not.

In this experiment, a significant portion of the queries have no ground truth entities attached, and this explains why micro measures are so much smaller than macro measures, especially for systems like AIDA that return very few entities: we recall the reader that, while macro- measures are the average of each measure on the documents, micro- measures consider the whole dataset as a single document and computes the measures on it. This implies that macro- measures capture the performance on both (i) queries with non-empty ground truth (i.e. the ability of the system in assigning the right entity) and (ii) without empty ground truth (i.e. the ability of the system in not assigning any entity to query that do not mention any). Micro- measures, on the other hand, especially for conservative systems that return few entities, is more focused on the performance of non-empty queries.

**Experiment #2: Generic entity detection (C2KB task)**

This experiment is similar to Experiment #1 but without any restriction on the kind of detected entities, which may be now all entities represented in Wikipedia, including generic concepts. Formally, this is the C2KB problem, using the whole English Wikipedia as Knowledge Base. Since the ERD dataset only covers named entities, we can perform this experiment only on GERDAQ, that uses the whole Wikipedia as Knowledge Base.

In comparison to the previous experiment on named entities only, we notice that the ranking of the systems by result is similar, but in terms of measures, detecting generic entities is a harder problem: \(F_1\) decreases by about 7\% (micro-\(F_1\)) and 22\% (macro-\(F_1\)). The significant decrease might be attributable to the fact that NEs are easier to detect because they are less ambiguous. Moreover, the number of queries with empty ground truth is obviously lower than the previous experiment, and it is harder to detect the right entity among the knowledge base than to detect the absence of entity mentions in a query.

<table>
<thead>
<tr>
<th>Annotator</th>
<th>(P_{\text{mac}})</th>
<th>(R_{\text{mac}})</th>
<th>(F_{1\text{mac}})</th>
<th>(P_{\text{mic}})</th>
<th>(R_{\text{mic}})</th>
<th>(F_{1\text{mic}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIDA</td>
<td>94.8</td>
<td>59.6</td>
<td>58.4</td>
<td>31.6</td>
<td>4.8</td>
<td>8.3</td>
</tr>
<tr>
<td>TagME</td>
<td>75.4</td>
<td>83.3</td>
<td>63.2</td>
<td>52.1</td>
<td>49.6</td>
<td>50.8</td>
</tr>
<tr>
<td>WAT</td>
<td>70.3</td>
<td>85.3</td>
<td>64.4</td>
<td>42.8</td>
<td>66.9</td>
<td>52.2</td>
</tr>
<tr>
<td>SMAPH-1</td>
<td>85.5</td>
<td>82.7</td>
<td>74.5</td>
<td>62.0</td>
<td>62.5</td>
<td>62.2</td>
</tr>
<tr>
<td>SMAPH-S</td>
<td>82.6</td>
<td>82.3</td>
<td>73.1</td>
<td>58.7</td>
<td>59.7</td>
<td>59.2</td>
</tr>
<tr>
<td>SMAPH-2</td>
<td>85.8</td>
<td>84.5</td>
<td>\textbf{76.0}</td>
<td>65.5</td>
<td>62.9</td>
<td>\textbf{64.2}</td>
</tr>
</tbody>
</table>

Table 8.6: C2KB over GERDAQ dataset (test portion) and NEs only.
Table 8.7: C2KB results on GERDAQ test, all entities.

<table>
<thead>
<tr>
<th>Annotator</th>
<th>$P_{mac}$</th>
<th>$R_{mac}$</th>
<th>$F_{1mac}$</th>
<th>$P_{mic}$</th>
<th>$R_{mic}$</th>
<th>$F_{1mic}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIDA</td>
<td>94.0</td>
<td>12.2</td>
<td>12.6</td>
<td>28.6</td>
<td>1.5</td>
<td>2.8</td>
</tr>
<tr>
<td>TagME</td>
<td>60.4</td>
<td>51.2</td>
<td>44.7</td>
<td>52.7</td>
<td>49.6</td>
<td>51.1</td>
</tr>
<tr>
<td>WAT</td>
<td>49.6</td>
<td>57.0</td>
<td>46.0</td>
<td>43.0</td>
<td>56.4</td>
<td>48.8</td>
</tr>
<tr>
<td>SMAPH-1</td>
<td>77.4</td>
<td>54.3</td>
<td><strong>52.1</strong></td>
<td>58.5</td>
<td>54.0</td>
<td><strong>55.9</strong></td>
</tr>
<tr>
<td>SMAPH-S</td>
<td>64.8</td>
<td>56.2</td>
<td><strong>51.4</strong></td>
<td>57.0</td>
<td>54.7</td>
<td><strong>55.8</strong></td>
</tr>
<tr>
<td>SMAPH-2</td>
<td>72.1</td>
<td>55.3</td>
<td><strong>54.4</strong></td>
<td>64.1</td>
<td>51.3</td>
<td><strong>57.0</strong></td>
</tr>
</tbody>
</table>

Table 8.8: A2KB results on GERDAQ test. Metrics based on Strong Annotation Match.

<table>
<thead>
<tr>
<th>Annotator</th>
<th>$P_{mac}$</th>
<th>$R_{mac}$</th>
<th>$F_{1mac}$</th>
<th>$P_{mic}$</th>
<th>$R_{mic}$</th>
<th>$F_{1mic}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIDA</td>
<td>94.0</td>
<td>12.2</td>
<td>12.6</td>
<td>28.6</td>
<td>1.5</td>
<td>2.8</td>
</tr>
<tr>
<td>TagME</td>
<td>58.4</td>
<td>49.7</td>
<td>43.0</td>
<td>50.3</td>
<td>47.9</td>
<td><strong>49.1</strong></td>
</tr>
<tr>
<td>WAT</td>
<td>47.2</td>
<td>54.3</td>
<td><strong>43.6</strong></td>
<td>40.3</td>
<td>53.0</td>
<td>45.8</td>
</tr>
<tr>
<td>SMAPH-S</td>
<td>59.5</td>
<td>50.6</td>
<td><strong>46.3</strong></td>
<td>51.0</td>
<td>49.4</td>
<td><strong>50.2</strong></td>
</tr>
<tr>
<td>SMAPH-2</td>
<td>68.4</td>
<td>52.3</td>
<td><strong>51.4</strong></td>
<td>59.9</td>
<td>47.9</td>
<td><strong>53.2</strong></td>
</tr>
</tbody>
</table>

Again, off-the-shelf text annotators (AIDA and WAT) are worse than query annotators. SMAPH-S is again worse than SMAPH-1, but with a smaller gap, thus providing some credit for the usefulness of local link-back. SMAPH-2 is still the best entity annotator with a significant increase with respect to WAT of about 9% in macro/micro $F_1$ (absolute) and about 2% with respect to SMAPH-1.

**Experiment #3: Annotation detection (A2KB task)**

The goal of this experiment is to evaluate annotators over the most general scenario of the detection of all entities and their mentions. As in Experiment #2, we will use here only GERDAQ, the only dataset that provides annotations on queries. SMAPH-1, which solves the C2KB problem, cannot be tested in this experiment (see Section 4.2). As motivated in Section 8.5.1, we employ the Strong annotation match (namely, exact match on both entities and mentions) as match relation to instantiate the measures for A2KB on queries. Results for this experiment are reported in Table 8.8.

We first notice that performance on A2KB is lower than that in C2KB (Table 8.7). This is motivated by the fact that a solution for A2KB is intrinsically harder to find than a solution for C2KB: not only the entity has to be right, but also the mention. For example, the best macro-$F_1$ on C2KB (i.e. 54.4% of SMAPH-2) decreases on A2KB to 51.4%.

Again, off-the-shelf text annotators (AIDA and WAT) are worse than query
8.5. Evaluation of SMAPH

annotators, but now SMAPH-S improves over them by about 3-4% in macro/micro $F_1$. SMAPH-2 is still the best annotator with a gap over SMAPH-S slightly larger on A2KB than that observed in the C2KB task; about +5% macro-$F_1$ and +3% micro-$F_1$. This enforces our previous observation on the importance of designing specific query annotators. We notice that link-back, even in the local-disambiguation approach, improves text annotators by at least 3%. The increased gap between SMAPH-S and SMAPH-2 in the A2KB task, with respect to the C2KB one, seems to provide solid evidence of the value of the simple joint entity mention detection and linking we have introduced with SMAPH-2.

In conclusion, after having compared it with all query annotator publicly available, we can say that SMAPH-2 is the state of the art for what concerns entity linking on queries, that the idea of generating candidate entities by piggybacking on search engine is promising and that joint annotation prediction models the task of query annotation in a better way compared to judging each entity/annotation individually, and this lets us build solutions with higher precision and recall.

Experiment #4: Performance on exotic queries

This experiment is specific to SMAPH, and aims at finding out its robustness when treating “exotic” queries. By exotic, we indicate a query whose information need is more specific and harder to find on the web, as the search engine provides fewer information resources for it. For example, a query about an event widely covered by the media (2017 superbowl) is less exotic than a query about a small village in Kazakhstan (economy of karsakbay).

The experiment consists in analyzing the performance of SMAPH on queries with varying “exoticism”. We emulate exoticism with the number of total web results found by the search engine for that query, arguing that exotic queries have a lower number of results. We divide the queries of GERDAQ Test in four buckets of equal size (around 62 queries per bucket) according to the number of web results for those queries. The first result tells us that the number of web results follow an exponential distribution: the quantiles computed at probabilities $p = (0, 0.25, 0.5, 0.75, 1)$ are found respectively at 0, 189k, 913k, 4.9M, and 175M number of results. There are four queries with no results, and the query with the highest number of results (175M) is music player download (note the typo).

The second result regards the robustness of SMAPH with respect to the exoticism of queries. For each bucket, we report the average $F_1$ reached by SMAPH-2 for the C2W problem. Table 8.9 reports those results. The macro-$F_1$ has some degree of variation with respect to the macro-$F_1$ over the whole dataset (which is 54.4%, see Table 8.7), but does not seem to be affected by the exoticism of the query. Indeed, queries for which the search engine finds a lower number of re-
8.5. Evaluation of SMAPH

<table>
<thead>
<tr>
<th>Web results</th>
<th>$P_{mac}$</th>
<th>$R_{mac}$</th>
<th>$F_{1mac}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 189k</td>
<td>73.1</td>
<td>65.2</td>
<td>62.5</td>
</tr>
<tr>
<td>189k - 913k</td>
<td>57.3</td>
<td>51.1</td>
<td>49.3</td>
</tr>
<tr>
<td>913k - 4,916k</td>
<td>65.2</td>
<td>59.9</td>
<td>57.2</td>
</tr>
<tr>
<td>4,916k - 175,000k</td>
<td>59.0</td>
<td>49.3</td>
<td>48.4</td>
</tr>
</tbody>
</table>

Table 8.9: C2KB results on GERDAQ test, bucketized by number of web results.

results are those on which SMAPH reaches a higher $F_1$. This is partially due to the fact that such queries are likely to have an empty ground truth (the first bucket has 12 queries with empty ground truth out of 62, whereas the other buckets have around 5), and it is easier to find the correct solution for them compared to non-empty queries. But even discarding those queries, the trend does not change. We conclude that the performance of SMAPH is not affected by the number of results returned by the search engine and that it is robust on exotic queries.

8.5.5 Runtime evaluation

One context where entity linking on queries may be deployed is that of web searches, that usually take a few milliseconds to be performed. To make it feasible to have a step of entity linking as part of the web search process, entity linking must be performed in a comparable time span. To shed some light on the feasibility of this, we have measured the runtime of SMAPH. It is important to note that our main focus during the development of SMAPH was the quality of results rather than the runtime performance, and further work may lead to important gains in runtime, keeping a high quality, hence the figures presented in this experiment must be taken as preliminary.

The reader may refer to Figure 8.1 (Page 102) for an overview of the steps performed by the three variants of SMAPH. In this section, we will report the runtime for the three systems, measuring the length of the period of time between the call to SMAPH and the return of the result. Moreover, we do not consider the time for submitting the two queries to the search engine (the two top boxes in the figure), as it heavily depends on the connection, and only measure the additional time needed by the SMAPH systems to perform the annotation.

The experiments have been run on a consumer PC (Intel i7 CPU), deploying the process on a single core. The runtime has been measured several times showing minimal differences (below 1%) among runs.

The steps performed by the three variants of SMAPH are somehow incremental: SMAPH-1 (Section 8.2) generates candidate entities and judges each of them through an SVM prediction, hence the runtime is proportional to $|E_q|$; SMAPH-
Table 8.10: Time employed by SMAPH variants to annotate a query. Time is additional with respect to querying the search engine.

<table>
<thead>
<tr>
<th>System</th>
<th>Avg. time (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMAPH-1</td>
<td>70.1</td>
</tr>
<tr>
<td>SMAPH-S</td>
<td>182.3</td>
</tr>
<tr>
<td>SMAPH-2</td>
<td>1296.5</td>
</tr>
</tbody>
</table>

S (Section 8.3) generates candidate annotations by linking candidate entities to all segments of the query, and judges each annotation, hence the runtime is proportional to $|E_q| \cdot |Seg(q)|$; finally, SMAPH-2 (Section 8.4) generates all subsets of these annotations (excluding overlaps), leading to a runtime proportional to $|BIO_q| \cdot |q| \cdot |E_q|$.

Results show that SMAPH-1 has an average annotation time of 70 milliseconds. Considering mentions-entity pairs (SMAPH-S) increases the runtime by a factor of 2, while considering subsets of annotations further increases the runtime by a factor of 7.

In conclusion, SMAPH-1, that reaches an interesting performance (52.1% macro-F1 for C2W on all entities, see Table 8.7), is fast enough to be deployable without increasing search time by an important margin, while to reach a higher performance (54.4%) with similar runtime, further work must be put into optimizing SMAPH-2.
Conclusions and Future Work

Entity linking is a fundamental component of natural language understanding, and has been effectively used in many applications. The main purpose of identifying entities mentioned in the text is to build a bridge between unstructured, natural language text and knowledge bases providing rich, curated and structured information.

This thesis presents two main contributions to the field: we first defined the BAT-Framework, a framework to formalize the problems of entity linking and a set of measures to fairly compare them to each other, according to specific aspects we need to measure. This is an important contribution for the community, as it defines a common ground to develop and test new algorithms for entity linking. The BAT-Framework for benchmarking entity linking systems has been adopted by a large number of researchers, partially thanks to GERBIL, an implementation that provides an easy way to benchmark systems.

We later moved our focus on the problem of entity linking in queries. We describe the creation of GERDAQ, a well curated, highly accurate benchmarking dataset for the annotation of queries. In particular, we describe in detail the process we employed to build the ground truth for a set of queries via crowdsourcing. We also proposed SMAPH, a family of query annotators built on top of a search engine and a text annotator, and proved that it outperforms state of the art query annotators on both the GERDAQ and the ERD datasets.

A big effort has been put in place to build entity linking systems, and the annotation of many kinds of documents has reached reasonable performance. Current state-of-the-art annotators achieve a performance good enough for many kinds of applications. That said, a lot still has to be done to achieve a full comprehension of the meaning of a document. Our research in this field will follow these two vectors: on one side, improve entity linking, on the other side propose methods that use entity linking to improve information extraction, question answering, information retrieval, document clustering, information visualization, and all applications that could experiment a dramatic improvement thanks to an annotation
system working properly.

There are many directions to explore in order to increase the performance of annotators, keeping in mind the constraints about runtime when it comes to processing big data. Here we cite a few.

**Improving the similarity measure.** A recurrent core component of many text annotators, including best performing ones, is the similarity measure between entities. Some of them, for example TagME, is built on the assumption that semantically close entities are more likely to be mentioned in the same document. Currently, TagMe employs a similarity measure defined in (Milne and Witten, 2008a) that takes into account the amount of common incoming links in the Wikipedia graph between two entities. The assumption here is that if many Wikipedia articles $A_1, \cdots, A_n$ talk about both $E'$ and $E''$, then these two entities are probably semantically close.

Despite being extremely fast, this similarity measure does not take into account many factors. For example, the amount of outgoing links of $A_i$: if $A_i$ is a long page with many links, the evidence that $E'$ and $E''$ are semantically close should be weaker. Secondly, if $A$ is an important hub in the graph, its assessment of $E'$ being close to $E''$ should be considered stronger. Finally, the similarity measure does not take into account the similarity between $A_i$ and the two entities $E'$ and $E''$ for weighting the contribution of $A_i$.

There are many variants that could improve the similarity measure that deserve further investigation, both in terms of efficacy and efficiency. In particular, efficiency is fundamental for Web-scale applications. In particular, we would like to investigate the application of DeepWalk (Perozzi et al., 2014), a measure that considers embeddings of nodes by means of random walks in the graph. Applied to the Wikipedia graph, these technique should build, for each entity, an embedding representation of the entity that can be compared to that of other entities to build a relatedness function (Zwicklbauer et al., 2016b; Piccinno, 2016).

**Improving semantic segmentation.** The disambiguation algorithms of many annotators assume a “global coherence” among the whole document, meaning all mentioned entities should be semantically related to each other. Nonetheless, a document may talk about distinct sub-topics, and parts of the same document could be unrelated to each other. Enforcing document-wise coherence in this case would lead to errors. How can we segment a text in order to identify semantically coherent subsets of entities?

This could be formulated as a clustering problem and, once solved, might improve disambiguation, either for very long documents and for short, un-
structured ones like search-engine queries.

**Modeling probability à la “language model”**. Statistical language modeling serves the purpose of forecasting what word will follow after $n$ consequent words have been observed. The probability distribution of the $(n+1)$th word is machine-learned simply reading a corpus. For example, after the list of 6 words “Silvio Berlusconi is the president of”, a high probability would be assigned to words such as Milan and Italy, while a low probability would be assigned to UN.

We can apply a similar model to the entity disambiguation phase. Suppose we want to disambiguate sentence “Berlusconi is the president of Milan”, and that we have analyzed 10000 articles drawn from a newspaper and observed for 100 times sentences in the form “[person] is the president of [soccer_team]”, while sentences in the form “[person] is the president of [city]” never appeared.

Since entities are categorized, we know that AC Milan is a [soccer_team], while Milan is a [city]. A “language model” that takes into account entities would boost the disambiguation phase, suggesting that AC Milan is the correct entity to assign to the mention Milan.

**Relevance of the entity.** In applications such as text summarization and clustering, it’s fundamental to understand what entities are most relevant to describe the contents of the document. In a sentence like “On Sunday, Michelle Obama flied from New York to Haiti after the earthquake”, entities such as Michelle Obama and Haiti are clearly more relevant than Sunday and New York City.

A strategy to understand the relevance of entities could be based on the position in the sentence where the entity has been spotted. Subordinate sentences may tend to mention less relevant entities. That of finding the grammatical structure of sentences is a well-known problem (Klein and Manning 2003; Attardi et al. 2007), for which there are tools that offer good performance in terms of effectiveness and runtime.

Paranjpe (2009) addresses the issue using machine learning on query logs, to learn what terms are more important to characterize a document, while Gillick and Dunietz (2014) use machine learning with features derived from a standard NLP pipeline to associate a salience score to entities.

**Compressed data structures for fast access and succinct storage.** The Wikipedia graph features 5.2 million nodes (entities) and 70 million edges. It is ever-evolving, currently growing at the rate of 795 new entities per day. The graph is a small world graph with an average shortest path of 4.5. The Wiki-
data knowledge base offers 21.6 million entities. Though it is currently possible to store these graphs in memory without any compression, for some computations performed by entity linkers the space may grow too much. For example, TagMe stores in memory the matrix of relatedness between entities, and its uncompressed size goes with the square of the size of Wikipedia. In such cases, we need to use proper compression storage schemes that also support fast access to the compressed data. The accessory information needs to be compressed, but the runtime of the algorithm can’t be affected.

We aim at investigating the extension of the novel compression scheme proposed by Farruggia et al. (2014) for textual data, which ensures to control the trade-off between compressed space and decompression time, to the context of labeled graphs, by taking inspiration from the Web-Graph compression scheme proposed by Claude and Ladra (2011).
In this appendix we will present the statistical machine learning algorithms employed in this thesis.

A.1 Support Vector Machines (SVM) for binary classification and regression

In Chapter 8 we use a Support Vector Machine (SVM) binary classifier as a fundamental component of our system SMAPH. Here we briefly recall how SVMs work, in the hope of providing the reader with the main intuition behind it, sparing the mathematical details, and limiting the scope to the methods employed in this thesis.

A.1.1 Linear SVM classifiers

In SVM, like most machine learning algorithms, objects of a domain are represented as feature vectors. Each feature is an individual measurable property of an object. In SVM, feature values are real numbers, and a feature vector can be viewed as a point in the $d$-dimensional space $\mathbb{R}^d$, where $d$ is the number of features. A linear SVM model for binary classification consists of an hyperplane in $\mathbb{R}^d$, that is the locus of points $x$ verifying

$$w \cdot x + b = 0$$

where $\cdot$ is the dot-product, $w$ is a $d$-dimensional vector perpendicular to the hyperplane, and $b$ is a real number.

This hyperplane has the purpose of discriminating positive from negative objects, depending on which side of the hyperplane their feature vector falls in. The
A.1. Support Vector Machines (SVM) for binary classification and regression

A hyperplane is found as the solution of an optimization problem resulting in the choice of a subset \( X' \) of points in the training set. Points in \( X' \) are called support vectors and are the points in the training set that have smallest margin. The margin is the minimum distance between correctly classified objects and their projection in the hyperplane. During optimization, support vectors \( x'_i \in X' \) are associated a multiplier \( \alpha_i \). Prediction of objects with feature vector \( x \) depends on the verification of

\[
\sum_{x'_i \in X'} \alpha_i (x'_i \cdot x) + b \geq 0 \tag{A.1}
\]

if the inequality holds, the object associated to \( x \) is categorized as positive, otherwise it is negative. Values of \( \alpha_i \) determine how much the closeness of vector \( x \) to support vector \( x' \) indicates its membership to the positive (\( \alpha_i > 0 \)) or negative (\( \alpha_i < 0 \)) class.

A.1.2 Non-linear SVM classifiers

In some cases, an hyperplane defined on the input \( d \)-dimensional space is too rigid to separate categories. The alternative to defining the hyperplane in the original feature space are inner-product kernels (Boser et al., 1992), that can be used to construct better fitting and more general hyperplanes in a \( n \)-dimensional space \( \mathbb{R}^n \). Formally, by using non-linear kernels, original feature vectors in \( \mathbb{R}^d \) are mapped into \( \mathbb{R}^n \) through function \( \Phi : \mathbb{R}^d \rightarrow \mathbb{R}^n \). In \( \mathbb{R}^n \), points should be better separable by a linear hyperplane defined in \( \mathbb{R}^n \). The dot product in the prediction inequality becomes \( \Phi(x_i) \cdot \Phi(x) \).

Thanks to the so-called kernel trick, this projection does not need to be explicit, and dot-products are computed by means of a non-linear kernel function. The kernel function we will use in Chapter 8 is the Radial Basis Function (RBF), defined as:

\[
K(x, x') = e^{-\gamma ||x-x'||^2}
\]

where \( \gamma \) is a free parameter and \( x, x' \) are two \( d \)-dimensional feature vectors of the input space, hence \( ||x-x'||^2 \) is the squared Euclidean distance of the two vectors. The value of \( K(x, x') \) ranges between 0 (in the limit, for very distant vectors) and 1 if \( x = x' \).

Just like linear SVM, also building a non-linear SVM model consists in finding the set of support vectors \( X' \) (hence, a set of vectors in the original \( d \)-dimensional space). But now, points in \( X' \) are those that lie on the margin in the \( n \)-dimensional feature space. The classification inequality (Inequality A.1 for linear SVM) becomes:
A.1. Support Vector Machines (SVM) for binary classification and regression

\[ \sum_{x'_i \in X'} \alpha_i K(x, x'_i) + b \geq 0 \] (A.2)

if it is verified, the object associated to feature vector \( x \) is categorized as positive, otherwise it is negative.

Intuitively, RBF creates a “bump of influence” around each support vector \( x'_i \): the closer feature vector \( x \) is to \( x'_i \), the more likely the object has to be put in the same category of the object associated with \( x'_i \). The \( \gamma \) parameter defines how far the influence of a representative reaches: small values of \( \gamma \) indicates influence on points that are far away. In other words, the \( \gamma \) parameter can be seen as the inverse of the radius of influence of support vectors in \( X' \).

A.1.3 Linear inseparability, parameter optimization, and feature scaling

As said, SVM aims at finding the hyperplane that better separates the space between positive and negative examples, but in some cases this hyperplane fits on training data but is not general enough for later classification. This problem is commonly called over-fitting. In these cases, it is actually desirable to have a model that, despite being less precise on the training set, is general enough to perform better as a predictor.

Model training is not only aimed at finding the hyperplane that minimizes misclassifications on the training set, but also accepts that a number of misclassified objects of the training set are ignored, in gain of a wider margin.

The balance between fitting the model on the training dataset and generalizing on new objects is chosen by the regularization parameter \( C \), that indirectly regulates how many objects can be misclassified in gain of a wider margin. Small values of \( C \) bring higher margins, allowing a higher number of misclassifications of outlier objects, thus a less precise but more general model. Higher values of \( C \) bring more precise, less general models.

Per-category weights \( p_+ \) and \( p_- \) are multipliers of \( C \) that are used during training depending on an object’s category. Intuitively, category weights determine how much we agree to ignore outliers for each category in gain of a wider margin. If a category is more noisy and has a higher ratio of outliers, it has to be assigned a lower weight. In other words, weights can be seen as the confidence in that category.

To sum up, training an SVM binary classifier with RBF kernel depends on four parameters: parameter \( \gamma \) explained above, regularization parameter \( C \), positive and negative weights \( p_+, p_- \).
As the math above suggests (see Inequality A.2), if values of a feature have a much wider range with respect to another feature, it would dominate the classification process. For this reason, it is common practice to scale the value of features with Z-scaling, which scales each feature’s distribution to a Normal distribution, having mean of 0.0 and variance of 1.0.

A.1.4 Support Vector Machines for Regression (SVR)

Though SVM has been originally proposed for categorization, the intuition behind it has also been applied to regression by Smola and Vapnik (1997). In regression, we are given a set \( T \) of training examples \( \vec{x}_i \in \mathbb{R}^d \), each associated with a real value \( y_i \in \mathbb{R} \) called label, and our goal is to find a function \( f \) such that \( \forall \vec{x}_i \in T, f(\vec{x}_i) = y_i \). We can then use \( f \) to predict new values associated to previously unseen vectors in \( \mathbb{R}^d \).

Smola and Vapnik (1997) propose \( \epsilon \)-SVR, a formulation of regression that ignores errors on the training set no larger than \( \epsilon \), i.e. we relax the constraint over \( f \) to

\[
\forall \vec{x}_i \in T, \ |f(\vec{x}_i) - y_i| \leq \epsilon
\]

This is motivated by the need to accept some degree of error on the training set in gain of a flatter function \( f \). \( \epsilon \)-SVR finds a function of the form

\[
f(x) = w \cdot x + b
\]

where \( \cdot \) is the dot-product, \( w \) is a \( d \)-dimensional vector and \( b \) is a real number. The reader may note a similarity with the SVM classification function (Equation A.1).

In real-case scenario, such a function may not exist or be over-fit on the training set, not being able to generalize to new examples. To deal with this, the optimization problem for finding \( f \) is further relaxed, and defined by means of a loss function. Among the (generally infinite) set of functions that satisfy the constraints, \( \epsilon \)-SVR searches for a function as flat as possible. The function being flat corresponds to vector \( w \) having small norm \( ||w||^2 \).

Similarly to SVM classifiers, in \( \epsilon \)-SVR vector \( w \) is defined as the sum of a subset \( X' \subseteq T \) of support vectors, hence the function \( f \) can be rewritten in the form:

\[
f(x) = \sum_{x'_i \in X'} (\alpha_i - \alpha_i^*) x'_i \cdot x + b
\]

The optimization process results in the choice of the set of support vectors \( X' \) and associated Lagrangian multipliers \( \alpha_i, \alpha_i^* \).
As in the original SVM, we can replace the linear dot product \( x_i \cdot x \) with a kernel function, obtaining non-linear SVR, with outcomes similar to SVM classifiers.

### A.1.5 Feature selection by ablation

In some cases, overfitting may be caused by non-significant features on which the model “adapts”, building a model that performs best on the training set but is not general enough. For this reason, parameter optimization also aims at defining the optimal set of features to consider. In this thesis, we employ a simple feature selection process by ablation: starting with the full set of defined features, we check the performance of the model on a development set, separated from the training set, after features are removed one by one. A feature is discarded if its presence leads to worse or equal results on the development set. Eventually, the minimal, best performing set of features is found.

Feature selection by ablation works by searching, among the set of features, the subset that leads to the best value according to an objective function. This objective function is computed on the development set, which is separated from the training set. Let \( F \) be the initial set of features and \( r(F) \) be a method that returns the value of the objective function (higher is better) over the development set of a model trained with features \( F \); the algorithm returns the minimal set of features that achieves the best score. The feature selection algorithm is defined as follows:

**Algorithm A.1 feature ablation**

1: \( b \leftarrow \text{false} \)
2: \( \text{while not } b \text{ do} \)
3: \( f \leftarrow \arg \max_{f \in F} (r(F - f_i)) \)
4: \( \text{if } r(F - f) \geq r(F) \text{ then} \)
5: \( F \leftarrow F - f \)
6: \( \text{else } b \leftarrow \text{true} \)
7: \( \text{return } F \)

In our applications, method \( r(F') \) computes the value of the objective function on the full pipeline that would result in the model being trained with the set of features \( F' \).

### A.2 Learning-to-rank with LambdaMART

In this section we provide an overview of LambdaMART (Burges, 2010b), a machine learning algorithm used to construct ranking models. Say we have a set of objects \( S \) we need to rank. Formally, learning to rank is the process of learning a
bijective function $R : S \mapsto [1, |S|]$. In other words, a function that defines a total ordering over objects in $S$.

This algorithm will be used in the SMAPH-2 system (Section 8.4) for ordering candidate solutions for a given query, and pick the best one according to the order defined by the function $R$. The ranking function $R$ will be trained by maximizing the Normalized Discounted Cumulative Gain (NDCG). It has been proposed in [Jarvelin and Kekalainen (2000)]. Let us start by discussing it.

### A.2.1 (Normalized) Discounted Cumulative Gain

NDCG is a popular function for evaluating ranking of documents. It has been defined in the field of information retrieval and is a normalized version of the Discounted Cumulative Gain (DCG). Both NDCG and DCG serve the purpose of assigning a quality score to a ranking. For a ranking over set $S$ we can define its associated list of labels $L = [l_1, \ldots, l_i, \ldots, l_{|S|}]$, representing the fact that the element that has been assigned to the $i^{th}$ position has a relevance score of $l_i$ (higher score indicates more relevant document). DCG is defined as:

$$ DCG@k = \sum_{i=1}^{k} \frac{2^{l_i} - 1}{\log_2(1+i)} $$  \hspace{1cm} (A.3)

where $k$ is the truncation level ($\text{NDCG}@k$ measures the quality of the top-$k$ objects according to the ordering, ignoring objects after $k$).

The DCG measure is higher if relevant documents are assigned a higher rank (a lower index), and lower if irrelevant documents are assigned a higher rank. Relevance of documents in the top positions is weighted more than in lower positions, and this intuitively reflects the fact that it is more important to get right objects in the first positions and less important to get wrong objects in lower positions.

For example, if a set is made of three objects $S = \{d_1, d_2, d_3\}$ respectively having relevance scores 2, 3, 1 (meaning $d_3$ is the most relevant object), and a ranking function defines the ranking $r = [d_1, d_2, d_3]$ (meaning $d_1$ is in first position), then the associated labels are $L_r = [2, 3, 1]$, and $\text{DCG}@3 = 3.0 + 4.416 + 0.5 = 7.916$. Note that in this example, the ideal ranking would be $R^* = [d_2, d_1, d_3]$, with associated labels $L^* = [3, 2, 1]$, that would lead to $\text{DCG}@3 = 7.0 + 1.893 + 0.5 = 9.393$.

The NDCG is a version of DCG normalized in $[0, 1]$:

$$ \text{NDCG}@k = \frac{\text{DCG}@k}{\max \text{DCG}@k} $$  \hspace{1cm} (A.4)

where the denominator is the DCG@k obtained by the ideal ordering (in the example, $R^*$). It is simply the value of DCG@k of an ordering divided by the DCG@k
of the ideal ordering. In the previous example, it would be

\[ NDCG@3 = \frac{DCG@3(L)}{DCG@3(L^*)} = \frac{7.916}{9.393} = 0.843 \]

### A.2.2 Gradient Boost and LambdaMART

After we have defined NDCG, let us give an overview of LambdaMART and eventually explain how the latter uses the first as a scoring function.

LambdaMART (Burges, 2010b) combines LambdaRank (Quoc and Le, 2007) with the Gradient Boosted Regression Trees (GBRT) framework (Friedman, 2000). Let us start by describing the main idea behind GBRT.

GBRT is a non-parametric statistical learning technique for classification and regression. It generates a model consisting in an *ensemble of regression trees* (the ensemble is also called a *forest*). Regression trees are decision trees where nodes are pairs \( \langle i, t \rangle \) (where \( i \) is an index of the feature vector and \( t \) is a real-valued threshold) and leaves contain real values. Given a feature vector \( \vec{x} \) of dimension \( d \) associated to an object, at prediction time the tree is traversed starting from the root, and for each node \( \langle i, t \rangle \), we follow the path on the left if the value of feature \( x_i \) is lower than \( t \), on the right otherwise. Eventually we reach a leaf containing a real value, and that is the output for \( \vec{x} \) for that specific tree. The process is repeated for each tree in the ensemble, and the final output of GBRT is a linear combination of the values given for \( \vec{x} \) by each tree. Note that a single regression tree can be viewed as a partition of the \( d \)-dimensional space, where each region is assigned a real value, hence a tree can be viewed as a function \( h \) that maps a feature vector (a point in the \( d \)-dimensional space) to a real value. The output of the whole model can be written as:

\[
F(x) = \sum_{m=1}^{N} h_m(\vec{x})
\]

(A.5)

where \( N \) is the number of regression trees in the ensemble and \( h_m \) is the function associated to tree \( m \).

The question is: how are regression trees built? GBRT builds the ensemble of trees incrementally. Given a model \( F_{m-1} \) consisting of \( m - 1 \) regression trees, a new regression tree \( h_m \) is built so to correct model \( F_{m-1} \). Model generated at step \( m \) can be written as

\[
F_m(\vec{x}) = F_{m-1}(\vec{x}) + h_m(\vec{x})
\]

(A.6)

and tree \( h_m \) is chosen to minimize a given loss function \( L \), usually least squares:
A.2. Learning-to-rank with LambdaMART

\[ h_m = \arg \min_{h_m} \sum_{i=1}^{\lvert T \rvert} L(y_i, F_{m-1}(\vec{x}_i) + h_m(\vec{x}_i)) \]  

(A.7)

where \( T \) is the training set.

As shown in Equation (A.7), at step \( m \), we add to the ensemble a new tree \( h_m \) such that, for each point \( \vec{x}_i \) of the training set \( T \), we minimize the loss function \( L \), namely the distance between \( y_i \) (the ideal label associated to \( \vec{x}_i \)) and \( F_{m-1}(\vec{x}_i) + h_m(\vec{x}_i) \) (the score computed by the would-be model \( F_m \) if we added tree \( h_m \)).

Minimization is solved numerically using steepest descent, that is the direction of the negative gradient of the loss function evaluated at the previous model at step \( m - 1 \). Unfortunately, in order for a gradient to be computable, we need a continuous function (such as least squares), but in learning-to-rank we need to optimize NDCG, whose value is flat except in the points where the order of object changes, where the function has a point of jump discontinuity. This makes the gradient of NDCG either zero or non-computable.

To overcome the fact that NDCG is not pluggable into GBRT as-is, LambdaMART borrows a key idea behind LambdaRank: let \( (\vec{x}_i, \vec{x}_j) \) be a pair of training points, and let their associated relevance be respectively \( y_i \) and \( y_j \). For each such a pair where \( y_i > y_j \) (i.e. object \( i \) should have a higher rank than object \( j \)), we compute a value \( \lambda_{i,j} \) through a continuous function that acts as the gradient that we need. The \( \lambda \)'s can be interpreted as forces: in case \( \vec{x}_i \) should have a higher rank, it will get a “push” upwards of magnitude \( |\lambda_{i,j}| \). On the contrary, \( \vec{x}_j \) will receive a push downward of the same magnitude.
References


References


References

SIGKDD international conference on Knowledge discovery and data mining, KDD ’09, pages 389–396, New York, NY, USA. ACM.


References


Papers published by the author during his Ph.D.


Acknowledgments

During my PhD I had the privilege to get to know truly impressive people. The impact that the ideas exchanged with them had on me is hard to quantify, but this thesis would definitely be poorer without their contribution. My advisors Paolo Ferragina and Massimiliano Ciaramita have always been ready to offer advice, not only of strictly scientific nature. Thank you for counseling me with dedication and for being such great mentors. A major part of this work is the fruit of a cooperation with Stefan Rüd and Hinrich Shütze (University of Munich), whose open mind and impressive knowledge of anything related to NLP gave me important directions. This thesis was reviewed by Gerhard Weikum and Soumen Chakrabarti, whom I thank for the insightful comments and the interest shown. Other members of our research team were crucial: Francesco Piccinno (whose creativity inspired me to find who to blame in case of broken code) and Marco Ponza (who taught me how to properly order a hamburger).

During my internship at Google Zurich I had the opportunity to work with Enrique Alfonseca, Katja Filippova, Daniele Pighin, Dimitris Leventeas, Ioannis Tsochantaridis, Evangelos Kanoulas and Yasemin Altun. Thank you guys for making me feel at home from the very first day. My visit at the University of Munich let me meet with Ryan Cotterell, Wenpeng Yin, Heike Adel, Luciadwi Krisnawati, Thang Vu, Katharina Kann, Sebastian Ebert, Irina Sergienya and Yadollah Yaghoobzadeh. Thank you all for making my stay at CIS such a great time.

I’d like to thank all fellow PhD students at the University of Pisa: among others, my office buddies Tiziano De Matteis and Lillo Galletta, Andrea Farruggia, Rita “black hole” Pucci, Giovanna Broccia, Alessio Conte, Andrea Marino. Thank you guys for the laughers, mutual support and food shared during these years.

Thanks to my parents Polina and Fulvio (who taught me to love science), my brother Davide, Iliana and little Carola.

Thanks to Cesare, Gigia and Piu for being fluffy.

These years of PhD were challenging, frustrating, demanding, joyous, rewarding, satisfactory. While life was unfolding, one person got the closest to my heart, always being by my side. Thank you Simona, for loving me thoroughly.