UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

DOTTORATO DI RICERCA IN INFORMATICA

PH.D. THESIS

Improving the Efficiency and Effectiveness of Document Understanding in Web Search

Salvatore Trani

SUPERVISOR

Raffaele Perego

REFEREE

Srinivasan Parthasarathy

REFEREE

Jimmy Lin

"A true scientist must never be satisfied
there is always scope for improvement"
— Srinivasan Parthasarathy

Acknowledgements

I would like to express my deep gratitude to all the people who, consciously and unconsciously, contributed in shaping this thesis during the years of my PhD. Undertaking this PhD has been a truly life-changing experience for me and it would not have been possible to achieve without their support and guidance.

First of all, I would like to express my sincere gratitude to my advisor Raffaele Perego, for his valuable guidance, strong support and consistent encouragement I received during these years. Raffaele gave me the opportunity to undertake my Ph.D. research within the HPC lab of ISTI, CNR and the freedom to choose the topic I liked most. The independence, he granted me, helped me to grow both as a researcher and as a person. His guidance helped me constantly during the time spent in research and writing this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would also like to special thanks to Claudio Lucchese, Diego Ceccarelli and Fabrizio Silvestri, also (present or past) members of HPC lab. Claudio is a very strong researcher. I had many fruitful academic discussions with him and several ideas in this work raised from these discussions. He has been also a great travel mate, and a important guide in my growth as a researcher. With Diego I share the passion for the Semantic Web. It is easy to say that without his strong enthusiasm for the topic, I probably would have taken a different research direction. Last but not least, I would probably not have studied Information Retrieval and Web Mining if it wasn't for Fabrizio. He is genuinely passionate about research, and one of the brightest minds I encountered during my life. His support and suggestions have been invaluable to my scientific growth. I wish to all of them a fruitful and prosperous career.

To my HPC lab mates, thanks for the fun and support. The last four years were anything but simple, but sharing deadlines, hard work, events

and travels with such special people made this achievement less hard to be earned. In particular I would like to individually thank Franco, Cris, Daniele and Matteo for their precious help and encouragement. More than colleagues, they are good friends. Thanks also to the Brazilian guys with whom I shared the office for almost one year, Vinicius, Livia and Regis: if I'm able to understand (despite only partially) Brazilian spoken language is up to them!

Moreover, thanks to the external reviewers, Jimmy Lin and Srinivasan Parthasarathy, for taking the effort to contribute to this work by reviewing it: I am really honored of having their endorsement. I am also grateful to my committee members Rossano Venturini and Roberto Grossi for serving as my committee members even at hardship. They gave me priceless comments and suggestions to enhance the presentation and quality of my Ph.D. thesis.

I am also really grateful to all my friends for the time we spent together and for being the surrogate family during the many years of my stay far away from my parents. Only by learning to detach from the thoughts that run through your mind is lasting happiness found.

Finally, I sincerely and wholeheartedly express my gratitude to all my family for their love and support. In particular, I would like to express appreciation to my beloved wife Federica who spent sleepless nights with me and was always supporting me in the moments when there was no one to answer my queries. And to my little daughter Martina, for the beauty and the sweet she brought into my life.

Abstract

Web Search Engines (WSEs) are probably nowadays the most complex information systems since they need to handle an ever-increasing amount of web pages and match them with the information needs expressed in short and often ambiguous queries by a multitude of heterogeneous users. In addressing this challenging task they have to deal at an unprecedented scale with two classic and contrasting IR problems: the satisfaction of effectiveness requirements and efficiency constraints. While the former refers to the user-perceived quality of query results, the latter regards the time spent by the system in retrieving and presenting them to the user.

Due to the importance of text data in the Web, natural language understanding techniques acquired popularity in the latest years and are profitably exploited by WSEs to overcome ambiguities in natural language queries given for example by polysemy and synonymy. A promising approach in this direction is represented by the so-called *Web of Data*, a paradigm shift which originates from the Semantic Web and promotes the enrichment of Web documents with the semantic concepts they refer to. Enriching unstructured text with an entity-based representation of documents - where entities can precisely identify persons, companies, locations, etc. - allows in fact, a remarkable improvement of retrieval effectiveness to be achieved.

In this thesis, we argue that it is possible to improve both efficiency and effectiveness of document understanding in Web search by exploiting learning-to-rank, i.e., a supervised technique aimed at learning effective ranking functions from training data. Indeed, on one hand, enriching documents with machine-learnt semantic annotations leads to an improvement of WSE effectiveness, since the retrieval of relevant documents can exploit a finer comprehension of the documents. On the other hand, by enhancing the efficiency of learning to rank techniques we can improve both WSE efficiency and effectiveness, since a faster ranking technique can reduce

query processing time or, alternatively, allow a more complex and accurate ranking model to be deployed.

The contribution of this thesis are manifold: i) we discuss a novel machine-learnt measure for estimating the relatedness among entities mentioned in a document, thus enhancing the accuracy of text disambiguation techniques for document understanding; ii) we propose novel machine-learnt technique to label the mentioned entities according to a notion of saliency, where the most salient entities are those that have the highest utility in understanding the topics discussed; iii) we enhance state-of-the-art ensemble-based ranking models by means of a general learning-to-rank framework that is able to iteratively prune the less useful part of the ensemble and re-weight the remaining part accordingly to the loss function adopted. Finally, we share with the research community working in this area several open source tools to promote collaborative developments and favoring the reproducibility of research results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Before the invention of the Web, the vast majority of human knowledge was authored by trustworthy people and printed on books collected in libraries. This situation dramatically changed in the last two decades, with the rapid adoption of Web-based forms of communication that have quickly surpassed any previously known medium. At an accelerating pace, people are producing huge amount of digital information and share it through email messages, web pages, news articles, social networks, blogs.

The net result of this on-going revolution is that we are today overwhelmed with data: data about our daily life, worldwide news, characteristics of products, targeted advertisments, government documents, food recipes, etc. It was observed that there are today about 47 billion pages on the Web [56], and 30 trillion unique URLs [53], 30 times more than the figures observed only in 2008 [5]. This phenomena, known as *Data Deluge* [11], makes it almost impossible for a user to find the information she is searching for by simply browsing the Web. Hence, there is an urgent need for powerful tools to help searching and make use of all information, since the achievable knowledge that can be extracted from data is playing an increasing role in our daily life.

Web Search Engines (WSEs) such as Google, Bing or Yahoo, are by far the most complex and useful tools to combat information overload. They rely on advanced Information Retrieval (IR) techniques designed to help users to find the information satisfying their needs in Web pages. In a broad sense, IR can be described as the activity that deals with the representation, storage, organization and access to information items [10]. However, IR is not limited only to indexing and searching but it concerns also other tasks related to exploit information in general, such as text categorization, clustering and summarization, question answering and information filtering.

1. Introduction

Given a text query expressing the information need of the user, a WSE has to retrieve a small set of Web documents relevant to the query and rank them according to some notion of relevance. To this end, the retrieval process has to interpret both the query and the documents in order to perform the matching, and ranking has to provide the user with a Search Engine Results Page (SERP) in which 10 blue links - the first 2-3 to an extreme - are the most important elements according to the subjective quality she perceives [10].

In order to satisfy user requests, WSEs have to deal with two classic IR challenges: effectiveness requirements and efficiency constraints. The former regards improving the quality of search results by adopting sophisticated matching and ranking models as well as smart relevance signals. The latter consists in reducing as much as possible the response time related to the presentation of the SERP to the user. This is achieved by optimizing all the phases of query processing. In this respect, researchers from Google have shown that when a WSE is even slightly slower than normal to return search results, the delay leads to significant decrease in user engagement [20]. To address these two challenges, large-scale WSEs adopt a multi-stage architecture: i) a large set of documents is retrieved by the first stage using a fast *base ranker* designed to optimize the recall, and ii) these documents are re-ranked by the second stage using complex ranking functions as well as a larger number of relevance signals [26]. The second stage, which usually makes use of sophisticated learning to rank (LtR) models, is possibly split in a ranker pipeline as well in order to take query processing latency under control.

One traditional problem WSEs have to face derives from characteristics of the language which compose both the queries and the documents. Indeed the natural language is not easy to handle due, for example, to polysemy and synonymy, i.e., words with several meanings and words with the same meaning, respectively. Moreover, queries are typically short [80] (31% are composed by a single term, 62% by two terms or less) and often carry some degree of ambiguity. Indeed, 16% are ambiguous, many of the remaining ones can still be too broad to have a single, specific meaning [154].

Several approaches have been proposed to mitigate the effects of ambiguities in the natural language. At the query level, a trivial solution is to completely ignore the problem or to provide results satisfying the most plausible meaning (e.g., the most popular). Smarter solutions can be based on asking the user for a feedback on what she actually mean [9], or on applying a diversification strategy in order to maximize the probability for a user to fulfill her information need with the results in the SERP [45]. Overall, none of these techniques resolve the problem. Indeed,

WSEs generally adopt a bag of words model for documents retrieval (first stage of the WSE pipeline), failing consequently to focus on the exact information need of the user. A promising solution that is starting to be used nowadays by several WSEs to overcome this issue, is to semantically enrich queries and documents [126, 177]. This approach falls into the so-called *Web of Data* [15], originating from the Semantic Web [14]. The *Web of Data* constitutes a first attempt to make a paradigm shift, moving from unstructured to semi-structured data by enriching web documents with semantic tags. By enabling seamless connection between datasets, this approach presents a revolutionary opportunity for deriving insight and value from the data [74].

Figure 1.1 shows a typical usage of the *Web of Data* from a WSE. By searching with the query "tim berners lee", the scientist behind the original idea of the Semantic Web, Google automatically shows on the right side of the SERP a box with the major information about the scientist: some photos, a short description as well as some structured metadata (birth date and place, nationality, education, etc), some popular quotes from him as well as social profiles and links to related people. This information is mainly extracted from Wikipedia or FreeBase, but other trusted sources may contribute to this enrichment, e.g., social media for connections, vertical authoritative websites for specific topic, like *IMDB* for movies or *Google Play Music* for lyrics.

The enabling technology allowing similar info boxes to be automatically built leverages techniques to enrich a text with entities and semantic concepts it refers to. By identifying entities mentioned in queries and documents we can easily manage the ambiguity of natural language and exploit the relations among entities to give more precise answers to users. A possible solution to achieve the above enrichemnt is Named Entity Recognition and Disambiguation (NERD). This approach originates from the Natural Language Processing (NLP) area and adopts a pipeline approach: named entities, i.e., fragments of text possibly referring to the names of people, organizations, locations, are first recognized in the text by means of Named Entity Recognition (NER) algorithms performing a syntactical analysis of the text. Then, these named entities are linked to entities in a given Knowledge Base (e.g., Wikipedia), thus solving named entity ambiguity, i.e., multiple entities with the same name. Alternatively, an increasingly popular solution is represented by Entity Linking (EL), which adopts an orthogonal approach aimed at directly identifying the fragments of text referring to an entity listed in a given knowledge base, without pursuing an intermediate goal by performing syntactic analysis.

The popularity of EL solutions increased a lot in the latest years. Semantic WSEs started to appear that try to go beyond the standard bag of words model in the

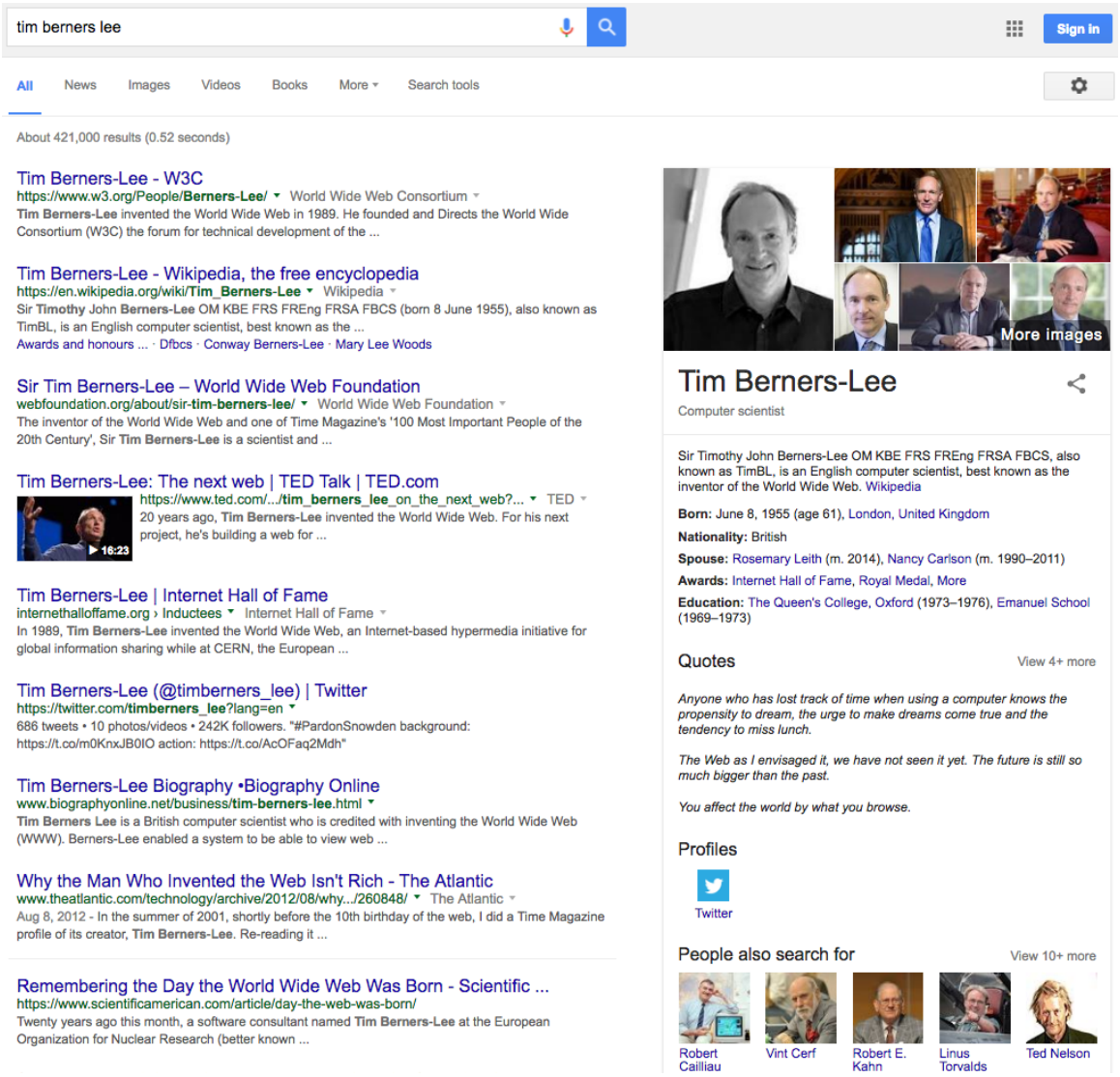

Figure 1.1: Web of Data usage from Google

direction of the bag of concepts model (e.g., Sindice [123]). However, such a paradigm shift is still far from being mature and a lot of research and engineering work has to be done. In this thesis, we go some steps ahead in this direction by introducing efficient and effective techniques based on machine learnt models. Specifically, we contribute to state-of-the-art solutions for document understanding by proposing a entity similarity measure that improve entity disambiguation and by exploiting saliency in the EL process. In addition, we introduce a novel LtR strategy that remarkably improve efficiency and effectiveness of ranking. Finally, we share with the research community open source frameworks with the hope that they can facilitate and boost the advance of research in this field.

1.1 Thesis Statement

The statement of this thesis is that a semantically enriched learning-to-rank strategy can simultaneously improve both efficiency and effectiveness of document understanding and retrieval tasks.

Indeed ranking plays a central role in many IR problems, and many tasks are by nature ranking problems. Among them, web search is probably only the most famous. Learning to rank strategies raised as the de-facto choice when dealing with such ranking problems and labeled data are available for training a ranking model in a supervised manner. The results discussed in this thesis focus on improving the effectiveness of the semantic enrichment with a two-fold contribution: i) a very effective machine-learnt relatedness measure; ii) a proposal to rank the entities mentioned in a text by their relevance and utility in understanding the topics being discussed. The former contribution is based on the assumption that the relatedness measure is the building block of most state-of-the-art Entity Linking algorithms and is responsible for the correct disambiguation of the entities even in other applications. Entity ranking is instead crucial to obtain a fine-grained document understanding, thus permitting to distinguish between main and satellite concepts discussed. We believe that advancements in the two aforementioned tasks can remarkably improve the annotation process and the use of the knowledge acquired. Improving efficiency is, on the other hand, important in every step of the long chain involving the exploitation of web data to satisfy user needs. An key factor of this chain is ranking efficiency that is achievable by using more efficient learning to rank models. To this end, we propose a novel LtR framework targeting models based on ensemble of decision trees. It allows to enhance the learning phase by pruning part of the trees and re-weighting the remaining ones accordingly to the loss function adopted (e.g., *NDCG*). The relevant result of our iterative cut&re-weight process is a ranking model that is more efficient and/or effective of the one trained on the same data and the same algorithm without using our technique. Finally, we claim the importance of having open source frameworks for the research community, both for reproducing research results and for allowing other researchers to go beyond current solutions without having to start from scratch.

1.2 Thesis Contributions

The key contributions of this thesis can be summarized as follows:

1. We propose a novel entity relatedness measure by formalizing entity-relatedness as a learning-to-rank problem and by devising a effective solution to learn from data high-quality relatedness functions. Entity Linking algorithms adopt several signals and features to drive the annotation process. Entity relatedness is the most important of such features, and is the main responsible for a correct disambiguation process. Most state-of-the-art Entity Linking algorithms adopt a Wikipedia-based relatedness function proposed by Milne and Witten [116, 115]. In this thesis we show that our machine-learnt relatedness measure is much more effective and, more importantly, improves the overall performance of EL algorithms. The work discussing our proposal previously appeared in a CIKM 2013 paper [37].

2. We introduce a novel algorithm comprehensively addressing both Entity Linking and Saliency detection. The Salient Entity discovery problem has been initially proposed in literature by Paranjipe [127] and Gamon *et al.* [68]. The task aims at labelling the entities mentioned in the document according to a notion of saliency, where the most relevant entities are those that have the highest utility in understanding the topics discussed. This problem is of fundamental importance for document understanding and impacts on a variety of IR tasks. We model the problem as a supervised two-steps algorithm, where the first step is devoted at identifying a small set of candidate entities, while the second, besides detecting the entities that actually occur in the document, also predicts their saliency. This work appeared in ACM DocEng 2016 [164] and was awarded with the Best Student Paper. An extended version of this paper, experimentally assessing the impact of saliency detection on text summarization, is currently under review for publication in an international journal [165].

3. We present X-CLEAVER, a novel meta-learning algorithm aimed at improving both efficiency and effectiveness of ensemble-based models trained with state-of-the-art LtR algorithms. The proposed solution entails two major steps: i) a pruning strategy that delete the redundant trees from the ensemble aimed at making the ensemble more efficient without hindering effectiveness; ii) a greedy optimization responsible for fine tuning the weights of the trees to maximize effectiveness according to the loss function adopted. We investigate several pruning strategies for selecting the less relevant trees to be removed, and exploit a line search algorithm to optimize non-derivable ranking loss function (i.e., *NDCG*). The above two steps are integrated in our X-CLEAVER meta-learning

algorithm that interleaves the learning of a set of trees with the pruning and re-weighting of some of them. A preliminary version of this work appeared in a ACM SIGIR 2016 short paper [98] while the extended version presented in this thesis is currently under review for publication in a top-tier journal [99].

[**salvo**: *Togliere l'elemento sottostante dalla bullet list e metterlo come subsection dal nome "minor contributions" o "Indirect Contributions" ?*]

4. We made publicly available several open source frameworks, mainly developed for validating the proposed solutions. The first framework is DEXTER and it address the Entity Linking problem. It provides out-of-the-box efficient implementations of some popular EL algorithms, while its modularity makes easy to integrate and test other solutions. It was presented in a CIKM 2013 workshop (ESAIR 2013) [36] and then extended as described in a demo paper presented at ISWC 2014 [38]. The second is ELIANTO that was introduced in a demo paper at ACM CIKM 2014 [163]. It aims at crowd-sourcing the production of manually annotated datasets for Entity Linking and Saliency Detection. The third framework is QUICKRANK, a modular and extensible LtR framework designed with efficiency in mind. QUICKRANK provides efficient solutions for both the learning and the scoring processes. This framework was independently introduced in [30], while, in the scope of this thesis, it was extended to incorporate the work presented in a ACM SIGIR 2016 short paper [98], and the final solution presented in this thesis which is currently under review for publication in a top-tier journal [99].

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 discusses background information about the main topics involved in this work. We start by introducing the Web of Data, an approach for semantically connecting data in a single, massive graph, such that computers can conceptually understand documents and perform complex operations on them such as inferring meanings or answer questions. Then we introduce the ranking problem with a specific focus on the learning-to-rank approach, a machine-learnt technique that is pervasively used in a variety of IR tasks: from document retrieval (e.g., web search) to sentiment analysis, document summarization and information filtering. The chapter ends with a discussion about retrieval evaluation in IR

and the metrics adopted for a proper evaluation of the experiments conducted in this thesis.

- Chapter 3 proposes a new entity relatedness measure for Entity Linking. The problem has been formalized as a learning to rank problem, and the resulting measure has been compared with state-of-the-art solutions when used in an original algorithm and incorporated in other state-of-the-art EL algorithms.
- Chapter 4 starts by discussing the importance of the Salient Entity discovery problem for the semantic enrichment of text. Then it introdices the formalization of the problem by proposing a novel algorithm for both Entity Linking and Salient Entity detection. The solution proposed adopts two different machine-learnt models motivated in a two-steps process motivated by efficiency and effectiveness reasons.
- Chapter 5 proposes a novel meta-learning algorithm that encompasses the benefits of existing state-of-the-art LtR models based on additive ensemble of small regression trees with. Our solution aims at reducing the similarity between the trees and consequently at improving both efficiency and effectiveness of the resulting model.
- Chapter 6 closes this thesis by providing a summary of the contributions, as well as presenting some possible future research directions exploiting our results.

Chapter 2

Background

Web Search Engines are probably the most popular IR systems nowadays. Their goal is to present to the users Search Engine Result Pages (SERPs) containing links to documents relevant to the user's query. In doing this, they need to satisfy two important requirements: i) a strict constraint on the response time, since a small delay in answering a query negatively affects user engagement [20]; ii) a high quality of search results, since users expect to find what they are searching for. The former requirement is addressed by the adoption of efficient IR models, and in this regards ranking technologies play a central role. The latter requirement, on the other hand, is more difficult to achieve and requires to address effectively multiple search sub-tasks (e.g., improving the candidate retrieval process or the final ranking, understanding the exact meaning of both the queries and the documents, etc). To this end, learning to rank technologies and semantic enrichment of text are providing an important boost to efficiency and effectiveness of Web search.

With this in mind, Section 2.1 introduce the Web of Data and discuss its challenges as well as the benefits provided. Moreover, it explains how to enrich unstructured documents with structured information. Section 2.2 describes the importance of ranking in IR, with an overview on conventional ranking models and a specific focus on learning to rank methodologies. Lastly, Section 2.3 discusses the main evaluation methodologies and IR metrics, adopted for assessing the solutions proposed in this thesis.

2.1 Web of Data

The World Wide Web has been conceived as a global space where to share knowledge by simplifying the creation and the publishing of Web documents. One of the main reasons it became so popular is related to the simple format used to describe the Web

pages, the Hyper Text Markup Language (HTML), and in particular to the hyper links linking to other Web pages. These links on the one side allow users to traverse the Web, on the other are exploited by modern Web Search Engine to crawl the Web discovering new content and infer relevance of a page by analyzing the network structure of the Web (e.g., PageRank [124] and HITS [88, 70]).

While HTML simplicity favored an unconstrained growth of the Web, it represented also a limit. Indeed HTML was designed to give structure to textual documents, without considering the data perspective. To this regard, it is commonly accepted that we are surrounded by data: data in HTML tables, spreadsheets, linked files (CSV, JSON, XML), textual statements, etc. However, we are not able to use this huge amount of data because it is presented in an unstructured way, i.e., it is difficult to automatically extract and process without introducing noise and errors. Moreover, the way hyper links are described limits the expressiveness of the connections, since it is not possible to state which kind of relation exists between two concepts in the source and target Web pages.

The Web of Data has been specifically proposed to go beyond the traditional Web of Documents model, by placing data at the heart. This model shares the same rules that made popular the traditional Web (i.e., anyone can publish any kind of data, not limiting to a specific vocabularies; it is highly suggested to create connections between entities in order to favor the browsing and the discover of new things; there is not a central authority which control what to publish and what to discard, etc.). However, it has been conceived considering the properties of the data and favoring its re-usability. To this end, the Web of Data encourage the sharing of structured information, hyper linked by typed relationship between entities described in vocabularies. These vocabularies are not strictly fixed, and could belong to different data sources, thus favoring the existence cross links between the datasets. The main rationale behind promoting the usage of the Web of Data is to allow a machine to consume data, infer new things, and in general automatize some tedious tasks previously hand made by humans.

2.1.1 Principles of Linked Data

The principle behind the Web of Data has been initially proposed in 2001 by Berners-Lee with the Semantic Web [14], and then refined and formalized in 2006 with the Linked Data design issues [13], where the same author stated a set of 4 rules for publishing data in such a way that the published data will become part of a single, global data space of structured data:

1. use URIs as names for things
2. use HTTP URIs, so that things are dereferenceable
3. provide useful information when a URI is looked up, using standard technologies (e.g., RDF, SPARQL)
4. include links to other URIs, so that to promote the discovery of new things

The linked data principles are based on the assumption that the Web of Data has to extend the traditional Web of Documents, and not revolution it. Indeed, it adopts the same decentralized architecture [79] and the same technologies: URIs as a global unique resource identifier [105], the HTTP protocol to access resources [62] and the HTML to describe documents. The hyper links between documents are extended with respect to the original hyper links, so that to include the typology of the relation in the link description. This is put in place by using the Resource Description Framework (RDF) [89], a model that encodes data in a triple composed by the subject, the predicate and the object. Both the subject and the object are URIs to entities/concepts belonging to a given vocabulary, while the predicate is a URI as well, but describing the king of relation that exists between the subject and the object (e.g., belongs to, is member of, is made by).

The four principles stated above claims that everything should be identified by a URI (not only Web pages but also abstract concepts or concrete entities) and that by looking up that URI using the HTTP protocol, a description of the concept has to be returned to the user using standard technologies (RDF among others). Moreover, hyper links to other resources (in the form of RDFs as well) have to be proposed to the user, including the typology of relationship that exists between the two concepts.

Web of Data and Linked Data are two closely related concepts. Indeed, while the Web of Data state the vision (putting data on the Web in such a way that machines are able to directly or indirectly process it), Linked Data state the principles and the means to reach that goal.

2.1.2 Topology of the Web of Data

The formal definition of the Linked Data principles [13] was given by Berners-Lee in 2006. A year later the W3C started the so called Linking Open Data (LOD) project[1], an attempt to bootstrap the Web of Data with the help of researchers of

[1] http://linkeddata.org

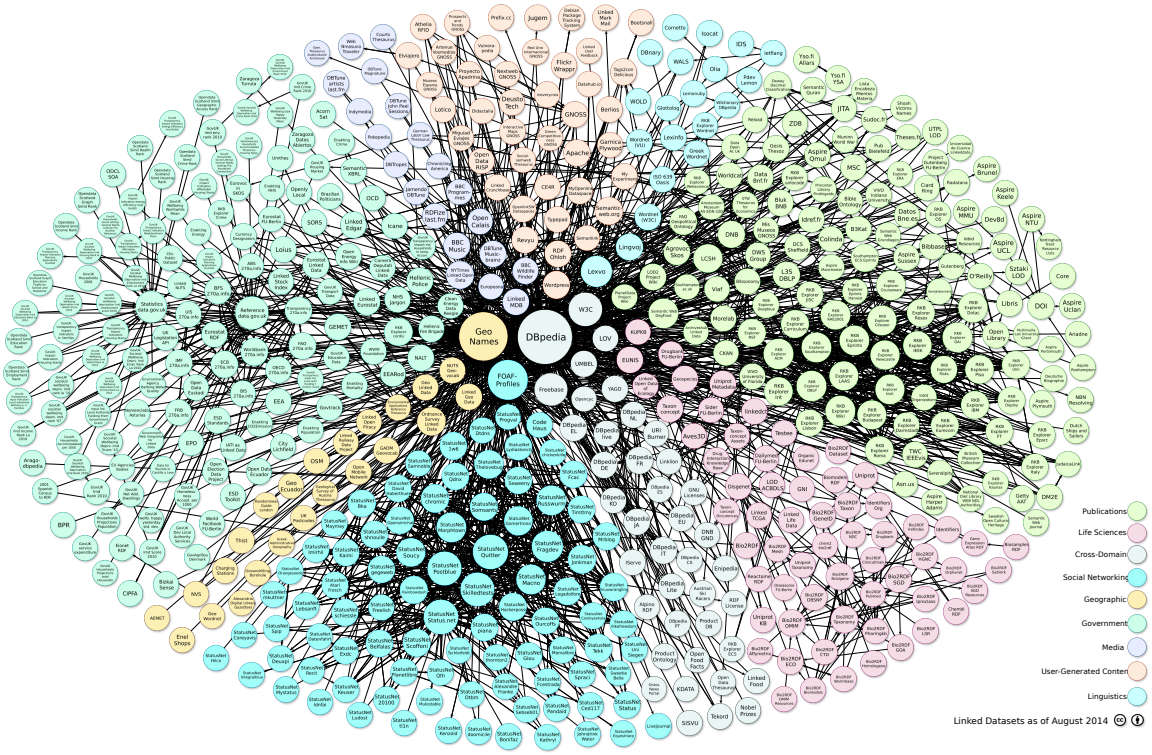

Figure 2.1: Topology of the Linked Data cloud diagram, with connections between different resource datasets

the semantic community. The project grew up quite fast. An overall picture of the datasets composing the Web of Data, as of April 2014, can be observed in Figure 2.1.

There are plenty of data resources (1041) subdivided in 8 major topical domains, ranging from (in order of datasets number) Government, Publications, Life Sciences, User-Generated Content, Cross-Domain, Media, Geographic and Social Web. The Cross-domain data resources are those not focused on a specific topic but linking together concepts from different areas, helping consequently to avoid the dispersion and the isolation.

The biggest resource in the Web of Data is represented by DBPedia [8], a cross-domain encyclopedia derived from Wikipedia[2]. Wikipedia is a the biggest encyclopedia ever conceived as well as one of the 10 most visited Website worldwide. The project is based on the concept of *collaborative encyclopedia*, where every user can submit new content or a modification to an existing page. To date, there are about 75.000 active contributors working on more than 41M articles in 295 languages[3]. Every Wikipedia page can be considered like the textual description of a concept/entity. Most of the

[2] https://www.wikipedia.org/
[3] http://reportcard.wmflabs.org/

content is unstructured but structured information are present as well in the so-called *infobox* (vertical box that appear on the top-right part of the page, with a tabular presentation of some data, e.g., geographic coordinates of a place, location and birth date of a person, population and area of a country, and so on), in categories, in links to outside Webpages, etc. The DBPedia project aims at automatically extracting all the structured information from Wikipedia such as to build a structured knowledge base to export using RDF triples. Entities in DBPedia are connected also to other data resources in order to fill the gap between different topics.

Another popular cross-domain data resource is FreeBase [17], a project that shares most of the DBPedia features: open domain, collaborative environment, structured information exported using RDF triples. The project has been acquired by Google in 2010, but in 2015 it has been dismissed and all the data migrated to the corresponding Wikidata [174] project. A further popular cross-domain datasets is YAGO [158], which similarly to DBPedia tries to automatically link structured information from Wikipedia with concepts from WordNet.

In this regard, WordNet [114] is a popular linguistic dataset, providing semantic lexicon for the English language. It provides for each word a list of the actual senses of the word. Different words with the same meaning (synonyms) are called *synset*, while a single word with different meaning (ambiguous) belong to different *synset*. WordNet provides also the semantic relationship between the *synset*, such as hyponymy and hypernymy (i.e., sub-name and a super-name), holonymy/meronymy (i.e., whole-name and part-name), antonymy (i.e., opposite-name) and others.

Despite data resources are abundant both in number and size, the vast majority of semantic resources shared in traditional Web sites is enriched with semantic concepts through microformats [86], RDFa or Microdata[4] annotations. The most popular semantic vocabulary adopted is the one encouraged by Google, Yahoo, Microsoft and Yandex, namely Schema.org[5]. This is the product of a joint effort to promote the adoption of schemas for exposing structured data on Web pages. There are many important Websites using semantic annotations, like the popular Internet Movie Database (IMDb) Website[6], which exports for each movie the title, the actors, the average rating, the number of voters as well lot of other information, AllRecipes.com[7]

[4] http://www.w3.org/TR/microdata/
[5] http://schema.org/
[6] http://www.imdb.com/
[7] http://allrecipes.com/

which exports for each recipe the list of ingredients, the cooking time and the nutritional facts or Facebook[8]. which exports details about events or profile information.

2.1.3 Semantic Search

Among different linked data applications, Semantic Search is probably the most interesting one. It provides the user with the capability to express its information need without having to take care of the problems deriving from the natural language, such as ambiguity and synonymy. Moreover, the search engines adopting this paradigm exploit the structure of the information to provide a powerful interaction to the user.

Several semantic search engine has been proposed in literature. Probably the most populars are Falcon [41] and Sig.ma [166]. Both of them crawl the Web of Data and index the structured information found in such a way to allow a powerful search experience. The Web interface is still keyword-based, like in popular WSEs. However, the main difference with them is in the exploitation of the underlying structure of the data to offer to the user a better user experience. For example, users can filter the result list by type on Falcon (e.g., selecting only persons or films) or filter out results from unwanted sources on Sig.ma (e.g., low quality sources). In both the cases, the search engine provide richer details about each result of the result list (metadata, links, relations, etc).

Despite the approach to propose semantic search engines for the Web of Data has been profitably investigated, it is clear that applying the same semantic approach to the traditional Web of Documents would be of great interest. To this regard, there is the need to fulfill the gap between textual Web documents and semantic concepts. A twofold problem arises: i) support the semantic retrieval of the documents, i.e., go beyond the standard bag-of-words model using a concept-based document representation, namely a bag-of-concepts model [106, 157, 176]; ii) enrich keyword queries with concepts, solving the difficulty of having short (or null) contextual text useful for disambiguation [108, 109, 112]. A panoramic view on semantic search is provided by Mangold [103], while an investigation of specific problems related to the semantic annotation, indexing, and retrieval can be found in [87]. This dissertation however is not about the specific problems above mentioned, but on general strategies able to close the gap between textual and conceptual representation.

[8] https://www.facebook.com/

2.1.4 Semantic Enrichment

Understanding the meaning of textual documents is one of the main goals in the field of Artificial Intelligence and it is a crucial activity for Web search and IR in general. Indeed, the vast majority of Web documents have a textual representation without any semantic information. Thus the related document understanding problem has attracted the interest of many researchers [170, 3, 160], and also a conference has been dedicated to this task [119] (the Document Understanding Conference, DUC, which from 2008 became part of the Text Analysis Conference, TAC).

One way to go beyond the standard textual representation of the documents is to enrich them with concepts. This process can be done in two ways: i) by asking users to manually annotate the documents or to add some semantic tags (e.g., by means of microformats [86]); ii) by building tools which are able to understand a document and automatically enrich it with links to unambiguous concepts. While the first solution is clearly the most accurate, it does not scale, mostly when compared with the size of the Web. Consequently a new research direction, which investigate solutions to enrich raw text with semantic concepts, acquired popularity in the last decade.

This semantic enrichment task is known in Literature with several names: Named Entity Recognition and Disambiguation), NERD (Named Entity Disambiguation), NEL (Named Entity Linking), Wikification or simply EL (Entity Linking). In the following of this dissertation we will refer to this task with the last acronym. EL has been introduced in 2007 by Mihalcea and Csomai [111], and consists in finding small fragments of text (hereinafter named *spots* or *mentions*) referring to an entity that is listed in a given knowledge base, e.g., Wikipedia. Natural language ambiguity makes this task non trivial. Indeed, the same entity may be mentioned with different text fragments, and the same mention may refer to one of several entities.

As an example, consider the annotations performed by an EL algorithm that uses Wikipedia on the following text:

> **Maradona** (→Diego_Maradona) played his first **World Cup tournament** (→FIFA_World_Cup) in 1982, when **Argentina** (→Argentina_national_football_team) played **Belgium** (→Belgium_national_football_team) in the opening game of the **1982 Cup** (→1982_FIFA_World_Cup) in **Barcelona** (→Barcelona).

An algorithm performs Entity Linking task by first spotting the fragments of text that are likely to refer to some entity, e.g., spots **Maradona** or **Belgium**. For each spot, a list of candidate entities is generated. Then, the algorithm proceeds by

trying to link each spot to the correct entity, e.g., links the spot **Maradona** to the corresponding Wikipedia page[9]. Due to the presence of multiple candidates for each spot and to the inherent ambiguity of natural language, the disambiguation phase of the Entity Linking process is not trivial, e.g., the mention **Belgium** does not refer to its most common sense, i.e., the country, but rather to its national football team[10]. A final stage of pruning discards annotations that are considered not correct or consistent with the overall interpretation of the document.

The spotting phase is usually performed by exploiting a catalog of named entities, or some knowledge base, to devise the possible mentions of entities occurring in the text. A popular solution is to resort to Wikipedia [73, 116]: each Wikipedia page is treated as an entity, and the anchor texts of the links pointing to each article, as well the title of the article, are considered a source of possible mentions to the entity. The spotter can thus process the input text looking for any fragment of text matching any of the Wikipedia mentions, and therefore potentially referring to an entity. This phase is of fundamental importance for the whole Entity Linking process, since the coverage of the source knowledge base and the accuracy of the spotter have a strong impact on the recall of the entity linking system [39].

The disambiguation phase exploits several signals and features to select the best entity for each spot[11]. The most important of such features is a coherence[12] function measuring the similarity between two entities. The typical approach to disambiguate is thus to maximize the coherence among the selected entities in the document.

Several Entity Linking approaches have been proposed in literature. For a thorough overview and analysis of the main approaches and their evaluation refer to the survey by Shen *et al.* [151]. However, Sections 3.3 and 4.2 provides a short description of the most popular Entity Linking systems [116, 73, 61, 179, 52, 58, 110, 42, 137, 78, 128].

Entity Linking can be considered as an extension of the Named Entity Recognition (NER) [117, 113, 136], where the objective is to identify fragments of text referring to a named entity, i.e., classification of the fragment in a small set of pre-defined categories such as people, organization, events, locations, etc. NER can also be executed as a preliminary task to Entity Linking: in this case, the candidate list of potential entities of each fragment has to take into consideration the class assigned by the NER tool to

[9] https://en.wikipedia.org/wiki/Diego_Maradona

[10] https://en.wikipedia.org/wiki/Belgium_national_football_team

[11] There exists a variant of the traditional Entity Linking process that links also unlinkable mentions, i.e., mentions without a corresponding record in the knowledge base. This specific problem, known in literature as the NIL problem [93], is out of the scope of this thesis.

[12] What is the coherence and how it is approximated is the topic of Chapter 3

the fragment, i.e., producing only entities related to persons if the class assigned is person.

2.2 Ranking

Ranking plays a central role in many information retrieval tasks. Indeed, many IR tasks, although not being by nature ranking problems, can be solved by applying a ranking strategy. Consider for instance Web spam [48], Opinion Mining and Sentiment Analysis [125], Question Answering [172], Recommender systems [25], Clustering [159]. In the context of Web Search, "ranking is the hardest and most important function search engines have to execute" [10]. Indeed, traditional Web search engines adopt a multi-stage architecture, where the first stage aims at retrieving a small subset - which run into thousands - of relevant documents, maximizing the recall and using fast *base rankers*, while subsequents stages are devoted to re-rank these documents by means of more complex ranking models using a larger number of features. A ranking function can be defined as a function $f(q, d)$ that takes in input a query q and a document d, taken from a collection D, and produces in output a list R_q of documents, according to their degree of relevance to the user query.

Over the past years, a multitude of IR ranking models have been proposed in literature for the document retrieval problem. In Section 2.2.1 we introduce common conventional ranking models. These models, although being very simple, are still used nowadays as features for more complex models trained using learning to rank approaches, as described in Section 2.2.2.

2.2.1 Conventional Ranking Models

Conventional ranking models can be roughly split in two different categories, depending on the evidence it leverages from the query q. On the one side we have *query-dependent* models, which rank the documents according to their relevance to the query, and on the other side *query-independent* models, that rank the documents according to their own importance, independently from the query but pertaining a global state.

2.2.1.1 Query-dependent models

One of the earlier model to be used in IR for document retrieval, belonging to the former category, is the Boolean model[10], described as a pure retrieval model, based on the set theory and boolean algebra. It consider whether the query terms are presents or not in each document, thus predicting if a document is relevant for a given

query, expressed using terms and basic boolean operators. Despite the simplicity of the model, one of the major drawbacks is that it is not properly a ranking model since it returns flat sets of documents, thus leading difficulties in the presentation of the results to the user (i.e., how many relevant documents to retrieve and which ones to select, among the relevant ones).

To this end, a substantial improvement is carried by the so-called term-weighting strategy. The rationale behind this model is that the terms are not equally important in describing the content of a document, e.g., consider the information carried by a very common word compared to a rare word. Consequently, to each term is associated a weight, and both documents and queries are represented as vectors in a Euclidean space [147]:

$$\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j})$$
$$\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q})$$

where $w_{1,q}$ is the weight associated with the term-query (or term-document) pair, and t represents the number of terms in the collection. Given the two vectors, computing the similarity is simply achievable for instance by the cosine angle between them:

$$cosine(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{||\vec{d_j}|| \times ||\vec{q}||} = \frac{\sum_{i=1}^{t} w_{i,j} \cdot w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

The most popular weighting strategy is TF-IDF, where the term TF stands for *Term Frequency* and describe the frequency of a term in a document [101], while the term IDF stands for *Inverse Document Frequency* and describe the rarity of a term t_i considering the full collection of documents [155, 139]:

$$TF(t_i, d_j) = f_{i,j} \qquad \text{and} \qquad IDF(t_i) = log\frac{N}{n(t_i)}$$

where $f_{i,j}$ is the frequency of occurrence of a term t_i in a document d_j, N is the number of documents in the collection and $n(t_i)$ is the number of documents containing the term t_i. IDF is called inverse document frequency because $\frac{n(t_i)}{N}$ represents the relative document frequency of term t_i in the collection. The logarithm in the IDF formulation is motivated by the power-law modelization of the Zipf's law [182], which states that the relative document frequency of the words can be approximated by a power law probability distribution. Thus, the TD-IDF is simply described as:

$$TF\text{-}IDF(t_i, d_j) = TF(t_i, d_j) \cdot IDF(t_i)$$

Many variants of the TF-IDF model have been proposed, as well as many improvements to the original formulation (i.e., considering the document length normalization). Interested readers can refer to the work by Baeza-Yates and Ribeiro-Neto [10], which provides a comprehensive and exhaustive panoramic of IR models.

The vector model used in conjunction with the TF-IDF weighting schema gathered lot of attentions in the latest years and is still used nowadays in many areas due to its simplicity. It is based on the assumption that the terms are mutually independent each other. A different approach is represented by the *probabilistic models* [142], where the relevance of a document to a query follows the Probability Ranking Principle (PRP) stated by Robertson [141]:

> *"If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system will be the best that is obtainable on the basis of that data."*

Many probabilistic models have been proposed in literature. Probably the most popular used in IR is represented by BM25 [140], which is the latest version of several best-matching (BM) probabilistic models proposed by Robertson [143, 144]. This model is described as follows:

$$BM25(d_j, q) = \sum_{t_i \in q} \frac{IDF(t_i) \cdot TF(t_i, d_j) \cdot (k_1 + 1)}{TF(t_i, d_j) + k_1 \cdot (1 - b + b \cdot \frac{len(d_j)}{avg\text{-}len})}$$

where t_i is a term appearing in the query q, TF and IDF are defined as described above, $len(d_j)$ characterize the length of the document d_j, *avg-len* is the average document length on a collection basis, k_1 and b are two empirical constants whose values can be fine tuned for particular collections (common values for this constants are $k_1 = 1$ and $b = 0.75$).

2.2.1.2 Query-independent models

The common characteristic of the models introduced to this point is that all of them are *query-dependent*, i.e., they model the relationship between the documents and the query and accordingly produce a ranked list of documents. To the contrary, in literature various *query-independent* models have been proposed for the document retrieval problem. Although taking into consideration the query is fundamental to

maximize the relevance of the retrieved documents, it is not sufficient to produce a good ranking. Indeed, many documents could have similar scores, thus being ranked near each other or worse, resulting almost in a random sorting. Moreover, no one of the *query-dependent* models consider the authoritativeness, the reliability or the trustworthiness of the documents, a signal that could be exploits to distinguish between high and low quality documents.

To this end, several models have been proposed to rank document according to their own importance. Some of them approach the problem of devising an effective function $rel(d_j)$ (based solely on the document) by analyzing the readability of the documents [12, 180], by trying to detect spamming techniques adopted by malicious users [122] or by using implicit feedbacks from the users (e.g., analyzing click-through data [83, 135] or the time spent by a user on a given page, after having clicked on the link in the result list [94, 2]). Others, instead, analyze the hyperlink structure of the Web for estimating the importance of a page. The idea behind these models is not completely new, since it derives from a well studied problem aimed at analyzing the scientific literature to exploit the importance of a scientific paper by analyzing its citations. Several models of page importance were proposed, starting from the simple link-counting [91] that quickly become evident to be easily manipulable by malicious users, to something more complex like the WebQuery approach [32], that re-rank a set of Web pages according to a notion of node connectivity.

A more sophisticated solution is the one proposed by Kleinberg [88] in 1999, namely Hypertext Induced Topic Search (HITS). The author proposed to separate the Web pages into two distinct sets: i) *hubs*, pages not actually authoritative but containing many outgoing links to authoritative pages; *authorities*, pages authoritative regarding a specific topic, having many ingoing links from hubs pages. The algorithm operates over a small set S of pages, obtained by retrieving with a *query-dependent* model a subset R of Web pages and then expanding this subset with pages pointed by or that points to a page in R. Then the algorithm assigns both an authority score and a hub score to each Web page belonging S by following an iterative process described as follows:

$$H(p) = \sum_{u \in S:(p,u) \in E} A(u) \qquad \text{and} \qquad A(p) = \sum_{v \in S \mid (v,p) \in E} H(u)$$

where $A(p)$ and $H(p)$ are respectively the authority score and the hub score for the page p, and E is the set of edges connecting the nodes S. This model proved to be effective, although the size of the set R of root relevant pages has to be limited to avoid

an explosion in the size of S, and consequently an explosion in the computational cost of the iterative algorithm (since the number of the edges are potentially quadratic in the number of nodes).

An alternative approach is represented by the PageRank model [124] proposed in the same year by Google creators Page and Brin. This algorithm simulates a user navigating the Web starting from a page p: with a given probability $1 - \alpha$, the user follows one of the outlinks of p, otherwise, with a probability α, he jumps to a random Web page. While the first part simulate the typical user behavior of following link present in a Web page, the second part is useful for avoiding dangling nodes (i.e., nodes without outgoing links) and self links (i.e., links pointing to the same page), as well as to simulate the intention of the user to start a new session from scratch. This navigational process is repeated several times. At the end, the probability an user visited each page is called PageRank and represent a property of the graph. The PageRank can thus be defined as follows:

$$PR(p) = \alpha \frac{1}{T} + (1 - \alpha) \sum_{i=1}^{n} \frac{PR(p_i)}{L(p_i)}$$

where T is the total number of pages in the Web graph, $L(p_i)$ is the number of outgoing links of page p_i and α is a free parameter, namely damping factor, commonly set to $\alpha = 0.15$.

2.2.2 Learning to Rank

In the previous section we described some ranking and retrieval models commonly adopted in IR in general and in Web Search in particular. These models work in an unsupervised manner, thus their parameters, if exists, are usually empirically tuned. It is clear, however, than combining together many of the aforementioned models, allow to create a single, more powerful model that can lead to important improvement in terms of effectiveness. Indeed, each model captures single facets related to the relevance of the documents with respect to the queries, an aspect of fundamental importance given the heterogeneity and complexity of web documents.

Combining together different models, however, is not straightforward. To this end, in recent years many Machine Learning techniques have been devoted to automatically learning an effective model from training data. These techniques are supervised learning methods, since they are designed to automatically learn a generic function using supervised data, i.e., data with relevance judgments. When the task is to train a ranking function, we refer to this techniques as *learning to rank* models.

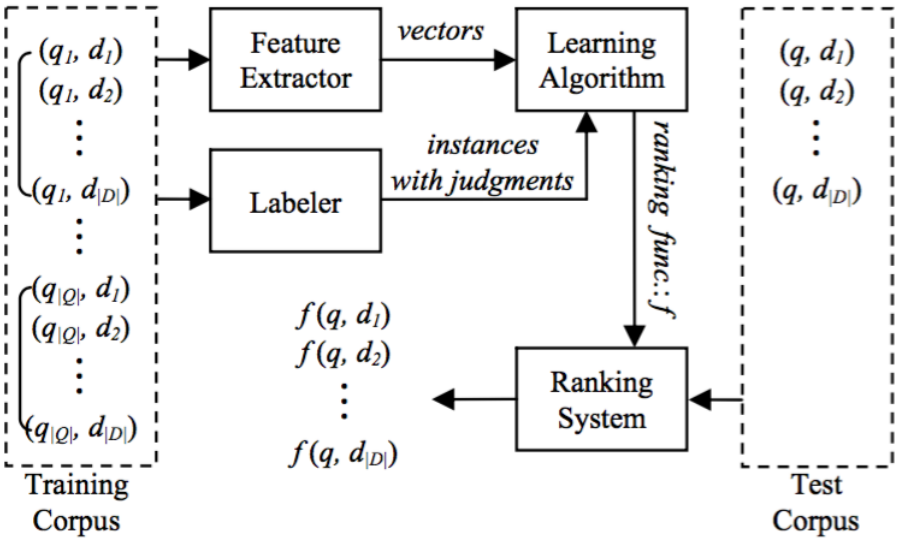

Figure 2.2: Typical flow of a discriminative learning system

Learning to rank models captured much attention in the latest years due to their simplicity and effectiveness in training a ranking function from labeled dataset. The learning process of these methods can be described in terms of the general discriminative learning framework [95], composed by four key elements: the input space $\mathcal{X}$, the output space $\mathcal{Y}$, the hypothesis space $\mathcal{H}$ and the loss function $\mathcal{L}$. The final goal of such methods is to train a ranking function f_{LTR} described as follows:

$$f_{LTR}(d, q) : \mathcal{X} \rightarrow \mathcal{Y}$$

The input space $\mathcal{X}$ contains the evidences of the objects under investigation, described in terms of feature vector x. To this end, it is common to design a function Φ, namely a feature extractor, that takes in input a query q and a document d and provides in output the feature vector $x = \Phi(d, q)$. The output space $\mathcal{Y}$ contains the target variable to learn. Depending from the ML task, the output space could be the space of real numbers $\mathbb{R}$ (e.g., for regression) or a set K of discrete values $\{0, 1, ..., K-1\}$ (e.g., for classification). The hypothesis space $\mathcal{H}$ denotes the class of functions that map the input space $\mathcal{X}$ to the output space $\mathcal{Y}$. Lastly, the loss function $\mathcal{L}$ represent probably one of the most important aspect in training a machine-learnt model, since it provides the information about how good is the predicted score with respect to the labeled instance in the ground truth. Widely adopted loss functions are the logistic loss, the exponential loss or the hinge loss.

The typical flow of a discriminative learning system is depicted in Figure 2.2. The supervised nature of the learning process requires a training corpus, composed by a

set of $|Q|$ queries $\{q_1, ..., q_{|Q|}\}$ and $|D|$ documents $\{d_1, ..., d_{|D|}\}$, where each training instance is represented by a query-document pair, i.e., $(q_i, d_j) \in Q \times D$. This training instance is represented by the feature vector $x_j^{(i)}$, extracted with the help of the feature extractor Φ, and is labeled with a ground truth label by expert annotators that provide the relevance judgment indicating the relationship between q_i and d_j. The features extraction and the relevance judgment tasks are usually done once and then saved on file, thus allowing the reproducibility of the experiments and the possibility to share the training dataset with other researchers. The learning algorithm then, is responsible to learn the ranking function f, according to the training corpus, trying to predict as accurately as possible the ground truth label, according to the loss function $\mathcal{L}$. Finally, the evaluation of a machine-learnt model follow the same schema: a different set of queries, sharing the same feature space with the training corpus, is used to test the ranking predicted by f with the ranking provided by the test corpus, according to the labeled relevance of its samples. Several metrics are used to this end, as we will see in Section 2.3.2.

2.2.2.1 Approaches to LtR

Most of the work done in LtR stems from machine learning area. The seminal work from which this area has stemmed is the research by Herbrich *et al.* [76] where authors present a new formulation of a problem consisting in learning a ranking model by minimizing a loss function that is acting on pairs of elements rather than single labeled training instances as it is common in regression and classification tasks. Since that seminal work has been published there has been a huge amount of works published with the goal of improving the quality of the generated models. An excellent survey on models and techniques specifically tailored on Information Retrieval (and Search in particular) is that of Tie-Yan Liu [95]. In that survey, most of the most important results aimed at improving the results of a search system using machine learning techniques has been summarized and organized. In particular, depending from their choice to adopt a particular instantiation of the four pillar of the discriminative learning, we can categorize these algorithms in three main classes: Pointwise, Pairwise and Listwise algorithms.

Pointwise approach. This is probably the simplest approach tackling the learning to rank problem, and the idea is to re-use existing machine learning methods by predicting the exact relevance score of each training sample, although this may not be necessary to reproduce the ranked list of documents according to the training corpus. Thus, the input space consists of the feature vector of each query-document pair, the

output space of its relevance score, the hypothesis space contains functions that maps these feature vectors in input to the relevance scores in output, and the loss functions measure how accurate is the prediction of each training sample, independently from others samples related to the the same query (i.e., this approach does not consider the ranked list as a whole but only the predicted score of the single training samples). The final ranking is achieved by simply sorting the documents by predicted scores.

Since the simplicity of this approach, all the standard regression and classification algorithms can be used for pointwise learning to rank [65, 18, 90, 50]. However, this approach is suboptimal because it does not consider the relative order between the documents, thus being unable to accurately learn the relative ranking, and for the loss function adopted, which have the visibility only of the single training sample and not of the full list of documents, on a query-basis.

Pairwise approach. This approach tries to go beyond one of the main limitations of the pointwise approach, represented by the relative ordering between the documents. Indeed, it focus on correctly classifying the relative order between a pair of documents, i.e., on predicting whether the first document is more relevant than the second or not. Thus, the goal is to minimize the number of miss-classified document pairs.

In such a situation, the input space consists of the feature vectors of the two documents in the pair, the output space by their relative ordering, assuming values in $\{-1, 1\}$, the hypothesis space contains bivariate functions mapping the two document vectors in input to their relative order in output, and the loss function is represented by the classification loss. For simplicity, the hypothesis h can be defined with a scoring function f such that $h(x_i, x_j) = 2 \cdot I_{\{f(x_i)>f(x_j)\}} - 1$, where $I_{\{A\}}$ assume a value of 1 if A holds, 0 otherwise.

Several algorithms follow the pairwise approach [22, 27, 149, 63]. However, still exists the limitation about the total ordering of the ranked list, i.e., being able to accurately rank two documents does not mean the model is able to reproduce properly the full ranked list.

Listwise approach. To overcome this limitation, the listwise approach is designed to work on the full list of documents at once. Thus, the input space consists of the full set of candidate documents, represented by their feature vectors. The output space consists either of the relevance scores of the documents or the ranked list of documents (i.e., a permutation). The hypothesis space contains functions mapping the feature vectors in input to their relevance degrees or their permutation in output. Depending from the choice of the output space, we have two loss function: i) when

the output space is a permutation, it measures the difference between the predicted ranked list and the ground truth ranked list; ii) when the output space is given by the relevance scores, it adopts one of the widely used IR metrics.

Popular listwise algorithms are described in [28, 161, 167, 178].

Since the seminal work by Herbrich *et al.* [76] presenting a new formulation for learning a ranking model based on ordinal regression, another pillar in the field is represented by RankBoost [63], the first learning algorithms for ranking which is based on the boosting principles introduced several years before with AdaBoost [64]. Boosting algorithms are currently among the most popular and most successful algorithms for supervised learning to rank tasks. They adopt an iterative technique designed to construct a "strong" learner using only a training set and a "weak" learning algorithm. Each "weak" learner is intended to perform only slightly better than a random learner, while the "strong" learner has much lower probability of error. Hence, these algorithms "boost" the weak learning algorithm to achieve a stronger learner. To each training instance is assigned a weight, representing its relative importance. Training examples that were miss-predicted by the "weak" learner at the current iteration will be boosted (i.e., they will receive an higher weight) at the following iteration. The end result is a final "strong" learner that linearly combine together a set of "weak" rankers. In this dissertation we focus on ensemble-based ranking models where the weak learners are decision trees.

Differently from the approaches using ensemble of weak learners, many other solutions were proposed for learning a ranking model. Burges *et al.* proposed RankNet [22], a neural network designed to model ranking function, trained using gradient descent method. This work is particularly important for two main reasons: i) authors formalized the usage of a probabilistic cost function, based on pairs of training instances, to use as a loss in place of the standard ranking function, with the former being differentiable (ranking is not) and thus prone to be used with gradient descent methods; ii) RankNet evolved in λ-Rank [24] by introducing the concept of the lambda-scores (an approximation of the gradient local to two training instances) and finally in λ-MART [23], by combining together a gradient boosting regression tree (MART) model with the lambda-scores. Despite λ-MART has been presented in 2010, it is still one of the state-of-the-art list-wise LtR algorithm. Argawal *et al.* [1] proposed a method for learning a ranking function from data represented in the form of a (similarity) graph. Also Dekel *et al.* [57] proposed a similar approach, but differently from the previous work, an arc from A to B means that A is to be ranked

higher than B (pairwise relative ordering). Radlinski and Joachims [134] presented an algorithm for learning the ranking of web search results by exploiting click-through data. Finally, several works addressed the problem of focusing mostly on top-rated documents [49, 90, 146].

2.2.2.2 Benchmark Datasets for LtR

The rapid expansion in popularity of learning to rank techniques for IR highlighted the need for large corpora of annotated datasets specifically designed for the task. Indeed, while the application of LtR techniques involves the creation of ad-hoc datasets with task-specific features, the proposal of new LtR models demands for a shared experimental platform to make meaningful comparison among different solutions in terms of common evaluation methodologies (e.g., fixed metrics). The main ingredients of such experimental platform are reliable corpora of documents, selected queries for training and test, human assessed relevance labels, and feature vectors extracted for each query/document pair.

In 2007 Microsoft tackled the problem, building a popular benchmark collection for LEarning TO Rank (LETOR)[13]. This collection is based on multiple, widely adopted, data corpora and query sets. The documents were sampled, feature vectors for query-documents pairs were extracted, and finally the dataset split in 5 folds according to standard IR techniques. The performances of popular LtR algorithms were also provided, in order to facilitate the comparison with state-of-the-art techniques. The framework has been actively maintained and updated several times, providing additional material for the research community in such a way to help researcher go beyond state-of-the-art LtR methods.

The latest LETOR update is dated 2010 and currently the collection is made up of the following datasets:

- Two Microsoft datasets, namely MSLR-WEB30k and MSLR-WEB10k, composed respectively by $30k$ and $10k$ queries, with the second that represents a random sampling of the first. Each document-query pair is represented with a 136-dimensional feature vector. These datasets have an average number of documents per query of ≈ 120, and were released on June 16, 2010.

- Two datasets, namely *Set 1* and *Set 2*, released by Yahoo! Labs in the context of the Learning to Rank challenge they organized in March 2010. The *Set 1*

[13] http://research.microsoft.com/en-us/um/beijing/projects/letor/

dataset, which is also the biggest one, contains about $30k$ queries and $710k$ query-documents pairs (on average ≈ 24 documents per query), and the documents are described by 519 features. The *Set 2* dataset contains $\approx 6k$ queries and $173k$ query-document pairs (≈ 27 document per query on average), with 596 features per document.

- Eight datasets released as part of the *LETOR 4.0* version, release in June 2009 and covering different methodology of the learning to rank problem (i.e., supervised ranking, semi-supervised ranking, rank aggregation and listwise ranking). These datasets are built using the *Gov2* web page collection ($25M$ pages) and two query set, namely *MQ2007* and *MWQ2008*, released by the Million Query (1MQ) Track in 2007 and 2008. The first query set contains 1700 queries, with the second 800.

- Several datasets, released as part of the *LETOR 3.0* version, which in turn represent updated versions of the *1.0* and *2.0* versions. This version is based on the *Gov* web page collection ($1M$ pages) and on the *OHSUMED* corpus [131], using respectively query sets from the *TREC Web Track* [51, 44] and from *OHSUMED* itself. These datasets are quite small (less than $1k$ queries) and are rarely used for LtR techniques, in favor of bigger datasets.

In 2016, the Tiscali Istella *WSE* released an additional learning to rank dataset to the public[14]. To date, this is the largest public LETOR dataset available, particularly useful for large-scale experiments on the efficiency and scalability of LETOR solutions. This dataset have been released in two different versions: i) a bigger version, namely *Istella LETOR* [55], composed of $33k$ queries and 220 features for each of the $20M$ query-document pairs, thus having 316 documents per query on average; ii) a smaller one, namely *Istella-S LETOR* [98], which represents a sampling of the irrelevant pairs with respect to the previous version, so as to reach an average of 103 documents per query on average. The number of queries and features of the second version are thus the same of the full dataset.

The relevance judgments of all the datasets except LETOR 3.0 range between 0 (irrelevant) to 4 (perfectly relevant), thus allowing the training of a fine grained ranking function. These datasets are usually exported in the SVM^{light} format, as shown in Figure 2.3, a matrix style where each row is a query-document pair, the first column identify the relevance judgment, the second the query-id and the others describe the features.

[14] http://blog.istella.it/istella-learning-to-rank-dataset/

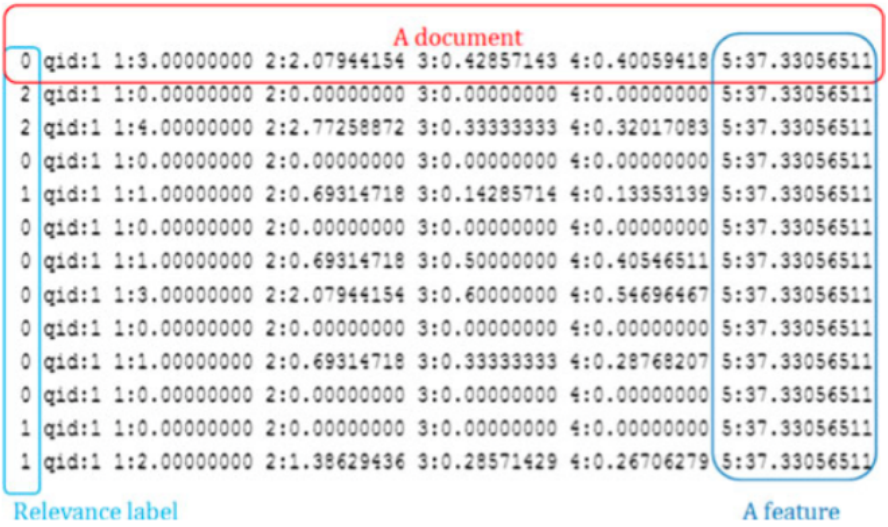

Figure 2.3: SVM^{light} format of the LETOR datasets

2.3 Retrieval Evaluation

Evaluate an Information Retrieval system means measuring its performance in answering user information needs. This represent a crucial activity, since we need objective evaluation prior to put a new IR system online. Performance can be measured both in terms of effectiveness, i.e., relevance of the retrieved documents in answering the user information need, and in terms of efficiency, i.e., speed and resource usage of the system in producing the results. While the latter is simpler to measure (e.g., average time spent on answering queries, memory required to execute the candidate retrieval and the ranking phases, etc.), the former is quite subtle. Indeed, user information need is an abstract concept, which is translated into something more concrete by the user writing the query. Moreover, the relevance of a document to a query is a subjective judgment, i.e., distinct users could consider the same result differently, either in terms of document relevance or ranking. However, this aspect can be mitigated by computing average metrics above a collections of queries and interpreting the result as a correlation with the preferences of a population of users. In the remainder of this Section, we will describe the typical evaluation process of an IR system. In Section 2.3.1 we present the methodologies adopted to evaluate such a system, while Section 2.3.2 describe the most popular metrics adopted to objectively measure IR effectiveness.

2.3.1 Evaluation Methodologies

One of the most critical aspect in performing the evaluation of an IR system is in the choice of the reference collection to use for measuring the metrics. Such a reference collection can be defined as the collection composed by a set of documents D, a set of queries Q, and a binary or multi-graded relevance score associated with each pair (d_i, q_j), where $d_i \in D$ and $q_j \in Q$. The relevance score is positive for documents relevant with respect to the query, 0 otherwise. Such collections are of fundamental importance, since they allow a clear comparison among different IR solutions. Moreover, evaluations can be done quickly and the results achieved are reproducible.

Several relevance assignment (i.e., the task of assigning a relevance score to each pair document-query) solutions have been proposed in literature [148]. The most popular is the one that follow the Cranfield paradigm proposed in the 1967 by Cleverdon [46], where expert judges are asked to provide relevance assessment to each document-query pairs. This approach is somehow infeasible for large collections, since it would be impossible to evaluate all the documents for each query (the number of documents could be millions or even worse billions). However, it has been profitably adopted by the Text REtrieval Conference (TREC) [173, 171, 121], one of the major conference in information retrieval, designed to encourage reproducible research on large text collections. TREC adopts the so called *pooling method* [156, 84] to mitigate the above mentioned problem: in place of evaluating all the documents for each query, they assume that most of the relevant documents can be found in the top-ranked k documents from each of n independent retrieval systems. With n and k large enough, the set of judged document can be assumed to be representative of the full documents collection. The *pooling method* is based on the assumption that an IR system rank documents according to their expected probability of relevance, following the probability ranking principle [141].

An alternative solution to the relevance assignment problem is represented by the usage of a crowd-sourcing platform to gather relevance scores at a larger scale [4]. This solution scales better than the Cranfield-derived paradigm, because of the potentially large number of "workers" that perform the annotations, and is usually cheaper, since it involves normal people in place of expert judges (which need also to be trained for the task). However, several considerations arise[33]: i) the design of the question to present to the user is of fundamental importance; ii) the interface plays a central role; iii) workers are not IR expert. Consequently the quality of the results has to be controlled by using some quality checking technique, i.e., by occasionally using

questions previously assessed by expert annotators checking whether the users answer them accordingly.

A completely different evaluation methodology, not based on relevance assignment, is the one exploiting users feedback on the relevance of the documents ranked in answer to a query. It relies on the implicit feedback provided by the user when interacting with the IR system. In this category falls the evaluations done using click-through data [83, 135] (i.e., how many clicks a document collects when it is returned by the IR system) or the time spent by a user on a given page, after having clicked on the link in the result list (dwell time) [94, 2].

2.3.2 Evaluation Metrics

Several metrics have been proposed to evaluate the effectiveness of an IR system over a benchmark test collection [148]. Probably the most adopted ones are precision and recall [85], described for the first time together by Kent *et al.* back in 1955. In order to give a formal definition of these and other metrics, let us introduce some formalism. Consider a query $q \in Q$, with Q the query set of the test collection, a set Rel_q of relevant documents for q, and a set Ret_q of documents retrieved by an IR system $\mathcal{S}$ in answer to the query q. Let $Ret_q^{(k)}$ be the set of top k documents retrieved, according to the ranking produced by $\mathcal{S}$ and to the cut off k.

Then, precision is defined as the fraction of retrieved documents that are relevant, while recall is defined as the the fraction of relevant documents retrieved. Such definitions assume that all the documents retrieved by the IR system are examined by the user. However, only top k documents are usually seen by user, since the presentation of the retrieved documents is split in several pages, according to the ranking provided the the system $\mathcal{S}$. Accordingly, revised definitions of precision and recall that take into accounts only top k retrieved documents are defined as follows:

$$P@k(q) = \frac{|Ret_q^{(k)} \cap Rel_q|}{|Ret_q^{(k)}|} \qquad \text{and} \qquad R@k(q) = \frac{|Ret_q^{(k)} \cap Rel_q|}{|Rel_q|}$$

Since a benchmark test collection is usually composed by several distinct queries, and the definition of precision and recall is for single queries, arise the need to aggregate these values among all the test collection. The commonly adopted aggregation approach is to average precision and recall for all the queries in Q:

$$P@k = \sum_{q \in Q} \frac{P@k(q)}{|Q|} \qquad \text{and} \qquad R@k = \sum_{q \in Q} \frac{R@k(q)}{|Q|}$$

Precision and recall have often an inverse relationship [75, 47], and it is common the need to find a good trade-off between the two values [21, 71] (i.e., one can select 100% of relevant documents, with a recall value of 1, at the cost of a very low precision, or the converse, improve the precision lowering the recall). This trade-off can usually be plotted in the so called precision-recall curve. Sometimes, however, does not make as much sense to find this trade-off, e.g., consider the first phase of a Web search engine, where the goal is to identify as many as possible of the relevant documents, thus maximizing recall.

When a single value which summarize both precision and recall is needed, it is usually adopted the so called *F-measure*, a variant of the *E-measure* proposed by Van Rijsgerben [168], where $F = 1 - E$. The *F-measure* is defined as follows:

$$F_\alpha@k = \frac{1}{\alpha\frac{1}{P@k} + (1-\alpha)\frac{1}{R@k}}$$

where α is a factor to weight differently precision and recall. Equally weighting the two terms is quite common and is obtained with a value of $\alpha = 0.5$. The resulting *F-measure* simplifies to:

$$F_1@k = \frac{2 \cdot P@k \cdot R@k}{P@k + R@k}$$

which represents the harmonic mean of precision and recall, and is named F_1.

The evaluation of an IR system using set-based metrics (like precision, recall and *F-measure*) is limiting, since these metrics do not take into account the ranking of the documents above the cut off k. Consider for example two systems retrieving the same amount of relevant documents in the top k results, but one placing the relevant documents at the top and the other at the bottom of this list. Their precision and recall values will be exactly the same, disregarding to where the relevant documents are placed.

Thus, one of the first metrics proposed to consider the order in which the returned documents are presented is the Mean Average Precision (MAP) [181], defined as follows:

$$AP@k(q) = \frac{\sum_{i=1}^{k} P@k(q) \cdot rel_q(i)}{|Rel_q|} \qquad \text{and} \qquad MAP@k = \sum_{q \in Q} \frac{AP@k(q)}{|Q|}$$

where $rel_q(i)$ denotes whether or not the i-th document in the ranking is relevant for the query q. Another popular metrics is the Mean Reciprocal Rank (MRR) [172], which is based on the assumption that for some tasks (e.g., question answering)

more than the full ranking it is important the position of the first relevant document returned. To this end, MRR is defined as follows:

$$MRR = \sum_{q \in Q} \frac{1}{rank_q}$$

where $rank_q$ is defined as the rank position of the first relevant document for the query q.

While binary assessed datasets are popular in IR, and the above presented metrics work well with these datasets, in some cases the relevance of a document to a query is not either positive or negative but range in different level of usefulness. Consider for instance the Web search engine scenario, where the documents are not of equal relevance to the query of the user. In such situations, it is fundamental to rank documents by their relevance, i.e., placing at the top of the result list highly relevant documents, before mildly and partially relevant ones. Accordingly, arise the need to develop novel metrics based on graded relevance judgments. To this end, Järvelin and Kekäläinen [81] proposed the Discounted Cumulative Gain (DCG) metrics, and its normalized version NDCG. These metrics adopt a log-based discount factor which progressively reduces the document contribution as its rank increase. The overall idea behind these metric is that the greater the ranked position of a relevant document, the less valuable it is for the user, since the probability that the user will never examine the document increase. The metrics are defined as follows:

$$DCG(q,k) = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)} \qquad \text{and} \qquad NDCG(q,k) = \frac{DCG(q,k)}{IDCG(q,k)}$$

where i denotes the position of the document in the ranked list, rel_i the graded relevance of the document for query q, and IDCG the ideal DCG obtained on the sorted list of documents by relevance. As usual, the NDCG for an entire test collection is obtained by averaging the NDCG of all the queries q in Q.

Chapter 3

Learning Relatedness Measures for Entity Linking

Entity Linking is the task of detecting, in text documents, relevant mentions to entities of a given knowledge base. To this end, entity-linking algorithms use several signals and features extracted from the input text or from the knowledge base. The most important of such features is *entity relatedness*. Indeed, we argue that these algorithms benefit from maximizing the relatedness among the relevant entities selected for annotation, since this minimizes errors in disambiguating entity-linking.

The definition of an effective relatedness function is thus a crucial point in any entity-linking algorithm. In this chapter we address the problem of learning high-quality entity relatedness functions. First, we formalize the problem of learning entity relatedness as a learning-to-rank problem. We propose a methodology to create reference datasets on the basis of manually annotated data. We then show that our machine-learnt entity relatedness function performs better than other relatedness functions previously proposed, and, more importantly, improves the overall performance of different state-of-the-art entity-linking algorithms. Finally, we present Dexter, an open-source Entity Linking framework implementing various entity linking strategies and developed as a product of the research activity for supporting the claim of this chapter.

3.1 Introduction

A typical entity linking system, as introduced in Section 2.1.4, performs the semantic enrichment of the text in three main steps: *spotting*, *disambiguation* and *filtering*. The *spotting* process identifies a set of candidate spots in the input document, and produces a list of candidate entities for each spot. The *disambiguation* process selects, for each spot, the most likely entity among the candidate list. The *filtering* process

discards annotations that are considered not correct or consistent with the overall interpretation of the document.

Let us introduce a simple example to describe how the entity linking process works:

> *On July 20, 1969, the Apollo 11 astronauts - Neil Armstrong, Michael Collins, and Edwin "Buzz" Aldrin Jr. - realized President Kennedy's dream.*

The text "President Kennedy" can be easily spotted and linked to *John F. Kennedy*, since in Wikipedia there are 98 anchors exactly matching such fragment of text and linking to the U.S. president page. In addition, the text "Apollo 11" may refer to two distinct candidates: the famous spaceflight mission, or the 1996 film directed by Norberto Barba. Similarly, the text "Michael Collins" may refer to either the well known astronaut, or to the Irish leader and president of the Irish provisional government in 1922. Indeed, mentions to the latter (408) are much more frequent than those to the former (141)[1].

The above spots and the relative candidate entities are further processed during the *disambiguation* step. The goal of disambiguation is to select the best candidate entity for each spot. This is usually done by considering the context of close mentions and by maximizing some measure of *relatedness* among the linked entities [52, 61, 77, 116, 152]. In our example, the astronaut "Michael Collins" and the "Apollo 11" spaceflight mission entities are preferred since they are clearly strongly related to each other and to the other entities found in the document, i.e., Buzz Aldrin and John F. Kennedy.

Lastly, the already disambiguated spot (i.e., each spot is associated with a single specific meaning) are processed during the *filtering* step, that is responsible for selecting only relevant annotations. For instance, the word "the" may refer to the entity associated with the definite article, but this linking might be relevant only for documents discussing the English grammar.

The effectiveness of the *entity relatedness* function adopted is thus a key-point for the accuracy of any entity-linking algorithm. In this chapter we propose a machine learning approach to devise a high-quality entity relatedness function. The main contributions can be summarized as follows:

- a formalization of the problem of devising entity relatedness functions as a learning-to-rank problem;

[1] Throughout this work, we used the 04/03/2013 dump, available at http://dumps.wikimedia.org/enwiki/20130403/enwiki-20130403-pages-articles.xml.bz2

- a novel technique to build benchmark datasets for learning and testing entity relatedness functions;
- an extensive experimentation showing that our automatically machine-learnt function outperforms state-of-the-art relatedness functions. More importantly, our approach can improve the performance of a whole class of entity-linking algorithms;
- an open source publicly available framework for addressing the entity linking problem and evaluating new algorithms in a fair test environment.

This chapter is organized as follows. In Section 3.2 we formalize the problem of learning automatically a entity relatedness function. In Section 3.3 we discuss related works, and how entity relatedness functions are used in these works. In Section 3.4 we evaluate some machine machine-learnt entity relatedness functions, and in Section 3.5 we evaluate their impact on entity linking algorithms. In Section 3.6 we describe DEXTER, an Entity Linking framework providing the implementation of some popular Entity Linking algorithms as well as the tools needed to develop new annotation techniques. Finally, Section 3.7 provide a summary of the chapter.

3.2 Entity Relatedness Discovery

Given a set of known entities $\mathcal{E}$ from a knowledge base $\mathcal{KB}$, and an unstructured text document D, entity linking aims at identifying all the relevant mentions in D to the entities of $\mathcal{E}$. The entity linking process involves three steps that we are going to detail in the following.

Spotting and Candidate Selection. Spotting aims at identifying spots, i.e., contiguous sequences of n terms (*n-grams*) occurring in D that might mention some entity $e \in \mathcal{E}$. A common method to identify the spots $S_D = \{s_1, s_2, \ldots\}$ is to exploit a *controlled vocabulary* of spots $\mathcal{L}$, and to search the input document for the n-grams that exactly match an entry of this vocabulary.

When Wikipedia is used as $\mathcal{KB}$, each Wikipedia article identifies an entity, and the vocabulary $\mathcal{L}$ can be easily built by considering the article *titles* along with the *anchor texts* of all internal Wikipedia hyperlinks.

Each spot $s_i \in S_D$ is then associated with a set of candidate entities $C(s_i) \subseteq \mathcal{E}$. This is done by considering all the entities of $\mathcal{E}$ that are referred to in $\mathcal{KB}$ by using spot s_i as an anchor text. Unfortunately the same spot s_i can occur in different places of

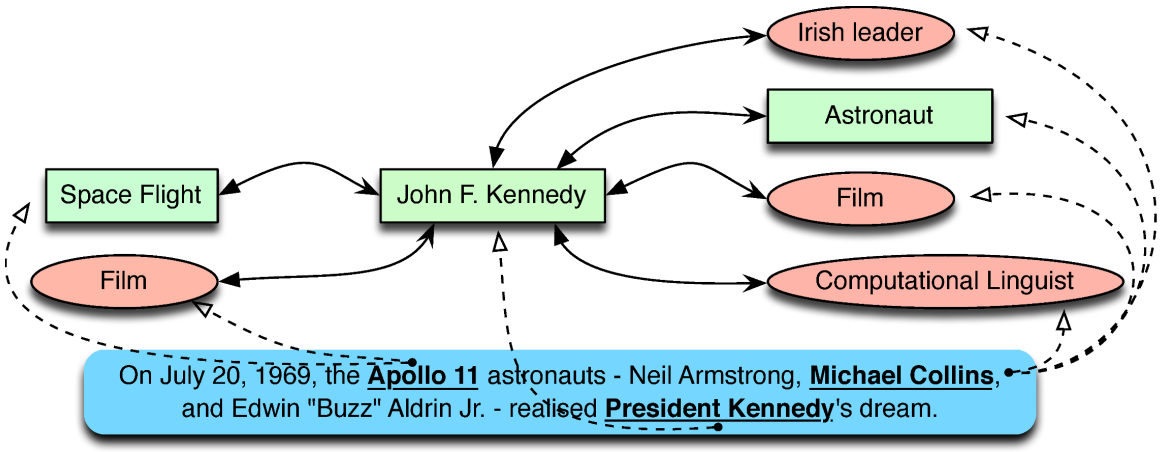

Figure 3.1: Entity relationships for spotting and disambiguation. Three spots extracted from the above example are underlined: $s_1 =$ "*Apollo* 11", $s_2 =$ "*President Kennedy*", and $s_3 =$ "*Michael Collins*". The graph shows relatedness edges connecting candidates entities. For simplicity of representation the candidates for spot s_2 are omitted. Rectangles are used to indicate correctly disambiguated entities, while ellipses refer to other candidate entities.

$\mathcal{KB}$ (and even of D!) and refer to distinct entities. Finally, we denote by $\epsilon(s_i) \in C(s_i)$ the entity that is actually mentioned by s_i in D.

Figure 3.1 illustrates the three spots $\{s_1, s_2, s_3\}$ (among others) detected in our text example. For each s_i, the outgoing dashed directed edges identify the set of candidate entities $C(s_i)$, where $\epsilon(s_i) \in C(s_i)$, i.e., the entity that is actually mentioned by s_i, is represented as a rectangle.

To limit the set of spots and candidate entities to the most meaningful ones, *link probability* and *commonness* properties can be usefully exploited [111]. The link probability for a spot s_i is defined as the number of times s_i occurs as a mention in $\mathcal{KB}$, divided by its total number of occurrences. This permits to discard spots that are rarely used as a mention to a relevant entity. For example the spot "July 20", introduced in the example of Section 3.1, occurs hundreds of times in Wikipedia, and, even if it is the title of an article, only in a few cases it used as anchor text.

The *commonness* of a candidate $c \in C(s_i)$ for spot s_i is instead defined as the fraction between the number of occurrences of s_i in $\mathcal{KB}$ actually referring to c, and the total number of occurrences of s_i in $\mathcal{KB}$ as a mention to an entity. For example, the spot "Michael Collins" may refer to more than 20 different entities, but the Irish revolutionary leader (421 mentions, commonness 0.5), the film about his life (126 mentions, commonness 0.15) and the astronaut (132 mentions, commonness 0.15) are largely the most common.

Setting a threshold on minimum linking probability and minimum commonness has been proven to be a simple and effective strategy to limit the number of spots and associated candidates, without harming the recall of the entity linking process [116].

Disambiguation and Linking. Since in many cases we have several candidates for a single spot s_i (i.e., $|C(s_i)| > 1$), the spot has to be disambiguated by choosing the right entity $\epsilon(s_i)$ among the candidates $C(s_i)$. For each spot, a disambiguation algorithm outputs the selected entity and a confidence score.

In order to choose the best entity for a spot, disambiguation may exploit different signals and features. These include commonness and linking probability, and many others features considering the text surrounding the spot, and the other spots of the document. The most important of such features is *entity relatedness*, usually defined as a real function $\rho : \mathcal{E} \times \mathcal{E} \to [0, 1]$, where 0 and 1 are the minimum and maximum relatedness measure, respectively. To guarantee the accuracy of entity linking, the entities selected by the disambiguation process to be linked to the detected spots have in fact to be strongly related to each other.

Figure 3.1 shows the relatedness graph referred to our example. The selection of the best entities is often implemented on top of this relatedness graph, where edges are weighted by some entity relatedness function. Therefore, the role of such entity relatedness function is crucial for the accuracy of the disambiguation process.

Even if the definition of a entity relatedness function is not a trivial task, several works agree on the effectiveness of the Wikipedia-based relatedness function proposed by Milne and Witten [116, 115]. The relatedness between two entities a and b is in this case computed by exploiting the graph structure of Wikipedia:

$$\rho^{\mathsf{MW}}(a,b) = 1 - \frac{\log(\max(|in(a)|,|in(b)|)) - \log(|in(a) \cap in(b)|)}{\log(|W|) - \log(\min(|in(a)|,|in(b)|))}$$

where W is the set of all Wikipedia entities, while $in(a)$ and $in(b)$ are the sets of Wikipedia articles linking to a and b, respectively. When $|in(a) \cap in(b)| = 0$, we have $\rho^{\mathsf{MW}}(a,b) = 0$. In addition, ρ^{MW} is maximum (equal to 1) when $in(a) \cap in(b) = in(a) = in(b)$, and thus all the articles that cite a also cite b, and vice versa.

The ρ^{MW} function, promoting entities that are co-cited by the same Wikipedia articles, is considered the *state-of-the-art* relatedness measure, adopted also in [61, 73, 78]. On the other hand, there is no guarantee that ρ^{MW} would produce a proper scoring of the candidate entities.

Example 3.2.1 *Given the entities* a="Andronicus of Rhodes", b="Chondrichthyes", *and* c="Aristotle" *occurring in a document, we have that* $\rho^{\mathsf{MW}}(a, b) = 0.54$ *and*

$\rho^{\mathsf{MW}}(a, c) = 0.56$[2]. *The connection between entities a and c is very strong since Andronicus of Rhodes is credited with the production of the first reliable edition of Aristotle's works. The (unexpected) high relatedness score between entities a and b is instead due to a single co-citing Wikipedia article (which is c) that reports about Aristotle's studies of a group of fishes he named selachians, a.k.a. chondrichthyes. Therefore, in this case a single co-citation is enough to produce an unexpected high value* $\rho^{\mathsf{MW}}(a, b)$*, which is similar to the expected large value of* $\rho^{\mathsf{MW}}(a, c)$.

Another interesting observation is that ρ^{MW} *is symmetric: Andronicus of Rhodes is relevant to Aristotle, to the same degree Aristotle is relevant to Andronicus of Rhodes.*

Filtering. Among all the annotated spots identified by the previous two steps, an *Entity Linking* system has to selects only those relevant for the document, i.e., that provides important facets or in general improve the comprehension of the document. A common approach to filter irrelevant entities is to resort to the confidence score produced by the disambiguation step. Indeed, this confidence score summarize the relatedness between each entity and all the others, thus allowing its usage to select the most likely matching entities, and to trade precision with recall. state-of-the-art *Entity Linking* systems adopts a mixing approach, where a threshold is set on the confidence score, the commonness and the link probability. The rationale behind this approach is to avoid annotating rarely linked entities or highly improbable meanings. Being dependent from the confidence score, also the *filtering* step is affected by the entity relatedness function.

Our claim is that a good entity relatedness function ρ can improve the performance of a large class of entity linking algorithms. In Section 3.3 we will discuss related works and show the crucial role of entity relatedness functions in many proposals. Here, we propose a set of properties that an optimal entity relatedness measure should satisfy, and we formalize the problem of discovering a good entity relatedness function into a learning-to-rank problem.

Relatedness as a Ranking Function. Suppose that an entity linking algorithm identifies only two spots s_h and s_i for a document D, and for these spots it generates the two sets of candidate entities $C(s_h)$ and $C(s_i)$ respectively. Most disambiguation algorithms assume that if one of the candidate entities in $C(s_h)$ is highly related to another entity in $C(s_i)$, then it is very likely that they are the entities $\epsilon(s_h)$ and $\epsilon(s_i)$ actually mentioned by the two spots.

[2]The values to compute the two ρ^{MW} measures are: $|in(a)| = 24$, $|in(b)| = 261$, $|in(c)| = 3502$, $|in(a) \cap in(b)| = 1$, $|in(a) \cap in(c)| = 17$, and $|W| = 4,255,306$.

We claim that a good entity relatedness function ρ should promote the relatedness of correct entities: given entity $\epsilon(s_h)$, its relatedness with $\epsilon(s_i)$ should be larger then that with any other candidate in $C(s_i)$. This should hold for every spot $s_i \neq s_h$.

Proposition 3.2.1 *Given D, $S_D = \{s_1, s_2, \ldots\}$, and, for each spot s_i, $C(s_i)$ and $\epsilon(s_i)$, a relatedness function ρ improves entity-linking accuracy if the following constraint holds:*

$$\forall s_h \in S_D, \quad \forall s_i \in S_D \setminus \{s_h\}, \quad \forall c \in C(s_i) \setminus \{\epsilon(s_i)\} : \\ \rho(\epsilon(s_h), \epsilon(s_i)) \; > \; \rho(\epsilon(s_h), c). \tag{3.1}$$

Indeed, the constraint in Eq. 3.1 nicely fits into a learning-to-rank based formulation [82]. The relatedness function ρ can be in fact modeled as a ranking function, with entity $\epsilon(s_h)$ used as a query. According to the above Proposition, function ρ should score all the entities actually mentioned in the document, i.e. $\epsilon(s_i)$ for all $s_i \neq s_h$, higher than any other false-positive candidate, i.e. $c \in C(s_i) \setminus \{\epsilon(s_i)\}$) for all $s_i \neq s_h$.

Given document D and spots $S_D = \{s_1, s_2, \ldots, s_h, \ldots\}$, we denote by $\mathcal{R}_D^h = \cup_{i \neq h} C(s_i)$ the set of candidates to be ranked for query $\epsilon(s_h)$, and by $\mathcal{E}_D^h = \cup_{i \neq h} \epsilon(s_i)$ the set of relevant entities for the query, where $\mathcal{E}_D^h \subseteq \mathcal{R}_D^h$. From an information retrieval perspective, items in $\mathcal{R}_D^h$ are relevant for query $\epsilon(s_h)$ if and only if they belongs to $\mathcal{E}_D^h$. Note that we are considering the spots of D altogether, and therefore there are potentially many related entities to the query $\epsilon(s_h)$ that should be ranked high. Let us denote with π_ρ^h the score descending ordering of $\mathcal{R}_D^h$ induced by our ranking relatedness function ρ for query $\epsilon(s_h)$. According to Proposition 3.2.1, a scored list π_ρ^h is effective when entities in $\mathcal{E}_D^h$ are in the top positions of the list. We can thus measure the effectiveness of our ranking relatedness function by using common information retrieval quality metrics such as *NDCG* [81]. In our context we define $DCG(\pi_\rho^h)$ as:

$$DCG(\pi_\rho^h) = \sum_{j=1}^{|\pi_\rho^h|} \frac{[\![\pi_\rho^h[j] \in \mathcal{E}_D^h]\!]}{\log(j+1)}$$

where $\pi_\rho^h[j]$ denotes the j-th item of the scored list, and $[\![x]\!]$ equals 1 if x is true and 0 otherwise. *NDCG* is defined as the usual normalized version of *DCG*.

We can now introduce the *Entity Relatedness Discovery* problem.

Problem 3.2.1 (Entity Relatedness Discovery)
Let $\mathcal{D}$ be a collection of entity-linked documents, where for each document $D \in \mathcal{D}$ and every relevant spot s_i of S_D we know $\epsilon(s_i)$. Given the entity $\epsilon(s_h)$ and the set $\mathcal{R}_D^h$, a ranking relatedness function ρ induces an ordering π_ρ^h of $\mathcal{R}_D^h$.
The Entity Relatedness Discovery Problem *requires to find the function ρ that maximizes the ranking quality:*

$$\frac{1}{|\mathcal{D}|}\sum_{D\in\mathcal{D}}\frac{1}{|\mathcal{S}_\mathcal{D}|}\sum_{s_h\in S_D}\mathsf{NDCG}(\pi_\rho^h)$$

In our experiments, we chose to optimize *NDCG* to find a good entity relatedness function, but we used several other ranking quality functions to assess the goodness of results.

Unlike previous approaches, we do not suggest a specific novel entity relatedness function. Rather, we define a learning-to-rank framework to discover the optimal entity relatedness function.

3.3 Related Work

In the following we discuss how the notion of *entity relatedness* is exploited by state-of-the-art entity linking algorithms. Emphasis is given to the solutions proposed in [116] and [73] and [61] which are the most relevant proposals in the field, and they are all adopting ρ^{MW} as entity relatedness function. We show that the entity relatedness function defined in Proposition 3.2.1 can replace ρ^{MW} since it fits better the framework and the objectives of the above algorithms.

WikiMiner [116]. Given a document D, let us consider its spots S_D and for each spot s_i the associated set of candidates $C(s_i)$. Let us suppose that a subset of the spots are associated with only a single entity. We denote with $U \subseteq \mathcal{E}$ the *context*: the set of unambiguous entities linked to spots in S_D, i.e., $U = \bigcup_{|C(s_i)|=1} \epsilon(s_i)$. The WikiMiner algorithm exploits the entities in U as safe reference points to help the disambiguation of the other ambiguous spots for which $|C(s_i)| > 1$ holds. The idea is to select for every ambiguous spot of S_D the entity which is, on average, the most related with the "safe" entities in U. The relatedness function adopted is ρ^{MW}. It is worth noting that not all the entities in U have the same impact: an entity $u \in U$ is in fact considered of high quality if it is strongly related to the other entities in U, and if the *link probability* of the corresponding spot is high. These two criteria allow a weight w_u, $0 \leq w_u \leq 1$ to be assigned to each entity u in U. Note that the main aim of this weight is to reduce the impact of low-quality entities occurring in U. When

applied to our simple example, the low resulting weight would demote the importance of the safe entity "July 20".

Every candidate c in $C(s_i)$ is scored according to the following function:

$$score(c \mid U) = \frac{1}{\sum_u w_u} \sum_{u \in U} w_u \cdot \rho^{\mathsf{MW}}(u, c). \tag{3.2}$$

It is easy to show that the accuracy of the disambiguation would improve if we adopted, instead of ρ^{MW}, a relatedness function ρ that satisfies our Proposition 3.2.1.

Given an entity $u \in U$, we can rewrite Eq. 3.1 and derive as follows:

$$\begin{aligned} \rho(u, \epsilon(s_i)) &> \rho(u, c) \quad \Rightarrow \\ \frac{1}{\sum_u w_u} \sum_{u \in U} w_u \cdot \rho(u, \epsilon(s_i)) &> \frac{1}{\sum_u w_u} \sum_{u \in U} w_u \cdot \rho(u, c) \quad \Rightarrow \\ score(\epsilon(s_i) \mid U) &> score(c \mid U) \end{aligned}$$

Therefore, a relatedness function satisfying Proposition 3.2.1 would always correctly rank entity $\epsilon(s_i)$ higher than any other candidate for the corresponding spot s_i even when integrated in the WikiMiner framework.

Interestingly, the authors of [116] use machine learning to combine the above relatedness score with other two features: *commonness* and *context quality* (measured as $\sum w_u$). They experiment with a training set built from 500 Wikipedia articles. However, machine learning is not exploited to improve the relatedness function as in our proposal.

Referent Graph [73]. *Referent Graph*, a graph-based method still exploiting the relatedness function ρ^{MW}, is proposed in [73]. Let $RG(V, E)$ be a weighted directed graph where the nodes includes all the spots s_i of S_D and candidates $C(s_i)$. RG has a directed edge from s_i to every $c \in C(s_i)$, and reciprocal edges connecting every pair of candidate entities a and b, $a \in C(s_i), b \in C(s_h), i \neq h$. Spot-candidate edges (s_i, c) are weighted according to the cosine similarity between the Wikipedia article corresponding to entity c and a local context window of 50 words around the spot s_i. The candidate-candidate edges (a, b) are weighted by using ρ^{MW}. Finally, weights are normalized so that weights on outgoing edges from a given node always sum up to 1. The graph shown in Figure 3.1 is a toy referent graph, where relatedness edges connecting candidates entities for the spot s_1 with candidates entities for the spot s_3 are omitted for clarity.

The score of a candidate entity for a given spot is given by the steady state distribution of a random walk with restarts [124] in RG, where candidate nodes have

restart probability 0, and spot nodes have a restart probability proportional to their inverse document frequency score in the Wikipedia corpus. Also in this case, assigning a different restart probability to spot nodes, and weighting as above explained spot-candidate edges is aimed to limit the impact of non relevant or incorrectly matched mentions.

The rationale of the random walk approach is to evaluate the relationships among the whole set of candidates simultaneously, in contrast to previous methods where the scores of candidate entities are assigned independently of each other. Also in this algorithm the choice of the entity relatedness function ρ has a strong impact on the performance since it drives the random walk process. A set of entities being very related to each other is likely to produce a reinforcement loop, and eventually include the most probable states of the random walk.

Even if we do not provide a formal proof as for WikiMiner, it is clear that a good relatedness function should promote the reciprocal relatedness among the right entities in the graph, thus helping the random walk to converge to the correct ranking of candidates.

A similar approach is used in [179], where a slightly differently weighted referent graph is pruned progressively by removing iteratively the node with the lowest weighted degree (sum of the weights of incoming edges). Even in this case, the weights of candidate-candidate edges are computed with the ρ^{MW} relatedness function. The paper do not compare performances with those of [116], [73], or other algorithms, and it is thus difficult to estimate the impact of this proposal.

TAGME [61]. TAGME is an annotation framework focussing on efficiency that exploits two main features: *commonness* and the ρ^{MW} relatedness. First, candidate entities for a spot s_i are ranked according to their average relatedness with other candidate entities for spots $s_j \neq s_i$, weighted by their commonness. Then, from the top 30% candidates of the resulting ranked list, the entity with the largest commonness is finally selected.

Also this algorithm would benefit by a relatedness function satisfying Proposition 3.2.1, since it would help to boost the score of the actual entities mentioned in the document. However, the benefit is limited, since the relatedness function impacts more on the pruning irrelevant candidates, while the final choice of the best entity is mainly driven by the commonness feature.

Other approaches. Relatedness function ρ^{MW} is partially inspired by the so-called Normalized Google Distance (NGD) [43], which borrows from Kolmogorov complexity and information distance concepts. While NDG is tailored to measure

similarity between words or phrases, ρ^{MW} measure is specifically tailored to entities represented in a graph structure such as the one of Wikipedia. In [67] another Wikipedia-based relatedness measure, named ESA, is proposed. A word is represented in a high dimensional space by considering for each Wikipedia article the relevances of the word in the article, and by summing such score vectors for longer text fragments. Also [77] investigates new text-based relatedness measures that try to go beyond link-based similarities. The study conducted in [115] shows however that ESA has a performance similar to that of ρ_{Milne}, with the latter being much cheaper to be computed since it does not require to index the whole Wikipedia textual content. In [152], the authors improve only slightly the solution proposed in [52], but they do not provide any comparison with [116, 73].

The authors of [169] propose a machine learning approach to rank entity-based facets related to a given Web search query. Since the paper focuses on a special set of entities, such as monument and celebrities, the presented technique exploits information coming from image search queries and Flickr image tags. The goal of [169] is not to discover the degree of relatedness between entities, but rather to suggest entities that are most likely to generate a large click through.

Finally, several works [58, 61, 116] exploit machine learning techniques for entity linking, but in this chapter we use learning to rank for improving the *relatedness function*, which is important for improving quality of entity linking task as well in other tasks, such as entity ranking [19] or entity suggestion [35].

3.4 Entity relatedness evaluation

In the following we describe the methodology adopted to build a reference dataset for the learning process, the feature used to describe entities, and finally the performance of two automatically machine-learnt relatedness functions.

3.4.1 Building a benchmark dataset

In order to evaluate the impact of different relatedness functions, we built a benchmark dataset for Problem 3.2.1. This dataset, used to train and test our relatedness function, contains a set of tuples in the form $\langle \epsilon, \mathcal{R}_\epsilon, \mathcal{E}_\epsilon \rangle$, where ϵ is an entity occurring in a document D, $\mathcal{R}_\epsilon$ is a set of *candidate entities* possibly occurring in D, and $\mathcal{E}_\epsilon \subseteq \mathcal{R}_\epsilon$ are the *relevant entities* occurring in D besides ϵ.

In order to build these tuples, we need both positive and negative examples, i.e., positive ones from $\mathcal{E}_\epsilon$ and negative ones from $\mathcal{R}_\epsilon \setminus \mathcal{E}_\epsilon$. In most entity-annotated datasets,

each document is annotated by one or more human assessors, who manually performed some kind of spotting and entity disambiguation tasks. Therefore, for each document D we only have positive examples, i.e. the set $\mathcal{A}_D$ of entities actually occurring in D. In addition, we do not know the spot in D that actually mentions each entity in $\mathcal{A}_D$.

Hence, to generate our dataset for training our relatedness function, we have to devise a sort of reverse annotation process, aimed at discovering the spots associated with the known entities, and the potential candidates of such spots. In this way, we identify also the negative examples to build the tuples $\langle \epsilon, \mathcal{R}_\epsilon, \mathcal{E}_\epsilon \rangle$. In more detail, we generate our benchmark dataset as described below:

1. we set up a knowledge base $\mathcal{KB}$ of entities based on Wikipedia. This contains entities, their mentions, i.e, anchor text of incoming links and page title, and the hyper-link structure; we created a vocabulary of entity mentions $\mathcal{L}$ containing only spots with link probability larger than 2%. Finally, for each spot we disregarded entities with commonness smaller than 3%;
2. we generate all *n-grams* of every given document D, with $n \leq 6$, and we match them against $\mathcal{L}$ to devise the spots S_D;
3. for each spot s_i in S_D, we retrieve the candidate entities $C(s_i)$ as the set of entities linked in Wikipedia by the same n-gram;
4. we finally consider the set of relevant entities $\mathcal{A}_D$ of D, as annotated by the human assessors. Since we do not know the real association between each spot s_i and the human annotated entities in $\mathcal{A}_D$, for each spot s_i we look for the actual entity $\epsilon(s_i)$ in the set $C(s_i) \cap \mathcal{A}_D$. If $C(s_i) \cap \mathcal{A}_D = \emptyset$, we assume that $\epsilon(s_i)$ is not known, and thus discard s_i. If $|C(s_i) \cap \mathcal{A}_D| > 1$[3], we also throw away s_i, since we are not able to disambiguate. Finally, if $|C(s_i) \cap \mathcal{A}_D| = 1$, then $\epsilon(s_i) \in C(s_i) \cap \mathcal{A}_D$ is the actual entity to link to s_i.

At the end of the process, for each document D we have: a set of spots S_D and, for each spot s_i, a set of candidate entities $C(s_i)$ and also the mentioned entity $\epsilon(s_i)$.

Thus, for every spot s_h of every document D, we can generate a tuple $\langle \epsilon, \mathcal{R}_\epsilon, \mathcal{E}_\epsilon \rangle$ for the benchmark dataset that contains: (i) the actually mentioned entity $\epsilon(s_i)$, (ii) the set of candidate entities for every other spot in the document, and (iii) the set of correctly linked, and thus related, entities in the document. By assuming that close spots are more likely to be related, we did not consider in this tuple generation step those spots occurring at a distance larger than $\omega = 150$ characters from the current spot associated with entity ϵ.

[3]In our datasets, this happens in only 2% of the cases.

In our experiments we used a subset of the CoNLL 2003 entity recognition [78] task dataset, which includes annotated news stories of the Reuters Corpus V1. The dataset contains 1494 documents with an average length of 187 terms. Each document contains on average 11.7 entities.

We processed the corpus as explained above, and we thus built a dataset for evaluating the relatedness containing over 1.6 million tuples. We split the tuples in training, validation, and test set, respectively containing $977,514$, $369,798$ and $302,529$ records. Please observe that we take care of producing each dataset from a disjoint subset of documents in the collection, so that the tuples in the training and test sets were actually generated from a different subset of documents.

3.4.2 Features

A pair of entities a and b, for which the relatedness $\rho(a, b)$ has to be estimated, is represented by a set of 27 features shown in Table 3.1. The choice of such features is driven by the following considerations. First, we want to maximize their applicability by using publicly available data, and by using measures that can be easily applied to other entity knowledge bases, e.g., FreeBase.[4] For this reason we do not use click-through, access log, or query log based data, which are very difficult to obtain. We use instead several features related to the link structure of our knowledge base, such as the number of in-links $in(e)$ and out-links $out(e)$ of an entity e.

Second, there are applications of the entity relatedness function where the concept of spot is not applicable. Consider, for instance, the case of related entity recommendation where the query is a entity that is not associated with any spot. Therefore, we do not include features such as *link probability* and *commonness*.

Finally, we do not include text-based similarity measures, such as cosine similarity between Wikipedia articles pages, because this kind of approaches have been proven to perform similarly to the ρ^{MW} measure, but are much more computationally expensive [115].

Note that, by using the proposed machine learning approach, the feature set we adopt can be easily enriched with any additional feature, or by analyzing any other different knowledge base.

We categorize the features listed in Table 3.1 in three categories: *singleton*, *asymmetric* and *symmetric*.

[4] http://www.freebase.com/

Singleton Features	
P(a)	probability of a mention to entity a: $P(a) = \|\|in(a)\|/\|W\|$.
H(a)	entropy of a: $H(a) = -P(a)\log(P(a)) - (1-P(a))\log(1-P(a))$.
Asymmetric Features	
P(a—b)	conditional probability of the entity a given b: $P(a\|b) = \|in(a) \cap in(b)\| \ / \ \|in(b)\|$.
$\mathsf{Link}(a \to b)$	equals 1 if a links to b, and 0 otherwise.
$P(a \to b)$	probability that a links to b: equals $1/\|out(a)\|$ if a links to b, and 0 otherwise.
$\mathsf{Friend}(a, b)$	equals 1 if a links to b, and $\|out(a) \cap in(b)\|/\|out(a)\|$ otherwise.
$KL(a\|\|b)$	Kullback-Leibler divergence: $KL(a\|\|b) = \log \frac{P(a)}{P(b)} P(a) + \log \frac{1-P(a)}{1-P(b)}(1-P(a))$.
Symmetric Features	
$\rho^{\mathsf{MW}}(a, b)$	co-citatation based similarity [116].
$J(a, b)$	Jaccard similarity: $J(a,b) = \frac{in(a) \cap in(b)}{in(a) \cup in(b)}$.
$P(a, b)$	joint probability of entities a and b: $P(a,b) = P(a\|b) \cdot P(b) = P(b\|a) \cdot P(a)$.
$\mathsf{Link}(a \leftrightarrow b)$	equals 1 if a links to b and vice versa, 0 otherwise.
$\mathsf{AvgFr}(a, b)$	average friendship: $(\mathsf{Friend}(a,b) + \mathsf{Friend}(b,a))/2$.
$\rho^{\mathsf{MW}}_{\mathsf{out}}(a, b)$	ρ^{MW} considering outgoing links.
$\rho^{\mathsf{MW}}_{\mathsf{in\text{-}out}}(a, b)$	ρ^{MW} considering the union of the incoming and outgoing links.
$J_{\mathsf{out}}(a, b)$	Jaccard similarity considering the outgoing links.
$J_{\mathsf{in\text{-}out}}(a, b)$	Jaccard similarity considering the union of the incoming and outgoing links.
$\chi^2(a, b)$	χ^2 statistic: $\chi^2(a,b) = (\|in(b) \cap in(a)\| \cdot (\|W\| - \|in(b) \cup in(a)\|) + -\|in(b) \setminus in(a)\| \cdot \|in(a) \setminus in(b)\|)^2 \cdot \cdot \frac{\|W\|}{\|in(a)\| \cdot \|in(b)\| (\|W\| - \|in(a)\|)(\|W\| - \|in(b)\|)}$
$\chi^2_{\mathsf{out}}(a, b)$	χ^2 statistic considering the outgoing links.
$\chi^2_{\mathsf{in\text{-}out}}(a, b)$	χ^2 statistic considering the union of the incoming and outgoing links.
$\mathsf{PMI}(a, b)$	point-wise mutual information: $\log \frac{P(b\|a)}{P(b)} = \log \frac{P(a\|b)}{P(a)} = \log \frac{\|in(b) \cap in(a)\|\|W\|}{\|in(b)\|\|in(a)\|}$

Table 3.1: Features for entity relatedness learning.

Singleton features regard a single entity. They include only frequency and entropy, computed on the basis of the frequency of Wikipedia links to the entity article page. These features are computed for both entities of a given pair (a, b), resulting in four scores.

We claim that a relatedness function should not be symmetric. Consider for example the entities *Neil Armstrong* and *United States of America*: it seems reasonable that the relatedness of *United States of America* given *Neil Armstrong* is greater than the relatedness of *Neil Armstrong* given the *United States of America*. For this reason we included five asymmetric features, which are computed in both directions of the pair, resulting in ten scores.

Last, we considered 13 symmetric features, such as ρ^{MW}. Some of these features derive from asymmetric ones, and others are variations computed by considering outgoing links of an entity instead of incoming ones.

All the above features are computed on the basis of the same Wikipedia dump mentioned in the Section 3.1. Therefore, features are not extracted on the training or test dataset.

3.4.3 Quality of entity relatedness

To solve the Entity Relatedness Discovery problem, we used an existing tool for learning ranking functions, named RankLib.[5] This includes the implementation of several effective algorithms. We report the results of the two most effective: Gradient-Boosted Regression Trees [66] and LambdaMart [178]. We denote the models built with those algorithm ρ^{GDBT} and $\rho^{\lambda\mathsf{MART}}$.

Note that the two models differ significantly in the objective function being optimized. The $\rho^{\lambda\mathsf{MART}}$ model was built by a list-wise algorithm and minimizing *NDCG*@10. This is indeed in perfect agreement with our definition of entity relatedness problem, and with the benchmark created. On the other hand, the ρ^{GDBT} model optimizes the error in predicting the class label (i.e., relevant vs. not relevant) of a given instance. Therefore, the prediction can be used to produce a ranking, but the model does not optimize the ranking directly.

In Table 3.2 we report the performance of the two relatedness functions ρ^{GDBT} and $\rho^{\lambda\mathsf{MART}}$, and compare it against ρ^{MW}. The improvement of using a machine-learnt function that exploits 27 features is apparent with every ranking quality measure adopted. If we consider *NDCG*@10, $\rho^{\lambda\mathsf{MART}}$ improves over ρ^{MW} by a factor of 25%. The two machine-learnt functions have very similar performance, with no significant difference. Recalling to the Example 3.2.1, $\rho^{\mathsf{GDBT}}(a,b) = 0.0015$ while $\rho^{\mathsf{GDBT}}(a,c) = 0.66$: the value of ρ^{GDBT} (a,b) (*Andronicus of Rhodes* and *Chondrichthyes*) is low as we expected.

[5] http://people.cs.umass.edu/~vdang/ranklib.html

Features	NDCG@5	NDCG@10	P@1	P@5	P@10	MRR
ρ^{MW}	0.59	0.63	0.62	0.42	0.31	0.72
$\rho^{\lambda\mathsf{MART}}$	**0.75**	**0.79**	**0.80**	**0.51**	**0.36**	**0.87**
ρ^{GDBT}	**0.75**	0.78	**0.80**	**0.51**	0.35	0.86

Table 3.2: Entity ranking performance of machine-learnt relatedness functions.

In order to gain some insight on the machine-learnt functions, and on the role of the different features, we run a study based on a naïve feature selection algorithm [69]. This algorithm ranks features by leveraging their similarity and the score of single-features models. It promotes effective features and demotes features similar to any other already selected one. Our objective here is not to find the best performing subset of features, but rather to investigate the importance of ρ^{MW} compared with other features not considered by state-of-the-art algorithm.

We measured the performance of the models built by means of LambdaMart algorithm when exploiting a single feature. In Table 3.3 we reported for each feature the score it can achieve. Recall that the relatedness function is required to learn a score of a candidate entity w.r.t. to a correct entity, which in the table are denoted with c and e respectively. Therefore, $P(c|e)$ is the conditional probability of finding the candidate entity c given our actually mentioned entity e, while $P(e|c)$ is the converse.

Results are very similar for every quality measure. Let's consider *NDCG* @10. The function ρ^{MW} is the fourth most effective feature with a score slightly below that of Jaccard and Friend functions. The most effective feature is $P(c|e)$, that is the conditional probability of the finding a mention to entity c given a Wikipedia page that mentions the entity e. Note that this quite intuitive feature behaves largely better than ρ^{MW} with a score of .72, but it is however far from the score achievable with the full set of features. Also, note that statistic $P(c|e)$ comes from a collection being completely different from the test set, since it was computed on the Wikipedia corpus and not on the train collection. A third interesting property is the asymmetry of this feature.

The second column of Table 3.3 reports the rank assigned by feature selection algorithm. While $P(c|e)$ is ranked first being the most effective features, ρ^{MW} is ranked only 19-th. This is due to the heuristic strategy of the algorithm, which demotes features if they are similar to previously selected ones.

Figure 3.2 shows the result of a multidimensional scaling mapping of the 27 features into a 2-dimensional space, thus approximately preserving feature similarity. We measured the similarity between a feature pair according to the Kendall's τ

Features	Rank	NDCG@5	NDCG@10	P@5	P@10	MRR
P(c—e)	**1**	**0.68**	**0.72**	**0.47**	**0.33**	**0.80**
J(e, c)	**2**	0.62	0.66	0.44	0.31	0.75
Friend(e,c)	24	0.59	0.64	0.42	0.31	0.71
$\rho^{\mathsf{MW}}(e,c)$	19	0.59	0.63	0.42	0.31	0.72
$J_{\mathsf{in-out}}(e,c)$	26	0.60	0.63	0.42	0.30	0.74
AvgFr(e,c)	**3**	0.57	0.62	0.40	0.30	0.69
P(e,c)	27	0.56	0.60	0.39	0.28	0.70
$\rho^{\mathsf{MW}}_{\mathsf{in-out}}(a,b)$	9	0.56	0.60	0.40	0.29	0.71
$J_{\mathsf{in-out}}(e,c)$	**4**	0.54	0.58	0.39	0.28	0.67
$\rho^{\mathsf{MW}}_{\mathsf{out}}(a,b)$	17	0.52	0.55	0.37	0.27	0.65
$\chi^2(e,c)$	25	0.51	0.55	0.37	0.27	0.64
$P(e\|c)$	22	0.48	0.54	0.36	0.28	0.60
H(c)	**5**	0.48	0.51	0.30	0.20	0.68
$\chi^2_{\mathsf{out}}(e,c)$	16	0.47	0.50	0.34	0.24	0.61
AvgFr(c,e)	21	0.44	0.49	0.33	0.25	0.56
P(c)	13	0.47	0.49	0.29	0.19	0.66
PMI(e,c)	23	0.42	0.48	0.32	0.25	0.53
$\chi^2_{\mathsf{in-out}}(e,c)$	11	0.44	0.46	0.33	0.23	0.58
$P(e \to c)$	18	0.37	0.38	0.24	0.15	0.55
Link$(e \to c)$	20	0.37	0.38	0.24	0.15	0.55
$P(c \to e)$	12	0.35	0.36	0.22	0.14	0.52
Link$(c \to e)$	15	0.31	0.33	0.21	0.14	0.46
$KL(c\|\|e)$	10	0.32	0.32	0.19	0.12	0.51
Link$(c \leftrightarrow e)$	14	0.28	0.29	0.17	0.11	0.45
$KL(e\|\|c)$	8	0.26	0.28	0.17	0.11	0.44
P(e)	6	0.08	0.11	0.06	0.06	0.17
H(e)	7	0.08	0.11	0.06	0.06	0.17

Table 3.3: Entity ranking performance with a single feature. Features are sorted by *NDCG*@10.

coefficient. We can identify two interesting clusters. The first contains ρ^{MW} together with $J_{\mathsf{in-out}}$ and χ^2, and, indeed, the first two have identical performance. The second cluster includes the two best performing features $P(c|e)$, $P(e,c)$ and also Jaccard similarity. Even if the features in those clusters are similar w.r.t. the Kendall's τ coefficient, the score of the corresponding single feature model is very different, in particular for the best scoring $P(c|e)$. This suggest that the Kendall's τ coefficient may not be the best indicator in this context, and the feature selection may not be trivial.

Finally, in Figure 3.3 we measured the relative improvement provided by each

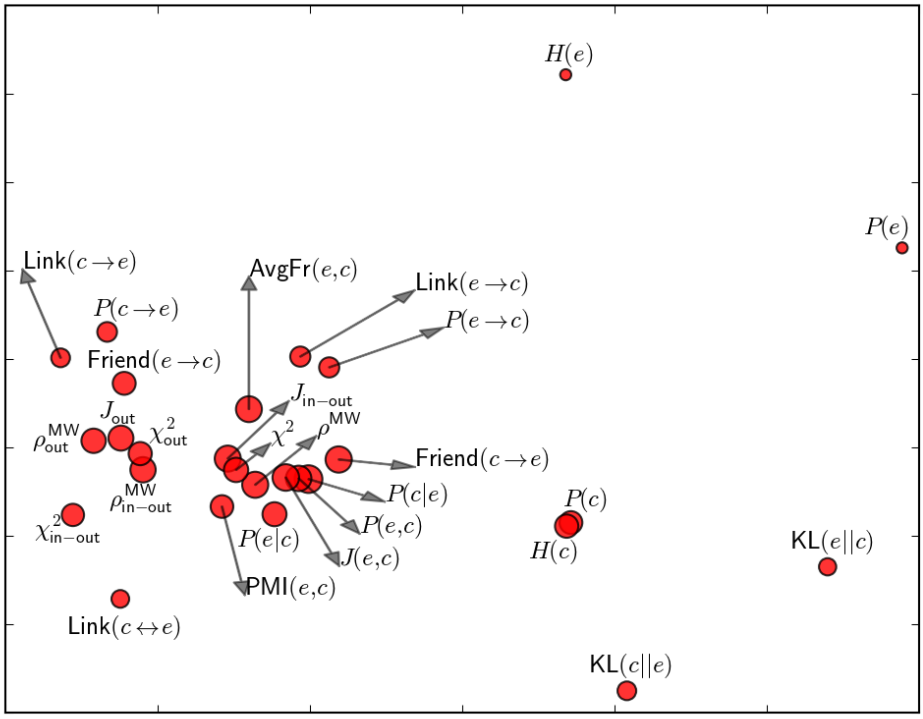

Figure 3.2: Multidimensional mapping of feature similarity computed using Kendall's τ coefficient. The size of each circle is proportional to the single-feature model score.

feature. Features are sorted according to the ranking given by the feature selection algorithm mentioned above, and we measured the performance of the model by adding features incrementally. The model achieves almost optimal performance with the first 5 features. Optimal performance are achieved after 9 features are introduced in to the model. This shows that not all the features are necessary, and that a wisely chosen subset of features can provide optimal performance, or help in trading accuracy with efficiency. Several existing feature selection techniques can be used to this end. However, this is outside the scope of this work.

3.5 Impact on Entity Linking

We run a set of experiments to show how the automatically machine-learnt relatedness function can be profitably exploited by a class of entity disambiguation algorithms. We plugged the machine-learnt function into several annotation methods, which can be considered the state-of-the-art ones:

WikiMiner. The method proposed by Milne and Witten that exploits the relatedness function to identify a subset of not ambiguous entities called *context*. Given an

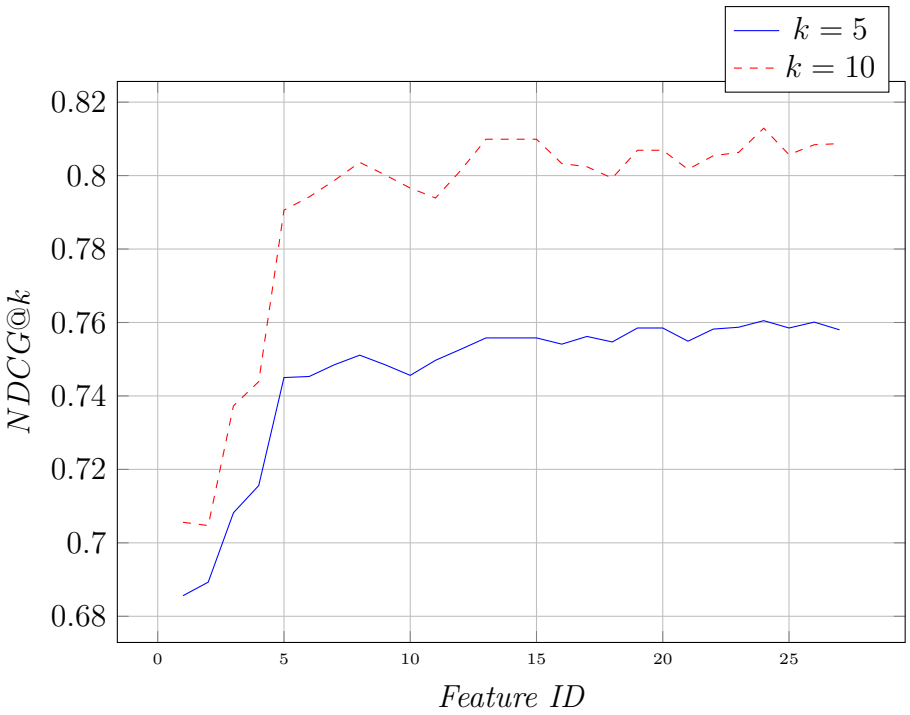

Figure 3.3: Incremental performance of $\rho^{\lambda\text{MART}}$.

ambiguous spot, the relatedness function is employed again to select the entity that is more coherent with the context;

Referent Graph. This method takes into account all the possible entities associated with the set of detected spots. The disambiguation is performed by modeling the entities as nodes of a complete graph, where the weight of each edge is the relatedness between the connected nodes;

TAGME. This annotator computes the weighted average relatedness between an entity and all the other possible entities associated with the spots. It disambiguates the entity by selecting the most common entities in the subset of the possible meanings with the highest average relatedness with the others.

With the exception of WikiMiner, the source code of the frameworks proposed is not publicly available. Furthermore the code released is not easy to extend for implementing other annotators. Annotation depends on several subtasks, i.e., *(i)* process Wikipedia (parse the dump, generate the possible spots, filter stop-words, etc.); *(ii)* perform the spotting (relying on a dictionary or using a name entity recognition framework, like the Stanford Named Entity Recognizer[6]); *(iii)* disambiguate the

[6] http://www-nlp.stanford.edu/software/CRF-NER.shtml

ambiguous spots, and *(iv)* rank entity candidates.

It is worth to observe that a good performance obtained in the first tasks may heavily impact on the performance of the whole system, as well as using a different dump of Wikipedia (i.e., old dumps contain less entities, but also have less ambiguity for each spot), or a different commonness or link probability thresholds. For these reasons, we strongly believe that for this kind of research it is important to share a unique framework where these tasks are well separated and easy to isolate in order to study their performance. This would also allow us to experiment hybrid solutions combining subtask solutions of different methods (e.g., the TAGME spotter with the WikiMiner disambiguation algorithm).

Due to the aforementioned reasons, we developed **Dexter** [36, 38], an entity linking framework, containing several utilities to manage the Wikipedia dump, a spotter based on the anchors and titles extracted from the dump, and data structures for retrieving all the features used by the annotators. Unlike WikiMiner, our framework does not rely on an external database to store the labels. In addition, during the execution it can maintain the model either on the disk or in main memory to improve performance. The framework runs also on normal hardware, since we exploit efficient data structures in order to maintain compressed data in main memory. The dataset we used, the source code and additional details about the framework can be found at this address: `http://dexter.isti.cnr.it/`. For a detailed description of Dexter refers to Section 3.6.

We implemented WikiMiner, Referent Graph and TAGME in our framework, in order to verify if our relatedness function is able to improve the annotator performance. During the implementation, we slightly modified WikiMiner and TAGME: in WikiMiner we decided to rank the entities using a linear combination of commonness, link probability, and average relatedness with the context (the authors employed a classifier trained with several features that were heavy to retrieve); in TAGME we relied on our spotter that returns all the possible spots detected in the text, while in the original version the authors employ a specific policy for deleting spots in case of overlaps (we remove overlapping annotations at the end of the process, relying on the final ranking of the entities). We set the commonness threshold to 0.03 and we discard spots with link probability lower than 0.02.

Note that we are not interested in the absolute entity-linking performance of WikiMiner, TAGME, and Referent Graph, but rather on how the relatedness function impacts on the disambiguation process. For this reason, we implemented all the three algorithms within the same framework, and thus providing them with the output of

the same spotter. For the same reason, the results of the Web services implementation of WikiMiner and TAGME are not reported. Those services use a different dump of Wikipedia, which is processed in a different way (e.g., tokenization, etc.), and they exploit a slightly different spotting algorithm, and this makes such results non significant within the scope of this work. However, it is important to report that we observed that our implementation always improves over the WikiMiner online service, and that it behaves only slightly worse then TAGME after the top 5 results, probably due to a different processing of Wikipedia.

We compared the results obtained by embedding different implementations of ρ: ρ^{MW}, $\rho^{\lambda\mathsf{MART}}$, and ρ^{GDBT}. Note that by embedding ρ^{MW} we are replicating the original algorithms that we consider as baselines to evaluate our proposed relatedness function.

The quality of the resulting algorithms is evaluated with the usual Precision@k ($k = 1, 5, 10$), Recall, and NDCG measures. We also report the interpolated precision at a certain recall cutoff r, iP_r with $r = 0.1$ and $r = 0.5$, the Mean Reciprocal Rank MRR and the Precision after R documents have been retrieved, where R is the total number of relevant entities for the document ($RPrec$).

We remind that in this evaluation we want to evaluate the number of correctly annotated entities *for a given document*; the evaluation is not spot-based, but we are rather considering the entity linking process as a whole, and its goodness on the full document.

The test dataset adopted is the same as the one of previous experiment, meaning that there is no overlap among the documents used for training the function ρ, and the documents used to evaluate its impact on the entity annotation process.

Table 3.4 reports the performance of the three annotators: for each annotator we show the performance using the original ρ^{MW} relatedness function, and then the effects of replacing the relatedness function with our machine-learnt relatedness $\rho^{\lambda\mathsf{MART}}$ and ρ^{GDBT}. The performance improvement given by the trained functions is significant:

Referent Graph. The proposed functions improve the ranking of results, in particular if we annotate only one entity per document using the ρ^{MW} relatedness only the 59% is correctly annotated, while with ρ^{GDBT} the percentage of correct documents is 74%. The relatedness function also reinforces the correct entities, improving the final ranking on the top entities as showed by the $NDCG$ measure which exhibits from a 14% up to a 25% of performance gain;

WikiMiner. ρ^{GDBT} improves both recall and $NDCG$, with gains superior to 10%. In both ρ^{GDBT} and $\rho^{\lambda\mathsf{MART}}$ the entity annotated with the largest confidence is

	Referent Graph			TAGME			WikiMiner		
	ρ^{MW}	$\rho^{\lambda\mathsf{MART}}$	ρ^{GDBT}	ρ^{MW}	$\rho^{\lambda\mathsf{MART}}$	ρ^{GDBT}	ρ^{MW}	$\rho^{\lambda\mathsf{MART}}$	ρ^{GDBT}
$P@1$	0.59	$0.68_{+15\%}$	$0.74_{+25\%}$	0.78	0.81 $_{+4\%}$	0.80 $_{+3\%}$	0.78	$\mathbf{0.86}_{+10\%}$	0.83 $_{+6\%}$
$P@5$	0.51	$0.62_{+22\%}$	$0.61_{+20\%}$	0.65	0.66 $_{+2\%}$	0.66 $_{+2\%}$	0.64	0.68 $_{+6\%}$	**0.69** $_{+8\%}$
$P@10$	0.44	$0.50_{+14\%}$	$0.51_{+16\%}$	0.50	0.50 $_{+0\%}$	0.51 $_{+2\%}$	0.50	0.51 $_{+2\%}$	**0.53** $_{+6\%}$
$iP_{r=0.10}$	0.76	$0.84_{+11\%}$	$0.87_{+14\%}$	0.87	0.89 $_{+2\%}$	0.89 $_{+2\%}$	0.88	**0.92** $_{+5\%}$	0.91 $_{+3\%}$
$iP_{r=0.50}$	0.55	$0.69_{+25\%}$	$0.70_{+27\%}$	0.67	0.68 $_{+1\%}$	0.69 $_{+3\%}$	0.66	$0.73_{+11\%}$	$\mathbf{0.77}_{+17\%}$
$NDCG$	0.64	0.70 $_{+9\%}$	$0.72_{+13\%}$	0.68	0.69 $_{+1\%}$	0.69 $_{+1\%}$	0.66	0.72 $_{+9\%}$	$\mathbf{0.75}_{+14\%}$
MRR	0.73	$0.81_{+11\%}$	$0.84_{+15\%}$	0.87	0.89 $_{+2\%}$	0.89 $_{+2\%}$	0.87	**0.92** $_{+6\%}$	0.90 $_{+3\%}$
$NDCG@5$	0.55	$0.67_{+22\%}$	$0.68_{+24\%}$	0.72	0.74 $_{+3\%}$	0.73 $_{+1\%}$	0.71	0.76 $_{+7\%}$	**0.77** $_{+8\%}$
$NDCG@10$	0.57	$0.68_{+19\%}$	$0.70_{+23\%}$	0.70	0.70 $_{+0\%}$	0.71 $_{+1\%}$	0.69	0.73 $_{+6\%}$	**0.75** $_{+9\%}$
$Recall$	0.76	**0.77** $_{+1\%}$	**0.77** $_{+1\%}$	0.68	0.69 $_{+1\%}$	0.69 $_{+1\%}$	0.64	0.70 $_{+9\%}$	$0.75_{+17\%}$
$Rprec$	0.46	$0.58_{+26\%}$	$0.60_{+30\%}$	0.56	0.58 $_{+4\%}$	0.58 $_{+4\%}$	0.56	0.60 $_{+7\%}$	$\mathbf{0.64}_{+14\%}$

Table 3.4: Entity Linking performance

correct in more than the 80% of the documents, with a improvement of 6% (ρ^{GDBT}) and of 10% ($\rho^{\lambda\mathsf{MART}}$) with respect to ρ^{MW};

TAGME. Recall, $NDCG$, and precision exhibit a positive improvement (from 1% up to 4%). The reader will note that ρ^{GDBT} and $\rho^{\lambda\mathsf{MART}}$ does not improve TAGME in the same measure as the other annotators: this is not surprising because the TAGME annotator is designed to manage short texts, and relies less on the relatedness and more on the commonness.

In general, the best result quality was obtained using the ρ^{GDBT} function.

3.6 Dexter - Entity Linking Framework

Several approaches have been recently proposed in literature to address the *Entity Linking* problem, but unfortunately only a few authors have released the source code or the APIs to test their system. Thus performing a fair comparison among different *Entity Linking* techniques is very hard. Moreover, Hackey *et al.* [72] propose a framework (not published) for entity linking in which they implemented and compared three methods in the state-of-the-art. In their experiments they found that spotting is more important than disambiguation and that mixing the spotting and the disambiguation strategies of different methods can lead to interesting results.

A good performance obtained in spotting may heavily impact on the performance of the whole system, as well as using a different dump of Wikipedia (i.e., old dumps contain less entities, but also have less ambiguity for each spot), or pruning all the candidate entities of a spot which have commonness below a given threshold, or spots with a low *link probability* (probability to be a link, estimated as the occurrences of the text as anchor divided the occurrences as pure text). Moreover, *efficiency* of the

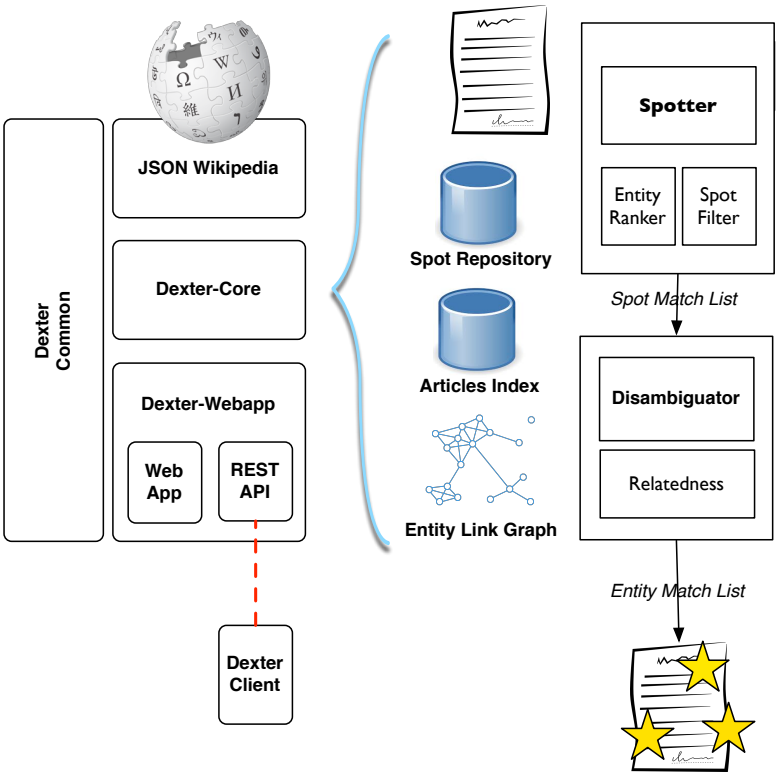

Figure 3.4: DEXTER Architecture

proposed methods is in many cases ignored even if, depending on the use case, it could be of paramount importance (e.g., for annotating a huge repository of web pages or for enriching queries on a search engine).

For these reasons, we strongly believe that for this kind of research it is important to share a unique framework where spotting, disambiguation and ranking are well separated and easy to isolate in order to study their performance. Our proposal goes in the direction of developing an open and flexible framework for implementing various entity linking strategies. The framework, called DEXTER[7], provides several facilities for implementing annotation strategies. Differently from other approaches which require high-performance hardware or to install additional software (e.g., databases), DEXTER is a standalone program, written in Java, designed to easily work with a small effort from the user and to run on commodity hardware. It is organized in several Maven[8] modules as shown in Figure 3.4:

Json-Wikipedia[9] This module converts the Wikipedia XML Dump in a JSON

[7]The project page is http://dexter.isti.cnr.it
[8]http://maven.apache.org/
[9]json-wikipedia is available at https://github.com/diegoceccarelli/json-wikipedia

Dump, where each line is a JSON record representing an article. The parser is based on the MediaWiki markup parser UKP[10]. DBPedia only contains semistructured data extracted from the dump (mainly from the infoboxes) in RDF format, while JSON-Wikipedia contains other fields, e.g., the section headers, the text (divided in paragraphs), the templates with their schema, emphasized and so on. These fields are not properly attributes of an entity but they can be useful for performing the annotation or a posteriori, for presenting an annotated entity. Another main difference is that in DEXTER dump all the articles are converted to JSON records, so we have also Disambiguation records, Redirect records, Category records etc. (the type of the record is encoded in a *type* field). The module is designed to manage different languages. Given a locale file describing how disambiguations, categories, redirects, etc are denoted in a given language, DEXTER parses the XML dump in the specified language and produces the JSON dump. All the languages will share the same JSON schema. This should simplify the development of techniques combining different languages. Moreover, JSON is easy to parse and to be used in scalable Map Reduce frameworks like Hadoop, Pig or Cascading. In the future, we plan to improve the creation of entity linking models to be used in map reduce jobs.

Dexter-Core The core contains all the code to manipulate the JSON dump in order to generate: the *spot repository*, the *article index*, and the *entity graph*. The spot repository contains all the anchors used in Wikipedia for intra-linking the articles. For each spot, the spot index contains the link probability and the list of entities that could be represented by the spot (i.e, all the articles targeted by that particular spot). The article index is a multi-field index of the article, built by using Lucene. The entity graph stores the connections among the entities. The Shingle Extractor produces a list of possible spots from a given document. At the time of writing, the extractor produces all the possible n-grams of terms, where n ranges from one to six. The Spotter then associates with each fragment a list of entity candidates (if any) using the spot repository. Finally the Tagger takes in input the list of spot matches produced by the spotter and selects the best entity for each spot, performing the disambiguation if the spot has more than one candidate. Disambiguation can be performed using the features provided by the spot repository, the article index, and the entity graph. The

[10] http://www.ukp.tu-darmstadt.de/software/jwpl/

Tagger outputs an entity match list, with the position in the original text of each annotated entity and a confidence score.

Dexter-Common This module contains the domain objects, shared among all the modules of DEXTER.

Dexter-Webapp This module exposes a REST API for performing the annotations. It also implements a simple web interface for performing a demo. The current version of the REST API is briefly described in Table 1, and it is organized in 4 logical categories: the Annotate API, used for annotating a document, the Spot API that allows to retrieve the candidate spots in a document and to visualize their features, the Graph API and the Category API that allow to browse respectively the Wikipedia article's link graph and the category graph. The current API is available and testable. We provide a well written documentation for each method, in a web page that also allows the user to test the service.

Dexter-Client A simple client to perform EL from a client machine, implicitly calling the REST API.

DEXTER come out of the box with the implementation of three *Entity Linking* algorithms described in literature: TAGME [61], the collective linking approach [73] and WikiMiner [116]. It allows to replace and combine different implementation of each module (the spotter, the disambiguator, the relatedness function *etc.*). An EL annotation can then be performed providing to the linker the symbolic names of the components that the developer wants to use (the spotter x, the disambiguator y ...), allowing to make use of different EL techniques at run-time.

Another interesting feature is the possibility to annotate *semi-structured documents*, in contrast to traditional EL frameworks that annotate only flat text, i.e., a plain string. Hence documents can be composed by several fields (*e.g.*, title, headlines, paragraphs), and new spotter/disambiguator strategies could exploit the information about the structure of a document. It is worth to highlight the system offers a wide set of REST API methods, to facilitate the interaction with the system and the evaluation of single module only.

3.7 Conclusions

In this chapter, we have proposed a machine learning based approach aimed at discovering the entity relatedness function that can better support the entity linking

task. We illustrated some of the properties that such function should preserve, and we presented a simple method to generate a training set form a collection of document human assessed entity linked documents. We casted the problem of discovering a suitable entity relatedness function into a learning to rank formulation. Our proposed approach is thus able to learn how to wisely blend the available features to generate a good entity relatedness function. We demonstrated that by exploiting our framework it is possible to better estimate the relatedness of two entities, and to compare and improve the performance of different state-of-the-art entity linking algorithms.

Chapter 4

SEL: A Unified Algorithm for Entity Linking and Saliency Detection

The *Entity Linking* task consists in automatically identifying and linking the entities mentioned in a text to their URIs in a given Knowledge Base, e.g., Wikipedia. However, not all the entities mentioned in a document have the same relevance and utility in understanding the topics being discussed. Thus, the related problem of identifying the most relevant entities present in a document, also known as *Salient Entities*, is attracting increasing interest.

In this chapter we propose *SEL*, a novel supervised two-step algorithm comprehensively addressing both entity linking and saliency detection. The first step is based on a classifier aimed at identifying a set of candidate entities that are likely to be mentioned in the document, thus maximizing the precision of the method without hindering its recall. The second step is still based on machine learning, and aims at choosing from the previous set the entities that actually occur in the document. Indeed, we tested two different versions of the second step, one aimed at solving only the entity linking task, and the other that, besides detecting linked entities, also scores them according to their saliency. Experiments conducted on two different datasets show that the proposed algorithm outperforms state-of-the-art competitors, and is able to detect salient entities with high accuracy.

Furthermore, we employed *SEL* for Extractive Text Summarization. We found that entity saliency can be incorporated into text summarizers to extract salient sentences from text. The resulting summarizers outperform well-known summarization systems, proving the importance of using the *Salient Entities* information.

4.1 Introduction

Lately, much research has been spent to devise effective solutions to Entity Linking (EL). The task, as introduced in Section 2.1.4 and formalized in Section 3.2, consists in finding small fragments of text referring to an entity that is listed in a given knowledge base, e.g., Wikipedia. Natural language ambiguity makes this task non trivial. Indeed, the same entity may be mentioned with different text fragments, and the same mention may refer to one of several entities.

EL is strictly correlated with another task, referred to as *document aboutness* problem [127] or *Salient Entities* (SE) discovery problem [68], which goal is labeling the entities mentioned in the document according to a notion of saliency, where the most relevant entities are those that have the highest utility in understanding the topics discussed.

This task can be combined with EL. The easiest integration is to perform the SE discovery as a subsequent step to EL, by finally choosing the most relevant entities that have high utility in understanding the topics being discussed among the set of entities returned by the EL algorithm. However, we claim this pipeline approach is somehow limiting since the disambiguation could benefit from the saliency signal.

Consider for instance the following example, where the most relevant entities are probably the ones referred by mentions **Maradona** and **1982 Cup**:

> **Maradona** (→Diego_Maradona) played his first **World Cup tournament** (→FIFA_World_Cup) in 1982, when **Argentina** (→Argentina_national_football_team) played **Belgium** (→Belgium_national_football_team) in the opening game of the **1982 Cup** (→1982_FIFA_World_Cup) in **Barcelona** (→Barcelona).

Entity saliency impacts on information extraction from text in a broader sense. Consider for example a semantic clustering approach where linked entities are exploited to provide a high-level summary of each document. In this application scenario the capability of weighting entities on the basis of their saliency is crucial. In addition, the knowledge about the saliency of entities recognized by an EL algorithm in a document should also impact on the evaluation of the effectiveness of the EL algorithm itself. Let us come back to the previous example where the entity **1982 Cup** provides much more information about the document than the entity **Barcelona**. Thus, an EL algorithm that links only the mention **1982 Cup** should be preferred in terms of effectiveness to another algorithm that only links the spot **Barcelona**.

4. SEL: A Unified Algorithm for Entity Linking and Saliency Detection

In this chapter we propose a novel supervised *Salient Entity Linking* (*SEL*) algorithm to comprehensively address EL and SE detection. The *SEL* algorithm entails two steps: *Candidate Pruning* and *Saliency Linking*. During the *Candidate Pruning* step, a classifier is used to prune the large set of candidate entities generated by the spotting phase. The aim is to detect a relatively small collection of candidates that encompasses all the entities actually mentioned in the document. Thus the emphasis is on training a classifier able to achieve a good precision without hindering recall. The proposed approach has proved to outperform heuristic methods that prune unlikely candidates on the basis of simple likelihood measures such as *commonness* or *link probability* [111, 116]. The *Saliency Linking* step also exploits machine learning, and, in addition to addressing EL, it is able to predict the saliency of the entities that survived the *Candidate Pruning* step. Thanks to the *Candidate Pruning* step, the candidate set processed during the *Saliency Linking* step is less noisy and smaller in size, which allows to use more complex and powerful graph-based entity correlation features.

The experiments conducted on two different datasets show that *SEL* outperforms state-of-the-art competitors in the EL task. In addition, it is able to detect salient entities with high accuracy. Since both steps of the algorithm are based on machine learning, we also analyzed in depth feature importance, and we took into consideration feature extraction costs. We show that an efficient and effective classifier for the first step can be trained on the basis of a small and easily computable set of features. This is particularly important since the classifier must be applied to a very large set of initial candidates. On the other hand, in the second step we have a reduced number of survived candidates and we benefit from the exploitation of further graph-based features, which are more expensive to compute, but which are proved to be very effective for improving the quality of entity linking and saliency detection.

Furthermore, we report on the exploitation of our entity saliency detection technology to feed novel text summarization techniques. SEL allows the entities that have high utility in understanding the topics of a document to be identified. The knowledge of such entities can be used to design novel extractive summarizers boosting the sentences mentioning the most salient entities in the document. We evaluated these new text summarizers on well-known single-document and multi-document summarization datasets, providing an empirical evidence of the positive effect of the salient-derived features.

In summary, the main contributions of this chapter are the following:

- a novel *Salient Entity Linking* (*SEL*) algorithm, that accurately estimates entity saliency and outperforms state-of-the-art EL techniques by providing a comprehensive solution to the EL and the SE detection problems.
- an evaluation of a wide set of heterogeneous features, including novel features, used to represent entities within the machine learning algorithms adopted.
- a novel framework, namely ELIANTO, for manually annotating text with both entities and saliency scores, thus allowing the creation of golden annotation datasets.
- a novel dataset of news manually annotated with entities and their saliency, hereinafter publicly available to the research community.
- novel single-document and multi-document summarizers that employ features based on entity-saliency to extract central sentences from documents.

4.2 Related Work

Entity Linking. Entity Linking algorithms usually work by following a well defined schema, that could be roughly summarized in three steps: *spotting*, *disambiguation* and *pruning*. *Spotting* detects potential mentions in a text and, for each mention, produces a list of *candidate entities*. *Disambiguation* aims at selecting a single entity for each mention produced in the previous step, by trying to maximize some *coherence* measure among the selected entities in the document. *Pruning* detects and removes non-relevant annotations in order to improve the precision of the system. In performing the three steps, EL algorithms rely on three effective signals: *(i)* the probability for a mention to be a link to an entity (*link probability*); *(ii)* the prior probability for a mention to refer to a specific entity (*commonness*); *(iii)* the *coherence* among the entities in a document, e.g., estimated by the Milne-Witten *relatedness* [116]. In addition to annotate mentions to the entities, EL algorithms usually assign to each annotation a *confidence score*, roughly estimating the correctness of the annotation.

Several EL approaches have been proposed following the problem formalization given by Mihalcea and Csomai with *Wikify* [111]. A substantial improvement has been the WikiMiner approach proposed by Milne and Witten [116]. It works by first identifying a set of non-ambiguous mentions and then using this set to disambiguate the ambiguous ones. Ferragina and Scaiella proposed an improved approach called Tagme [61], which tries to find a collective agreement for the best candidates using a voting scheme based on the the Milne-Witten relatedness. Candidate entities with a coherence below a given threshold are discarded, and for each mention the one with the largest commonness is selected. In Spotlight [110], Mendes *et al.* represent

each entity with a context vector containing the terms from the paragraphs where the entity is mentioned; they also exploit NLP methods, removing all the spots that are only composed of verbs, adjectives, and prepositions. In Wikifier 2.0 [42] (which is an extension of [137]), Cheng and Roth use a machine learning based hybrid strategy to combine local features, such as commonness and TF-IDF between mentions and Wikipedia pages, with global coherence features based on Wikipedia links and relational inference. This system combines Wikipedia pages, gazetteers, and Wordnet. In AIDA [78], Hoffart *et al.* proposed a weighted mention-entity graph for collective disambiguation. This model combines three features into a graph model: entity popularity, textual similarity (keyphrase-based and syntax-based) as well as coherence between mapping entities. The authors also published a manually annotated dataset for EL, named AIDA-CoNLL 2003. In WAT [128] authors extended Tagme with a new spotting module (using gazetteers, named-entity recognition analysis and a binary classifier for tuning performance), voting-based and graph-based disambiguation approaches as well as a pruning pipeline. Note that neither the source code nor a remote annotation service of WAT is publicly available. One of the main conclusions from their experiments was that while many systems focused on improving disambiguation, the spotter and the pruner are actually responsible for introducing many of the false positives in the EL process. A thorough overview and analysis of the main approaches to EL and their evaluation is presented by Shen *et al.* [151].

Entity Saliency. The problem of understanding the main topics of a document has been the goal of many IR tasks, including latent semantic topics and text summarization. In this work we tackle the related task of finding the most important entities mentioned in a given document. This task has previously been referred to as *document aboutness* [68] or *salient entity discovery* [145] problem.

Gamon *et al.* [68] studied the *aboutness* problem referred to the named entities occurring in Web pages. The approach used is partially inspired by [127], where click-through data are exploited to rank named entities mentioned in queries. The authors estimate the entity saliency for a Web page by exploiting the click-through recorded in a query log. Roughly, a document is considered to be relevant for a given entity when it is returned by a Web search engine and clicked by multiple users in answer to queries mentioning the entity. A number of text-based features are proposed in the paper, most of them applicable only to a Web scenario, e.g., url depth. In such work entities are just pieces of text (and not entities listed in a given knowledge base) and the disambiguation problem is not tackled at all.

When entities in a knowledge base such as Wikipedia are considered, rich contextual information coming from its graph structure can be fruitfully exploited. Given the set of entities occurring in a document, an *entity graph* can be built by projecting the subgraph of the knowledge base graph including all the entities possibly mentioned in the document. Entities can finally be ranked according to some measure of their importance in such a graph.

Dunietz and Gillick [59] proposed a method for classifying salient entities mentioned in news by exploiting graph-based measures. They show that the eigenvector centrality computed on the mentioned entities can slightly improve the performance of a binary classifier aimed at discriminating salient entities with respect to a classifier machine-learnt with text-based features only. The same task is addressed in [145], where text-based features are fruitfully complemented with graph-based ones to improve accuracy. The work by Dunietz and Gillick is closely related to ours but, in order to automatically generate the ground truth, they consider as salient entities those mentioned in the abstract of the news. Thus, the authors cannot use features related to the position of the mention for predicting the saliency, and how the graph-based and other features contribute to improve the classification accuracy. We instead exploited a manually assessed dataset that allows us to perform this analysis. Moreover, their paper assumes to know in advance the correct entities mentioned in the document, and addresses only the problem of ranking them by saliency. Instead we addressed comprehensively the EL and SE problems, and studied the importance of different features for identifying the correct entities mentioned as well as their saliency.

4.3 The Salient Entity Linking Algorithm

Let $\mathbb{KB}$ be a knowledge base with a set of entities $\mathbb{E}$. The EL problem is to identify the entities $\mathbb{E}_D \subseteq \mathbb{E}$ mentioned by the *spots* S_D of a given document D. As in state-of-the-art approaches, Wikipedia is used as knowledge base and every Wikipedia article is considered as an entity. Entities that are not in Wikipedia are not linked (i.e., we do not take into account the NIL problem).

In this work the *saliency* $\sigma(e|D)$ of the entities e mentioned in a document D is also considered. Without loss of generality, we define the domain of function σ as the set $\{0, 1, 2, 3\}$, with the following meaning:

- **3 - Top Relevant**: the entity describes the main topics or the leading characters of a document;

- **2 - Highly Relevant**: these are satellite entities that are not necessary for understanding the document, but they provide important facets;
- **1 - Partially Relevant**: entities that provide background information about the content of the document, but disregarding them would not affect negatively the comprehension of the document;
- **0 - Not Relevant/Not Mentioned**: any other entity in $\mathbb{E}$ that is not relevant or not mentioned in D.

The SE detection problem is to predict the saliency $\sigma(e|D)$ for each $e \in \mathbb{E}$. Note that the EL and SE problems are correlated and they almost coincide when a binary saliency function returning the relevance of an entity for D is adopted, i.e., $\sigma(e|D) = 1$ if $e \in \mathbb{E}_D$ and 0 otherwise.

The proposed *SEL* algorithm is able to discover $\mathbb{E}_D$, and in addition solves the SE problem, thus predicting $\sigma(e|D)$ for each $e \in \mathbb{E}_D$. The first step of *SEL* performs a *spotting* process, which detects potential entity mentions in the text. The hyperlink information of Wikipedia is exploited for this purpose. If the given document D contains a fragment of text s that is used as anchor text in Wikipedia to link to an entity e, then e is considered a *candidate entity* for the spot s. Since the same anchor text can be used in Wikipedia to reference any of several entities, a spot s might be associated with several candidate entities. The set of candidate entities can be very large, which makes it difficult to select the single correct entity for each spot, i.e., to disambiguate spots. However not all the possible entities are equally probable for a given spot, and candidate entities can be pruned to make the subsequent *disambiguation* step easier.

The first novelty in the proposed *SEL* algorithm is the usage of a machine-learnt classifier with a set of easy-to-compute features to prune the candidate entities before disambiguation takes place. The goal of such classifier is to improve the precision of the state-of-the-art unsupervised techniques, without hindering recall: the classifier aims at filtering a small set of candidates without pruning any entity in $\mathbb{E}_D$. To train the classifier we investigated a novel and rich set of features, from which we selected only 8 *light* features.

The second step implements spot disambiguation. We devise two different solutions: the former aimed at solving the EL problem only, and the latter that, besides linking spots to correct entities, also scores them according to their saliency, thus combining the EL and SE discovery tasks. Also this step is based on machine-learning, this time

using a regressor which is well suited for both the binary EL task (with a machine-learnt threshold value), or the multiclass SE problem.

The second novelty in the *SEL* algorithm is the blending of disambiguation and saliency prediction in a single step. We claim that this blending makes it possible to improve the accuracy of disambiguation for those spots/entities that are likely to be salient. The reason is that an EL task should not link everything, but just the relevant concepts, i.e., the salient ones (thus excluding not relevant concepts, with a saliency score of 0). To learn an effective regressor for disambiguation, we analyzed a feature set wider than in the first step. By focusing on the relatively small number of candidate entities coming from the first step, it is possible to exploit complex and computationally *heavy* features, like those considering the entity relatedness graph.

4.3.1 Supervised Candidate Pruning

Potential entity mentions in a text are detected by exploiting the $\mathbb{K}B$: all the possible spots occurring in a given document D are matched against all the anchor texts and page titles in Wikipedia, and in case of an exact match (without any normalization on the text), a relationship is created between a spot s and the entities referred by s in Wikipedia.

Due to language ambiguity, the number of entities for each spot can be large. Formally, let $S_D = \{s_1, s_2, \ldots\}$ be the set of spots detected in D and $C_D = \{c_1,\ c_2, \ldots\}$, $C_D \subseteq \mathbb{E}$, the set of candidate entities, each of which is associated with some spot s_i. Indeed, the output of the spotting phase is a directed bipartite graph $G_D = (S_D, C_D, E_D)$, where E_D are the edges of the graph such that $(s_i, c_j) \in E_D$ if s_i is a text fragment used in Wikipedia for referring to entity $c_j \in \mathbb{E}$.

The goal of *Candidate Pruning* is to devise an effective entity pruning function ϕ: given a set of candidate entities C_D of the bipartite graph G_D identified by the spotting phase, ϕ finally produces a new set $C'_D = \phi(C_D)$, such that $|C'_D|$ is minimized and $|C'_D \cap \mathbb{E}_D|$ is maximized.

State-of-the-art algorithms perform a Heuristic Pruning (HP) of candidate entities C_D, by exploiting two measures, namely *commonness* and *link probability*, that can be precomputed as follows:

- The commonness of a candidate $c_j \in C_D$ for spot $s_i \in S_D$ is defined as the prior probability that an occurrence of an anchor s_i links to c_j. The commonness is a property of the edges of our bipartite graph. Given a spot $s_i \in S_D$, it is possible rank the outgoing edges and remove edges with low commonness.

- The link probability for a spot $s_i \in S_D$ is defined as the number of occurrences of s_i being a link to an entity in $\mathbb{K}B$, divided by its total number of occurrences in $\mathbb{K}B$. Therefore a spot with low link probability is rarely used as a mention to a relevant entity, and can be pruned from graph G_D.

Let τ_c and τ_{lp} be the *minimum commonness* and the *minimum link probability* (heuristic thresholds), it is possible to discard those graph edges with commonness lower than τ_c, and those spots with *link probability* lower than τ_{lp}. Note that when a spot s_i is pruned, also its outgoing edges are removed. After pruning the graph G_D on the basis of τ_c and τ_{lp}, some candidate entities in C_D may result disconnected from any spot, and they can thus be removed as well.

Setting a minimum threshold on commonness and link probability has been proven to be a simple and effective strategy, although heuristic, to limit the number of spots and associated candidate entities, without harming the recall of the EL process. Table 4.1 reports the performance of such heuristic pruning (HP) method over the well-known AIDA-CoNLL 2003 dataset released by [78] and over a novel manually annotated dataset named Wikinews (see Section 4.4.1 for a description of the two datasets), for different values of τ_c and τ_{lp}. The metrics adopted are precision (i.e., ratio of positive entities retained to the whole set of entities retained) and recall (i.e., ratio of positive entities retained to the whole set of positive entities). It is worth noting that commonly adopted thresholds ensure a good recall at the cost of a very low precision. The same table also reports the performance of the proposed solution, which is described below.

For $\tau_c = 2\%$ the HP obtains up to 2% of improvement in recall with respect to the proposed method. On the other hand, with this setting the HP obtains a maximum precision of only 0.074, while the supervised solution achieves a precision of 0.367, i.e., 500% of improvement. Further experimental analysis is discussed in Section 4.2. Note that both Wikiminer [116] and Tagme [61] use $\tau_c = 2\%$, with the former using $\tau_{lp} = 6.5\%$ and the latter exploiting a more complex usage of the link probability value. In the following, we refer to the heuristic pruning strategy of Wikiminer as HP_W. This strategy is highlighted in Table 4.1 with a light gray background.

The *Candidate Pruning* method improves on the previous heuristic strategies by using a supervised technique. A binary classifier is machine-learnt to distinguish between relevant and irrelevant entities. Note that saliency has not taken into account in this step: a candidate entity c_j is considered relevant *iff* it is mentioned by the given document D. The training set is built from the ground truth on the basis of the

Table 4.1: Spotting performance for different values of τ_c and τ_{lp} on AIDA-CoNLL 2003 and Wikinews datasets. The heuristic pruning strategy of Wikiminer is highlighted with a light gray background.

		CoNLL		Wikinews	
Commonness	Link-Probability	Precision	Recall	Precision	Recall
0.005	0.02	0.022	0.907	0.016	0.925
0.005	0.03	0.025	0.900	0.020	0.922
0.005	0.04	0.029	0.893	0.024	0.919
0.005	0.05	0.036	0.893	0.027	0.915
0.005	0.065	0.044	0.892	0.033	0.909
0.01	0.02	0.032	0.891	0.026	0.921
0.01	0.03	0.038	0.884	0.031	0.917
0.01	0.04	0.043	0.877	0.036	0.915
0.01	0.05	0.052	0.877	0.041	0.911
0.01	0.065	0.063	0.876	0.050	0.905
0.02	0.02	0.048	0.864	0.040	0.915
0.02	0.03	0.056	0.856	0.048	0.911
0.02	0.04	0.063	0.850	0.056	0.909
0.02	0.05	0.074	0.850	0.062	0.906
0.02	0.065	0.089	0.849	0.074	0.900
0.04	0.02	0.072	0.839	0.060	0.908
0.04	0.03	0.082	0.831	0.072	0.904
0.04	0.04	0.092	0.826	0.083	0.901
0.04	0.05	0.103	0.826	0.092	0.898
0.04	0.065	0.121	0.826	0.109	0.893
Proposed *Candidate Pruning*		0.367	0.848	0.361	0.867

bipartite graph $G_D = (S_D, C_D, E_D)$ generated by the spotting phase. A positive label is associated with $c_j \in C_D$ if $c_j \in \mathbb{E}_D$, and a negative label otherwise. Each entity $c_j \in C_D$ is represented with a large set of features extracted from the document, from the bipartite graph G_D and from the knowledge base $\mathcal{KB}$. These features are deeply discussed in Section 4.3.3. Eventually, only the candidate entities that are predicted to be relevant by the classifier are saved for the subsequent *Saliency Linking* step.

There are a couple of aspects relative to the ground truth that is worth discussing. First, class imbalance characterizes the training dataset, since on average we have that $|\mathbb{E}_D \cap C_D| \ll |C_D|$. Unfortunately a classifier machine-learnt from a training set with a strongly skewed class distribution may lead to poor performance. This is because most algorithms minimize the misclassification rate on the training set, hence favoring

most frequent class, which in the specific case is the negative one. In order to deal with this issue, a cost model is introduced. Therefore, the classifier incurs a higher penalization when misclassifying an instance in a rare class. Another key property which deserves attention concerns the choice of the feature space used to represent instances. Indeed, we distinguish between *light* and *heavy* features, i.e., either cheap or expensive to compute. We show that a small subset of these light features is able to generate a good classifier for the *Candidate Pruning*. The resulting classifier improves state-of-the-art heuristic techniques in terms of precision without hindering the recall, thus retaining most of the positive entities for the *Saliency Linking* step.

4.3.2 Supervised Saliency Linking

The *spotting* step in EL algorithms is always followed by a *disambiguation* phase: among the several candidates for a given spot, only one entity can be selected. The proposed *SEL* algorithm distinguish the following two tasks:

i) disambiguating spots also using contextual features, thus addressing the EL problem;

ii) predicting a saliency score for the relevant entities, thus addressing the EL and SE problem at the same time.

Both tasks are solved by learning a predictor of entity saliency. In the former case, an entity is considered relevant or irrelevant, i.e., $\sigma(e|D) \in \{0, 1\}$, while, in the latter, we have several degrees of relevance, i.e., $\sigma(e|D) \in \{0, 1, 2, 3\}$. The training dataset is built from the ground truth by considering only the candidate entities filtered by the *Candidate Pruning* step, and each entity c_j is labeled according to $\sigma(c_j|D)$. Note that all candidate entities c_k not mentioned in the document are labeled with $\sigma(c_k|D) = 0$.

This training dataset has two interesting properties. First, thanks to the *Candidate Pruning* step, the number of irrelevant entities is significantly reduced, and therefore the predictor is able to train on a quite balanced dataset with less noise. Second, by having a smaller number of candidate entities to deal with, it is possible to exploit more complex and powerful features able to better capture entity correlations. Indeed, besides the set of light features used in the *Candidate Pruning* step, an additional set of *heavy* features is added. These are mainly computed on the graphs induced by the Wikipedia hyperlinks, thus modeling the relationships among the candidate entities. It is worth remarking that this expensive feature extraction becomes feasible because

the first step is able to strongly prune the original candidate set C_D. This new set of features is discussed in Section 4.3.3.

We remark that the *Saliency Linking* step implements disambiguation and saliency prediction at the same time. Disambiguation occurs implicitly as an incorrect entity c_k for a spot is predicted to have no saliency, i.e., $\sigma(c_k|D) = 0$. By tackling disambiguation and saliency prediction at the same time *SEL* achieves the goal of being accurate in linking the most relevant entities.

Note that during the *Saliency Linking* step the graph G_D is not considered, except via the features computed. When predicting the saliency of an entity, no information about the predicted saliency of other entities is exploited. Therefore, it is possible to have spots without any predicted relevant entity, and spots with more than one relevant entity. If needed, this can be easily fixed with a post-processing step not implemented in this work for the following reasons. First, it is much easier and clearer to consider the output of the *Saliency Linking* step as a flat set of entities, thus making it possible to easily adopt standard information retrieval measures, such as precision and recall. Second, it might be interesting in some application scenarios to have more than one annotation per spot, especially when more than one *facet* is relevant.

4.3.3 Features

Given the candidate entities devised by the spotting phase in document D, the *SEL* algorithm represents with a vector of numerical features each candidate entity $c_j \in C_D$ in the bipartite graph $G_D = (S_D, C_D, E_D)$. Specifically, we distinguish between *light* features (i.e., cheap to be computed) which are generated for all $c_j \in C_D$, and *heavy* features (i.e., computationally expensive) which are computed only for the filtered candidate entities $C'_D = \phi(C_D) \subseteq C_D$, where $|C'_D| \ll |C_D|$.

Light features. Light features, illustrated in Table 4.2, are mainly derived from *attributes* associated with the mentions in S_D, which are then aggregated to build features for the mentioned entities. Some of them are computed on the basis of the occurrences of spots $s_i \in S_D$ within document D. For example, the positions of spots (1–3), their count (4), some typesetting features (5–7), their length (8). Features 9–10,12,18, rely instead on Wikipedia, but they are precomputed and stored in the dictionary used for spotting. We included features related to spots ambiguity, see 16–17. Finally, we included two novel features, 19 and 21, trying to blend together commonness, link probability and ambiguity signals.

Table 4.2: Light Features for Supervised Candidate Pruning: features are relative to a candidate entity c_j

1. **positions**	first, last, average, and standard deviation of the normalized positions of the spots referring to c_j
2. **first field positions**	document D is subdivided in 4 fields: *the title*, *the first three sentences*, *the last three sentences*, and *the middle sentences*; the normalized position of the first spot referring to c_j is computed for each field
3. **average position in sentences**	the average position of spots referring to c_j across the sentences of the document (salient entities are usually mentioned early)
4. **field frequency**	number of spots referring to c_j computed for each field of the document
5. **capitalization**	True *iff* at least one mention of c_j is capitalized
6. **uppercase ratio**	maximum fraction of uppercase letters among the spots referring to c_j
7. **highlighting**	True *iff* at least one mention of c_j is highlighted in bold or italic
8. **average lengths**	average term- and character-based length of spots referring to c_j
9. **idf**	maximum Wikipedia inverse document frequency among the spots referring to c_j
10. **tf-idf**	maximum document spot frequency multiplied by *idf* among the spots referring to c_j
11. **is title**	True *iff* at least one mention of c_j is present in the document title
12. **link probabilities**	maximum and average *link probabilities* of the spots referring to c_j
13. **is name/person**	True *iff* at least one mention of c_j is a common/person name (based on Yago – http://goo.gl/glfBYN)
14. **entity frequency**	total number of spots referring to c_j
15. **distinct mentions**	number of distinct mentions referring to c_j
16. **not ambiguity**	True *iff* at least one mention of c_j for which c_j is the only candidate entity
17. **ambiguity**	minimum, maximum and average ambiguity of the spots referring to c_j; spot ambiguity is defined as 1 minus the reciprocal of the number of candidate entities for the spot
18. **commonness**	maximum and average *commonness* of the spots referring to c_j
19. **max commonness × max link probability**	maximum *commonness* multiplied by the maximum *link probability* among the spots referring to c_j
20. **entity degree**	in-degree, out-degree and (undirected) degree of c_j in the Wikipedia citation graph
21. **entity degree × max commonness**	maximum *commonness* among the spots of c_j multiplied by the degree of c_j
22. **document length**	number of characters in D

Note that some of the features (2–4) explicitly refer to a semi-structure present

in the dataset, with separate fields for different sections of each document. We exploited this semi-structure by distinguishing among spots occurring in the title of the document, in the first/last three sentences, and in the middle sentences. These features are aimed at exploiting information provided by the document structure.

Heavy features. These features are extracted for each candidate entity $c_j \in C'_D = \phi(C_D)$ to model the relationships among c_j and all the other entities in C'_D. To compute these features, specific subgraphs of Wikipedia graph are considered. Let $WG_D = (V_D, A_D)$ be one of such subgraphs, where both the set of vertices V_D and the set of arcs A_D can be defined in different ways:

Vertices V_D: the entities, i.e., Wikipedia nodes, identified by C'_D are extended with their neighborhoods in the Wikipedia graph. Two sets of vertices are exploited, denoted by V_D^0 and V_D^1: *i)* V_D^0 is simply equal to C'_D, as identified by our filtering step; *ii)* V_D^1 contains the vertices in V_D^0 extended with the entities associated with the Wikipedia pages that *link to* or are *linked by* entities in V_D^0.

Arcs A_D: three types of directed arcs are investigated: *i)* all the hyperlinks in Wikipedia between entities in V_D, considered as directed unweighted arcs. Therefore, we have two different sets of arcs, $A_D^0 \subset A_D^1$, one for each set of vertex sets $V_D^0 \subset V_D^1$; *ii)* the arcs derived from the Wikipedia hyperlinks, weighted by the Milne and Witten relatedness function [116], by pruning arcs whose relatedness is zero; *iii)* a weighted and undirected clique graph (i.e., each node is connected to each other), where edges are weighted by the Milne and Witten relatedness function. Also in this case, there are two sets of arcs $A_D^0 \subset A_D^1$. Finally, arcs with a weight below the median are discarded in order to preserve only the most important ones.

Heavy features, listed in Table 4.3, are computed on the 6 graphs resulting by the combination of the two vertex sets on the three edge sets described above. In total, each candidate entity is represented by a vector of 39 *light* features and 99 *heavy* features (16 features WG_D dependent times the 6 graphs, 2 from the TAGME-like scores and 1 the confidence score of the candidate pruning classifier at step 1).

It is worth remarking that the sets of vertices of WG_D (V_D^0 or V_D^1) are small enough to make the computation of these graph features feasible. This is due to the pruning capability of our first pruning step, which greatly reduces the size of the set of candidate entities.

Table 4.3: Heavy features for Supervised Saliency Linking: most features are global and depend on the structure of the graph WG_D, others are specific for an entity

1. **graph size**	number of entities in WG_D
2. **graph diameter**	the diameter of WG_D
3. **node degree**	degree of given entity e in the undirected version of graph WG_D
4. **node average/median in-degree**	average and median node in-degree of WG_D
5. **node average/median out-degree**	average and median node out-degree of WG_D
6. **node average/median in-out-degree**	average and median node degree in the undirected version of graph WG_D
7. **farness**	the sum of the shortest paths lengths between entity e and all the other nodes in WG_D
8. **closeness**	the inverse of farness
9. **eigenvector centrality**	a measure of influence of a node in a network ([60])
10. **random walk**	the probability for a random walker to be at node e while visiting WG_D
11. **personalized random walk**	same as random walk, with a preference vector given by the entity frequencies in D
12. **graph cliques**	number of cliques in WG_D
13. **cross-cliques centrality**	a measure of connectivity of a node e in WG_D
14. **TAGME-like voting schema**	for each $e \in V_D$, we propose two normalizations of the TAGME-like voting schema: $\sum_{e' \in V_D \setminus \{e\}} \frac{Max\_comm(e') \cdot rel(e,e')}{Max\_ambig(e')}$ $\sum_{e' \in V_D \setminus \{e\}} \frac{Max\_comm(e') \cdot rel(e,e')}{\lvert V_D \rvert}$ where $rel(e, e')$ is the Milne and Witten relatedness function, whereas $Max\_ambig(e')$ and $Max\_comm(e')$ are defined in Table 4.2 (sections 16-17). Feature not dependent from WG_D.

4.4 Experiments

4.4.1 Datasets

For the evaluation of EL performance we used the Test B part of the AIDA-CoNLL 2003 dataset [78]. This dataset contains a subset of news from Reuters Corpus V1 which were manually linked to Wikipedia entities starting from candidates generated by the spotter of Aida [78]. The CoNLL dataset is composed of 231 documents with an average of 10.94 entities per document, hence resulting in $\approx 2,500$ mention to

entities. Note that entities are not annotated with a saliency score. There exist other similar datasets such as the Knowledge Base Population track held by NIST Text Analysis Conference. However, the task is quite different as it requires annotating a given single mention in contrast to linking the full document, and it is released only with paid membership (free for the track participants).

In order to evaluate SE prediction performance, a human-assessed dataset of news was created and made publicly available, by relying on the Wikinews project[1]. Wikinews promotes the idea of participatory journalism, and provides a user-contributed repository of news. We chose this source for two main reasons: first, it is *open domain*, thus allowing us to redistribute the annotated dataset without the copyright constraints that affects similar datasets; second, because the news in Wikinews are already manually linked to entities of Wikipedia, thus making the dataset independent from the specific EL system used to detect entities. Due to some subjectivity in the assignment of a saliency score, each document (and thus also its entities) was annotated by multiple annotators, averaging the saliency scores.

An English dump of Wikinews containing news published from November 2004 to June 2014 was used, and the news that users linked to less than 10 or to more than 25 entities were filtered out. In addition, special news pages (e.g., News Briefs, or Wikinews shorts) were removed, as well as news longer than 2500 characters. The resulting dataset contains 604 news articles, uniform in text length and number of linked entities, each one with *title* and *body* fields.

Crowdflower[2], a crowd-sourcing platform, was then exploited for annotating linked entities with saliency scores. In order to get reliable human annotations, a *golden dataset* was created by asking to 4 expert annotators to provide entity saliency scores in a specific subset of 62 documents. These annotations were collected using Elianto [163], an ad-hoc solution developed explicitly for accounting this problem and facilitating the creation of human assessed datasets with both entities and saliency. A detailed description of this framework can be found in Section 4.6. Then, the Crowdflower quality control mechanisms allowed to use the golden dataset produced by the expert annotators to detect and ban malicious annotators. With a reward of 0.35$ per document, 400 documents (including the golden subset) were annotated by at least 3 different Crowdflower annotators in one week. Finally, documents where the annotators exhibited a low agreement were removed, obtaining the final Wikinews

[1] http://en.wikinews.org
[2] http://www.crowdflower.com

Table 4.4: Agreement between groups of Expert (Exp) or Crowdflower (CF) annotators.

Annotators	Docs	Kendall's τ	Fleiss' κ	Kendall's τ *binary*	Fleiss' κ *binary*
CF *vs* CF	329	$0.54_{\pm.03}$	$0.33_{\pm.03}$	$0.68_{\pm.08}$	$0.49_{\pm.10}$
Exp *vs* Exp	62	$0.67_{\pm.11}$	$0.44_{\pm.14}$	$0.72_{\pm.03}$	$0.66_{\pm.04}$
CF *vs* Exp	62	$0.40_{\pm.06}$	$0.19_{\pm.03}$	$0.48_{\pm.09}$	$0.40_{\pm.08}$

dataset, consisting of 365 annotated documents having an average of 12.02 entities per document, hence resulting in $\approx 4,400$ mentions to entities.

To evaluate the quality of the annotations we measured the Crowdflower annotators agreement with Fleiss' κ and Kendall's τ coefficients. The latter was measured by considering the ranked lists obtained by sorting the entities by the saliency label provided by the users. As reported in Table 4.4, we have $\kappa = 0.33_{\pm.03}$ and $\tau = 0.54_{\pm.03}$ among CrowdFlower users. The Fleiss' κ value suggests a *fair* agreement. This is due to the highly subjectivity of the task: different users may give different rates based on their experience, culture, etc. Our agreement results are however consistent with those reported in similar works [16]. Nevertheless, the Kendall's τ coefficient suggests a *good* ranking agreement. We also investigated agreement by collapsing *Highly Relevant* and *Partially Relevant* thus achieving a *binary* labeling. The agreement on such binary formulation is consistently higher, with $\kappa = 0.68_{\pm.08}$ and $\tau = 0.49_{\pm.10}$. This suggests that users agree in identifying *Top Relevant* entities, and they have slightly less agreement in discriminating between different degrees of relevance. Good agreement values were achieved also when comparing Crowdflower users with expert users.

Finally, the different saliency labels provided by annotators were aggregated in order to have one unique saliency label per entity. The aggregation was achieved by averaging the annotators labels and by rounding the average value when a sharp classification is needed. The Wikinews dataset is publicly available and can be downloaded at the address `http://dexter.isti.cnr.it/`. Comparing with other datasets, we believe the annotations it provide are of high quality since it is not biased by users' queries to a search engine as in [127], and it does not rely on the naïve assumption, as in [59], that entities occurring in news abstract are salient while others are not salient.

Table 4.5 reports some statistics about the two dataset used in our experiments. Note that only 10% of the entities annotated in the Wikinews dataset are considered as *Top Relevant*. This suggests the importance of being able to detect the most

Table 4.5: Datasets description and spotting results.

	CoNLL	Wikinews
Documents	231	365
avg. $\lvert\mathbb{E}_D\rvert$	10.94	12.02
Top Relevant	—	436 (10%)
Highly Relevant	—	1685 (38%)
Partially Relevant	—	2261 (52%)
avg. $\lvert C_D\rvert$	549.54	790.05
avg. Max Rec $= \frac{\lvert C_D \cap \mathbb{E}_D\rvert}{\lvert\mathbb{E}_D\rvert}$	0.907	0.925

salient entities in a document. We also report some statistics about the results of the Wikipedia-based spotter. The average number of candidate entities generated per document ranges between 500 and 800, corresponding to an average number of per-entity candidates of about 50 and 66 for the CoNLL and Wikinews datasets, respectively. These figures give a rough idea of the complexity of the disambiguation step. Altough the two datasets contain collectively ≈ 600 documents, they also contain a large number of mentions to entities, $\approx$ 6,900, which are essential in the creation and evaluation of the model, since the two phases are done on a per-entity basis.

The evaluation of the two steps of the *SEL* algorithms were carried out using *5-fold cross-validation* and averaging the results.

4.4.2 Candidate Pruning Step

For each document D, a set of candidate entities C_D was generated with a dictionary based spotter, which exploits the Wikipedia anchors' text and article titles. This preliminary step generates an average of 549.54 and 790.05 candidate entities C_D for the CoNLL and Wikinews datasets respectively, as illustrated in Table 4.5.

To prepare the training set for a classifier used to prune C_D, a *positive* class label was associated to entities in $C_D \cup \mathbb{E}_D$, and a *negative* one to entities in $C_D \setminus \mathbb{E}_D$. It is worth remarking the *highly skewed* class imbalance. Indeed only 2% of $\lvert C_D\rvert$ are positive on CoNLL and 1.5% on Wikinews (see the corresponding sizes of $\mathbb{E}_D$ in Table 4.5).

An interesting information reported in Table 4.5 is the maximal recall achievable for the EL task, averaged over the set of documents in the given collection. This is smaller than 100% because a few positive entities in $\mathbb{E}_D$ were not detected by the spotter, that is $\mathcal{E}_D \cap C_D \neq \mathcal{E}_D$. This depends on the human annotation: in these cases

Table 4.6: Recall-oriented spotting performance.

	CoNLL			Wikinews		
	Rec	Prec	$\lvert C'_D \rvert$	Rec	Prec	$\lvert C'_D \rvert$
GBDT-$\mathbb{F}_l$	0.63	0.76	8.9	0.66	0.76	11.6
GBDT$_\omega$-$\mathbb{F}_l$	0.85	0.39	27.1	0.87	0.37	31.9
GBDT$_\omega$-$\mathbb{S}_l$	0.85	0.37	28.2	0.87	0.36	33.1
HP$_W$	0.85	0.09	124.1	0.90	0.08	169.5

annotators were able to recognize an entity in $\mathbb{K}B$ even if its mention in D is different from all the ones used in the $\mathbb{K}B$ and stored in our dictionary.

Table 4.6 shows the performance of the various pruning methods producing $C'_D = \phi(C_D)$. Note the column $|C'_D|$, which reports the *average* number of entities obtained after the pruning step, and compares its size with the original size $|C_D|$, reported in Table 4.5. The table also shows the Recall/Precision of the various methods in detecting the positive instances, i.e., the entities of C_D that are in $\mathbb{E}_D$.

In particular, Table 4.6 compares the heuristic pruning strategy HP$_W$ with the proposed supervised method. Indeed, the *Candidate Pruning* step adopts a state-of-the-art classification algorithm, the *Gradient Boosting Decision Tree* (GBDT) provided by the `scikit-learn` python library for machine learning. GBDT is trained on the light set of features $\mathbb{F}_l$. We denote this classifier by GBDT-$\mathbb{F}_l$.

Unfortunately, due to the severe class imbalance in the training set, the recall of GBDT-$\mathbb{F}_l$ is significantly worse than the baseline HP$_W$. This means that the classifier prunes too many positive entities. As expected, the precision of GBDT-$\mathbb{F}_l$ is better than the one obtained by HP$_W$, but its global performance is not satisfying. It is worth remarking that different settings of HP, not reported here, did not exhibit better performance in terms of precision.

We mitigated the issue of class imbalance by a re-balancing weight strategy, which re-weights the samples in the empirical objective function being optimized by the classifier. The weight given to each sample is inversely proportional to the frequency of its class in the training set. We denote by GBDT$_\omega$-$\mathbb{F}_l$ this new trained classifier, whose performance is very good. Its recall is similar to the one obtained by HP$_W$, but its precision is remarkably higher. By comparing the number of pruned candidate entities (column $|C'_D|$) with the non-pruned ones ($|C_D$—), the superior pruning power of the proposed method over HP$_W$ becomes apparent. Our supervised method is in

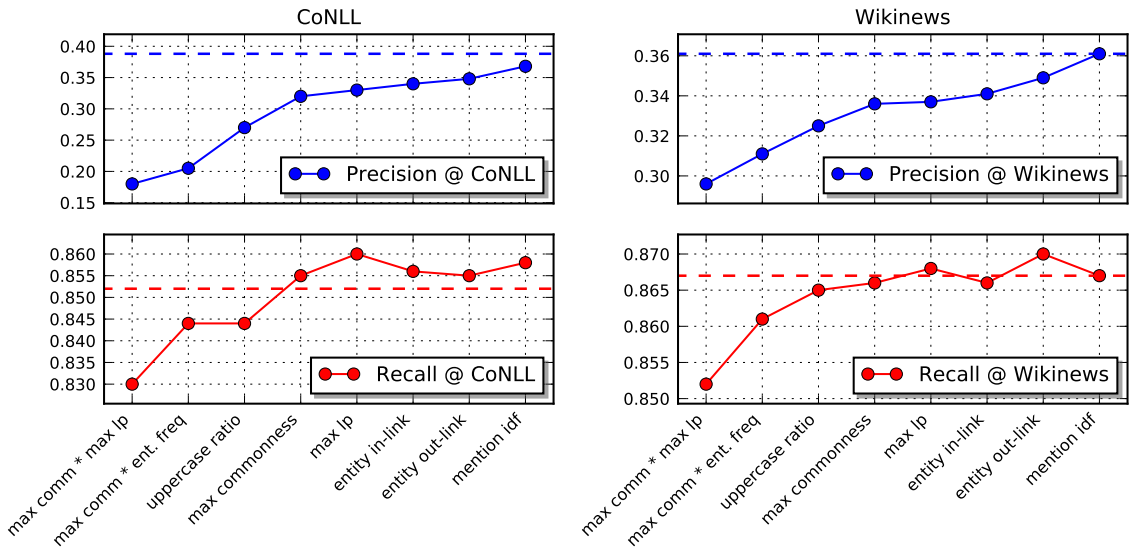

Figure 4.1: Incremental performance on step 1 using top k features.

fact able to prune $\approx 95\%$ of the initial set of candidates C_D, without hindering the recall.

The adopted GBDT implementation provides a standard measure of features' importance according to their contribution in optimizing the decision tree accuracy. We thus performed feature selection by considering the features sorted by importance, and trained a different classifier with the *top-k* features. Figure 4.1 shows the performance on the CoNLL dataset obtained by varying k up to the best 8 features. We denote this small set of top-8 features by $\mathbb{S}_l$. Note that the most important features are combinations of link probability, commonness, and entity frequency in Wikipedia. The performance of the classifier improves when we add further features. In fact, the performance of our GBDT_ω-$\mathbb{S}_l$ classifier which employs the top-8 features, turned out to be very similar to the one of the classifier that employs the full set $\mathbb{F}_l$ (dashed line). This can also be observed by considering Table 4.6, where the performance of GBDT_ω-$\mathbb{S}_l$ is reported for both CoNLL and Wikinews.

We conclude that the GBDT_ω-$\mathbb{S}_l$ classifier provides the best performance on average for the two datasets, and that the *light* feature set $\mathbb{F}_l$ provides sufficient quality. Indeed, a smaller set of eight light features $\mathbb{S}_l$ suffices to train an effective classifier GBDT_ω-$\mathbb{S}_l$, which is able to strongly prune the set of candidate entities, thus making feasible the subsequent step which needs to extract expensive graph-based features for each of these candidate entities.

4.4.3 Saliency Linking Step

In the second step, disambiguation and saliency prediction were performed by training a new model on the filtered set of candidates C'_D. In this case, the full feature set $\mathbb{F}$ was considered, including also an additional feature given by the confidence score of the candidate pruning classifier at step 1. The graph-based features are expensive to compute, but given the reduced number of entities per document, the computation is affordable.

In order to use the same model for both EL and SE tasks, we adopted a state-of-the-art regression algorithm, the *Gradient Boosting Regression Tree* (GBRT), again provided by the `scikit-learn` library, trained on the full set of features $\mathbb{F}$. The resulting model is denoted by GBRT-$\mathbb{F}$. A threshold was learnt on the training set by optimizing the F_1 measure, and then used to filter out not relevant entities, i.e., having a score smaller than the learnt threshold. The same linear search process was used for learning a filtering threshold on the confidence score for the competitors algorithms simply solving the EL problem.

To prove the benefits of the proposed two-steps algorithm, a regressor model trained on the original set of candidate entities C_D to predict the entity saliency (namely 1-Step GBRT-$\mathbb{F}_l$) was trained. This model exploited the light features $\mathbb{F}_l$ only, due to the high number of candidate entities, for which it was impossible to compute the heavy features.

The accuracy of the EL task was first analyzed by measuring precision, recall and F_1 score on the set of returned entities. The precision was also measured considering only the top-3 entities returned by the model, sorted by the annotation confidence for state-of-the-art algorithms or by the predicted score for our regression models. Note that, given the nature of the EL task, we are only interested in predicting relevant vs. irrelevant entities, resulting in the training of a binary model. Regarding the multi-class Wikinews dataset, all the positive scores were collapsed into a single relevant score. The distribution of positive and negative classes in $C'_D = \phi(C_D)$ became much more balanced after the pruning phase compared to the previous step (with a proportion of 35% / 65% respectively). Table 4.7 reports the EL performance for the various methods. In particular, state-of-the-art algorithms were compared with the proposed supervised method. The publicly available annotation service was used for each competitor algorithm except Wikifier, for which its available source code was used, with the best performing settings reported in the paper by the authors. The first two rows report the performance of the unbalanced model vs. the balanced one: since the dataset is only slightly unbalanced, they perform very similarly.

Table 4.7: Entity linking performance.

	CoNLL				Wikinews			
	Rec	Prec	F_1	P@3	Rec	Prec	F_1	P@3
GBRT-$\mathbb{F}$	0.76	0.71	0.72	0.82	0.75	0.72	0.72	0.87
GBRT$_\omega$-$\mathbb{F}$	0.73	0.74	0.72	0.81	0.75	0.72	0.72	0.87
GBRT-$\mathbb{S}_u$	0.71	0.71	0.69	0.80	0.76	0.70	0.71	0.86
GBRT$_\omega$-$\mathbb{S}_u$	0.70	0.72	0.69	0.80	0.73	0.74	0.71	0.86
Aida	0.76	0.72	0.73	0.82	0.66	0.73	0.68	0.80
Tagme	0.68	0.59	0.61	0.74	0.77	0.67	0.70	0.85
Wikiminer	0.55	0.43	0.46	0.65	0.78	0.53	0.62	0.87
Wikifier	0.52	0.33	0.36	0.43	0.41	0.34	0.36	0.35
Spotlight	0.48	0.30	0.32	0.46	0.56	0.31	0.38	0.54
1-Step GBRT-$\mathbb{F}_l$	0.69	0.69	0.67	0.81	0.70	0.73	0.69	0.86

Also for this study, a subset of the top-10 most important features, denote as $\mathbb{S}_u$, was selected. The models trained using only this subset of features are GBRT-$\mathbb{S}_u$ and GBRT$_\omega$-$\mathbb{S}_u$, with the latter denoting the model that adopts the class imbalance solution. The two models perform very similarly each other, and only slightly worse (-4% on F1 on CoNLL and -1% on Wikinews) than the models that uses all the features. Figure 4.2 reports the incremental F1 scores obtained by using this subset of features over the two datasets. It is worth noting that the top-2 features of this subset suffice to obtain performance higher then most state-of-the-art solutions. The most important features belong to different *families* of categories. We have some mention-based features (e.g., uppercase ratio or position first mention), some graph related features (e.g., eigenvector and Tagme-like) as well as features coming from the Wikipedia graph (e.g., entity degree) and the confidence score of the *Candidate Pruning* binary classifier.

The performance of the proposed solution were compared against state-of-the-art methods Aida, Spotight, Tagme, Wikiminer and Wikifier 2.0. The proposed full machine-learnt model obtained similar or even better performance when compared to the best performing algorithm on CoNLL (Aida) and Wikinews (Tagme), with an F1 of 0.72 on both the datasets. Indeed on Wikinews *SEL* exhibits +3% improvement on F1 compared to Tagme and +6% compared to Aida, while on CoNLL it performs only slightly worse than Aida (-1%) but it outperforms Tagme ($+18\%$). It is worth noting that CoNLL dataset was created by using the Aida spotter, thus giving Aida

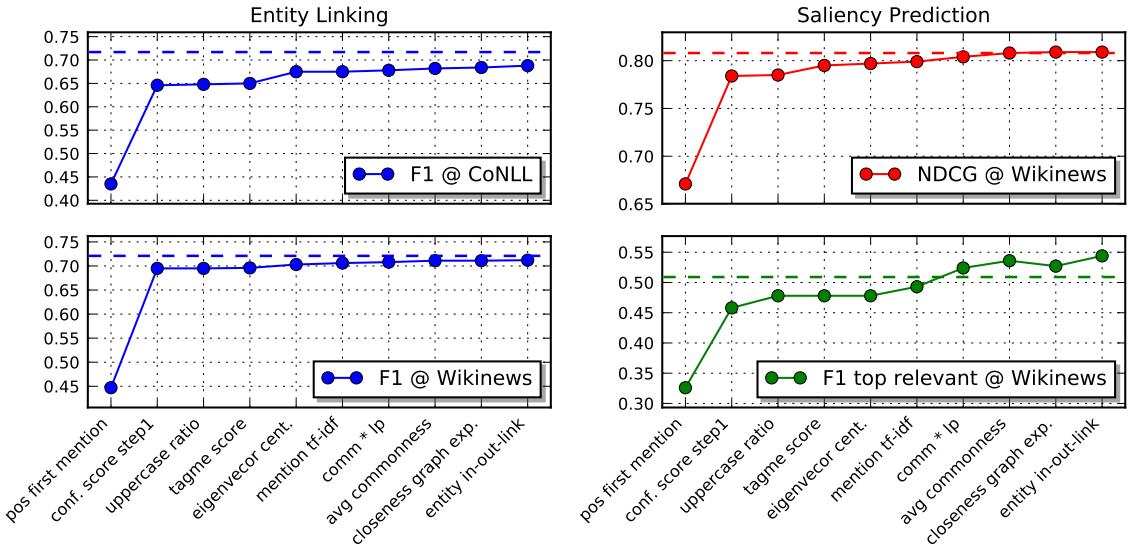

Figure 4.2: Incremental performance on step 2 using top k features.

an implicit advantage. Another interesting result is that it exhibits well balanced precision and recall values on both the datasets, while state-of-the-art competitors do not show a similar positive behaviour. Indeed, the proposed method shows the best performance on average across the two datasets for every measure adopted when using the full set of features, and it notably provides the best P@3 on average when using the feature set $\mathbb{S}_u$ only. Finally, some considerations about the 1-Step algorithm: despite its good performance, the method always performs worse than GBRT-$\mathbb{F}$ and GBRT-$\mathbb{S}_u$. It is worth noting that this single step algorithm provides EL annotations comparable or even better than most state-of-the-art algorithms. This confirms that entity saliency plays an important role as it also boosts entity linking methods. It is apparent that annotation confidence cannot approximate saliency.

Table 4.8 shows the saliency performance of the trained models. In this case the regressor makes use of all the saliency labels. For this experiment we used only the Wikinews dataset, since CoNLL is not annotated with the saliency. The performance on predicting the saliency was evaluated by using: i) the NDCG considering the entities sorted by saliency, in order to know how good is the function in ranking the entities by saliency, ii) Precision, Recall and F_1, considering only the most important entities, in order to know how good is our machine-learnt model in identifying the set of the *Top Relevant* entities (denoted as P^{top}, R^{top} and F_1^{top}). NDCG was measured on the set of entities selected by optimizing F_1 (as above), sorted by saliency/confidence score, whereas F_1^{top} is measured after optimizing a filtering threshold on the training data. To this purpose, the 61 documents without any *Top Relevant* entities has been

Table 4.8: Saliency prediction performance on Wikinews.

	NDCG	Rec^{top}	Prec^{top}	F1^{top}
GBRT-$\mathbb{F}$	0.82	0.50	0.46	0.43
GBRT_{ω}-$\mathbb{F}$	0.81	0.56	0.50	0.49
GBRT_{ω}-$\mathbb{S}_u$	0.81	0.61	0.50	0.52
Aida	0.58	0.71	0.12	0.19
Tagme	0.65	0.54	0.16	0.22
Wikiminer	0.64	0.37	0.14	0.19
Wikifier	0.32	0.66	0.06	0.11
Spotlight	0.47	0.40	0.08	0.12
1-Step GBRT-$\mathbb{F}_l$	0.73	0.56	0.36	0.41

discarded by the evaluation, so as to avoid misleading results. It is worth recalling that state-of-the-art algorithms do not provide saliency scores, so we used the confidence scores as an indicator of how related are the entities to the document.

We observe that in this setting, the weighted model performs better than the unweighted one, since the distribution of the positive labels is not uniform. Moreover, the model that makes use of only the subset $\mathbb{S}_u$ of features has similar or even better performance with respect to the model with all the features. As reported, *SEL* significantly outperforms the best performing state-of-the-art algorithm (Tagme) both in terms of NDCG and F_1^{top} with a relative improvement of +25% and +137% respectively. Furthermore, Figure 4.2 reports the incremental F_1^{top} and NDCG scores obtained by using the subset $\mathbb{S}_u$ of features over the Wikinews dataset. It is worth noting that the model trained using only the top-7 features obtains performance similar to that of the full feature set $\mathbb{F}$, and by using all the top-10 features the model performs even better, with a +6% improvement in terms of F_1^{top}.

We conclude that the recall-oriented pruning of the spotting results, along with the additional features extracted in the second step, provide a significant improvement over the 1-Step approach, with a substantial performance gap between the two models.

4.5 Summarization

Automatic Text Summarization is a powerful Text Mining technology that can rapidly digest and skim textual contents. Automatic summarizers are nowadays indispensable

for dealing with increasing online data in a wide range of application domains [104]. For instance, in web search, summaries –called *snippets*– are automatically built and attached to search engine hits. Automatic summarizers are also employed prominently for creating summaries of news stories, medical texts or biographical articles, just to name a few.

Extractive summarizers apply different methods to select salient parts of the source text. For example, cue words, position within the text, or centrality (estimated as the similarity to the centroid of the text) have been exploited for detecting salient extracts. Sentence-based summarizers identify the most important sentences in the source text and arrange them in some effective way. This involves three steps, namely: feature-based representation of sentences, sentence scoring, and summary creation by selecting sentences [120]. The first step often resorts to simplified representations of the sentences (e.g. bag of words and frequency-based weighting mechanisms), and content-based scores that estimate how central the sentence's words are. Other typical shallow features are location-based features. For instance, salient sentences tend to occur in certain specifiable positions within the text.

The saliency detection algorithm, which was described in the previous sections of this paper, is an effective solution for ranking the entities mentioned in a given text. This entity-based feature can be incorporated into text summarizers and exploited to extract salient sentences from text. This section is a report on our endeavors and experiments related to injecting entity-based features into standard text summarizers. Entities are core components of texts and they provide a great deal of information about the topics of the source texts. The most informative sentences might exhibit singular patterns of usage of entities and it might be the case that standard summarization features are unable to identify such patterns. In this section we try to shed light on these issues. More specifically, we define here new entity-based sentence features for extractive summarization. These features are computed with *SEL* and then combined with standard sentence summarization features (position, centroid and length). This leads to a sentence scoring method that aggregates multiple types of evidence. Next, we proceed to inject this sentence scoring method into a well-known summarization system that creates non-redundant summaries of the desired size. We perform single-document and multi-document summarization experiments and we analyze the effects of the newly-derived features.

4.5.1 Summarization Approach

We estimate sentence importance by combining multiple types of evidence. For each candidate sentence, standard features, such as the position of the sentence in the source text, or the content-based similarity between the sentence and the document's centroid, are combined with entity-based features. First, we briefly describe some standard sentence features and the main components of the summarization system. Next, we present the new entity-based sentence features.

In many summarization cases, the sentences appearing at the beginning of a document provide much information about the topics of the document. Therefore, standard summarizers often weight the leading sentences more heavily. Centroid similarity is another standard feature commonly employed in summarization. This works as follows. Using standard statistics, a centroid is computed for each document to be summarized (e.g., a vector of tf-idf weights). This centroid tries to capture which words are central in the document. Following a similar approach, we obtain a weight-based representation for each candidate sentence. Finally, a similarity score (e.g. cosine similarity) between the weighted representation of the centroid and the weighted representation of the sentence is computed. This content-based matching approach favors sentences whose overall resemblance to the whole document is high.

MEAD [133] is a popular system that supports a variety of summarization strategies. It provides the implementation of effective baseline summarizers and, additionally, it has a flexible and modular architecture that permits to incorporate your own sentence features. MEAD supports single-document summarization (the input is a single document) and multi-document summarization (the input is a cluster of documents). The following built-in features are automatically computed by MEAD and associated with each sentence of the document or cluster to be summarized[3]: Position, Centroid and Length. Position represents the position of the sentence in the document(s)[4]. Centroid is computed as the cosine overlap of the sentence with the centroid of the document (or cluster). Length is regarded as a cutoff feature: sentences whose length is below a given threshold are discarded. MEAD's aggregation module is based on linearly combining all feature weights and building a ranking of sentences by decreasing aggregated scores. This is an example of MEAD's sentence scoring approach for a summarizer that incorporates the three standard features:

[3] All features range from 0 to 1.

[4] The first sentence gets a weight equal to 1 and the remaining sentences are assigned linearly decreasing weights.

$$score(sen) = \begin{cases} w_{cen} \cdot cen(sen) + w_{pos} \cdot pos(sen) & \text{if } len(sen) \geq thr_{len} \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

$cen(sen)$, $pos(sen)$ and $len(sen)$ are the values of Centroid, Position and Length for the sentence sen to be scored; w_{cen} and w_{pos} are the weights of the summarizer for Centroid and Position, and thr_{len} is the threshold for Length.

All sentences in the document (or cluster) are scored using this formula and a ranking by decreasing $score(sen)$ is built. Next, this initial ranking of sentences is re-ranked by a redundancy removal module. This module downgrades sentences that are too similar to sentences ranked above. In MEAD, the redundancy removal re-ranker offers a more diverse collection of sentences by implementing Maximal Marginal Relevance (MMR). A full description of MMR can be found in [31]. Finally, the resulting ranking of sentences is employed to produce a summary of the desired size.

The standard sentence-based features described above have been enriched with several entity-based features, as to exploit the benefits of incorporating entity-derived information into text summarizers. We obtained these features by annotating each document independently from the others[5], and using the models trained on Wikinews for predicting the saliency of the linked entity. For single-document summarization we incorporated the following entity-based features:

- **SumSalMaxNorm**: sum of the predicted saliency of the entities annotated in each sentence. This sum was normalized into $[0, 1]$ by dividing by the maximum sum (computed across all sentences in the document).
- **SumSalLenNorm**: same as SumSalMaxNorm, but before normalizing by the maximum sentence score, a prior normalization is done by sentence length (so as to mitigate the advantage of long sentences above shorter ones).

For multi-document summarization we incorporated the following entity-based features:

- **SumAggSalMaxNorm**: the saliency score of each entity among the different documents is summed. This aggregation of scores leads to an overall estimation of entity saliency. This aggregated score is then used for summing the contribution of each entity to the sentence score, as described for the single-document feature SumSalMaxNorm. Finally, the sentence scores are normalized by their maximum score.

[5]This also holds for multi-document summarization.

- **SumAggSalLenNorm**: same as SumAggSalMaxNorm, but adopting the prior normalization approach as described in SumSalLenNorm (i.e., by sentence length).
- **MaxAggSalLenNorm**: same as **SumAggSalMaxNorm**, but aggregating the entity saliency as the max of their predicted saliency.
- **TopSalientRelScoresMaxNorm**: using the top 3 salient entities of each document in the cluster, identify a subset of entities acting like a centroid. Then, sum the contribution of each entity to the sentences where it appears in as the average relatedness between this entity and all the entities in the centroid set. The measure adopted for computing this similarity is the Milne and Witten *relatedness* [116]. Finally, normalize the sentence scores by their maximum.
- **TopSalientRelScoresLenNorm**: As TopSalientRelScoresMaxNorm, but adopting the prior normalization approach as described in SumSalLenNorm (i.e., by sentence length).

We proposed also a slight variant of most of these features, identified by the postfix '_s2', where the contribution given by each entity is computed as the square of its predicted saliency. The main idea behind this variant is to give a boost to sentences containing top salient entities.

4.5.2 Summarization Experiments

We worked with several collections created under the Document Understanding Conference (DUC)[6]. We performed the following generic summarization tasks: i) single-document summarization (automatic summarization of a single news article), and ii) multi-document summarization (fully automatic summarization of multiple news articles on a single topic). Table 4.9 reports the main statistics of the collections and how we used them for training and testing. All documents are news articles obtained from the Text Retrieval Conference (TREC) and the average number of sentences per document is about 27.

The training step consisted only of learning the weights assigned to the new sentence features. We did not adapt the entity-based saliency estimation to the characteristics of these collections (we simply used the configuration learned from Wikinews).

[6] `http://duc.nist.gov`.

Table 4.9: Summarization collections used in our experiments

	Single-document summarization		
	DUC2001T	DUC2001	DUC2002
# documents	298	308	534
required summarization length	100 words	100 words	100 words
train/test	train	test	test

	Multi-document summarization		
	DUC2001MT	DUC2001M	DUC2002M
# clusters	30	29	116
avg # documents per cluster	9.97	10.17	9.59
required summarization length	100 words	100 words	100 words
train/test	train	test	test

Following existing practice, we evaluated the summarizers using ROUGE measures [92]. This is a class of measures that automatically determine the quality of an automatic summary by comparing it to summaries created by humans (the DUC collections provide us with *manual* summaries for all documents and clusters). ROUGE measures count the number of overlapping units (e.g., n-grams) between the automatic summary and the manual summary. ROUGE-2 and ROUGE-SU4 are two widely adopted ROUGE measures. ROUGE-2 is focused on counting bigram overlapping. ROUGE-SU4 counts overlapping of unigrams and *skip-bigrams* (bigram overlapping allowing for gaps with maximum length of 4).

We experimented with the following summarization methods:

- **standard MEAD**. This is the default MEAD configuration based on centroid, position and length. The default feature weights are 1, 1, and 9, respectively (meaning that sentences with less than 9 words are discarded and the remaining sentences are assigned an aggregated score equal to the sum of the centroid and position scores).
- **lead-based MEAD**. This configuration of MEAD simply extracts the initial sentences of the document or cluster to build the summary.
- **random**. This is a naïve summarizer that randomly extracts sentences from the document or cluster.

- **MEAD + f_e** (where f_e is one of the entity-based features described above). This strategy consists of incorporating the feature f_e into *standard MEAD*. The weights and length threshold of the standard features are fixed to the default values (1, 1, and 9, respectively) and the weight of the new feature (e) is learnt by grid search on the training collection (the weights tested range from -1 to 1 in steps of 0.1). We optimized ROUGE-2. More sophisticated ways to optimize the weights can be implemented (e.g., Particle Swarm Optimisation, which was applied in [96] for creating summarizers that work with dozens of features). However, we work here with a reduced set of features and focus on individually incorporating (and testing) each entity-based features. We leave sophisticated combinations and optimizations as future work.

4.5.3 Results

The experimental results obtained for the test collections are reported in Table 4.10 (single-document summarization) and Table 4.11 (multi-document summarization). The random summarizer performs poorly for both tasks. This is as expected, given its lack of sophistication.

Let us first focus on the results of single-document summarization. The inclusion of entity-based features on the top of standard MEAD led to improved summarizers. As a matter of fact, MEAD + f_e performs better than standard MEAD (for all f_e and for both performance measures). This suggests that the standard summarizer is unable to select sentences with prominent entities, and injecting entity-based features into this standard summarizer helps to create summaries with more salient entities (and more overlapping with gold summaries). For instance, SumSalLenNorm_s2, which is the best performing entity feature for single-document summarization, had assigned a weight of 1 during the training stage (the maximum in the range of our tuning grid: $[-1, 1]$). This means that the resulting summarizer (MEAD + SumSalLenNorm_s2) gives extra weight to sentences with salient entities (on the top of their Centroid or Position scores). The improvements of SumSalLenNorm_s2 over the other entity-based features give also credit to the way in which SumSalLenNorm_s2 mitigates the advantage of long sentences above shorter ones. Still, the overall results of single-document summarization do not give much support to entity-based features. The main reason is that a simple summarizer based on selecting the leading sentences leads to the highest ROUGE-2. Furthermore, the ROUGE-SU4 of MEAD + SumSalLenNorm_s2 is greater than the ROUGE-SU4 of lead-based MEAD but the improvement is tiny and statistically insignificant. The lead-based summarizer is a competitive solution

Table 4.10: Test results (Single-Document Summarization). The performance scores are reported together with 95% confidence intervals (in brackets). For each metric and collection the highest score is shown in boldface.

	ROUGE-2	ROUGE-SU4
DUC2001		
standard MEAD	.1793 (.1660, .1941)	.1813 (.1698, .1926)
random	.1277 (.1167, .1401)	.1420 (.1336, .1517)
lead-based MEAD	**.1931** (.1796, .2071)	.1825 (.1726, .1934)
MEAD + SumSalLenNorm	.1871 (.1735, .2016)	.1842 (.1729, .1955)
MEAD + SumSalLenNorm_s2	.1927 (.1789, .2068)	**.1902** (.1790, .2019)
MEAD + SumSalMaxNorm	.1852 (.1710, .2007)	.1839 (.1723, .1957)
MEAD + SumSalMaxNorm_s2	.1860 (.1719, .2016)	.1852 (.1737, .1971)
DUC2002		
standard MEAD	.1995 (.1912, .2080)	.1928 (.1855, .2000)
random	.1437 (.1357, .1520)	.1506 (.1441, .1573)
lead-based MEAD	**.2067** (.1986, .2154)	.1928 (.1862, .2000)
MEAD + SumSalLenNorm	.2039 (.1953, .2122)	.1976 (.1908, .2049)
MEAD + SumSalLenNorm_s2	.2046 (.1962, .2129)	**.1984** (.1915, .2056)
MEAD + SumSalMaxNorm	.2013 (.1929, .2096)	.1937 (.1866, .2004)
MEAD + SumSalMaxNorm_s2	.2035 (.1950, .2117)	.1965 (.1896, .2033)

for single-document summarization but it is the worst performing approach for multi-document summarization. When summarizing a single news article we can benefit from the style of writing of typical journalists, who express the main ideas first. However, summarizing a cluster of documents is a more difficult task where choosing the leading sentences from the clustered documents is ineffective.

Let us now discuss the results obtained for the multi-document summarization task. Standard MEAD is here the best performing baseline summarizer. It performs subsantially better than both the random summarizer and lead-based MEAD. Again, many entity-based features lead to improvements over standard MEAD; but SumAggSalMaxNorm and SumAggSalMaxNorm_s2 are the most promising features. SumAggSalMaxNorm features score the salient entities within the cluster of documents in an aggregated form. Each entity weight is based on aggregating how salient the entity is in every document of the cluster. This promotes entities that are central to the cluster. The results show that these features produce better multi-document summaries.

Another interesting insight from our experiments is that all *s2* variants are better

Table 4.11: Test results (Multi-Document Summarization). The performance scores are reported together with 95% confidence intervals (in brackets). For each metric and collection the highest score is shown in boldface.

	ROUGE-2	ROUGE-SU4
DUC2001M		
standard MEAD	.0510 (.0374, .0646)	.0828 (.0682, .0986)
random	.0310 (.0213, .0424)	.0645 (.0544, .0747)
lead-based MEAD	.0303 (.0213, .0400)	.0639 (.0548, .0744)
MEAD + SumAggSalLenNorm	.0527 (.0378, .0697)	.0859 (.0714, .1022)
MEAD + SumAggSalLenNorm_s2	.0540 (.0408, .0681)	.0828 (.0683, .0989)
MEAD + SumAggSalMaxNorm	.0604 (.0445, .0775)	.0901 (.0762, .1055)
MEAD + SumAggSalMaxNorm_s2	**.0655** (.0483, .0841)	**.0925** (.0765, .1085)
MEAD + MaxAggSalLenNorm	.0466 (.0327, .0634)	.0790 (.0650, .0955)
MEAD + MaxAggSalLenNorm_s2	.0534 (.0405, .0680)	.0854 (.0715, .1009)
MEAD + TopSalientRelScoresLenNorm	.0510 (.0374, .0646)	.0828 (.0682, .0986)
MEAD + TopSalientRelScoresMaxNorm	.0587 (.0433, .0753)	.0873 (.0737, .1025)
DUC2002M		
standard MEAD	.0684 (.0610, .0769)	.0950 (.0870, .1032)
random	.0355 (.0301, .0413)	.0710 (.0659, .0764)
lead-based MEAD	.0433 (.0369, .0504)	.0659 (.0601, .0716)
MEAD + SumAggSalLenNorm	.0627 (.0554, .0704)	.0940 (.0870, .1012)
MEAD + SumAggSalLenNorm_s2	.0678 (.0596, .0762)	.0965 (.0893, .1037)
MEAD + SumAggSalMaxNorm	**.0708** (.0640, .0784)	.0970 (.0901, .1041)
MEAD + SumAggSalMaxNorm_s2	**.0708** (.0639, .0780)	**.0980** (.0914, .1050)
MEAD + MaxAggSalLenNorm	.0545 (.0483, .0607)	.0854 (.0792, .0920)
MEAD + MaxAggSalLenNorm_s2	.0607 (.0536, .0681)	.0892 (.0827, .0957)
MEAD + TopSalientRelScoresLenNorm	.0685 (.0610, .0769)	.0953 (.0873, .1035)
MEAD + TopSalientRelScoresMaxNorm	.0679 (.0605, .0753)	.0958 (.0890, .1034)

than their respective counterparts. This suggests that summarizers must focus on the top salient entities (rather than on marginally salient entities).

Attacking Text Summarization with entity-based features is a novel and interdisciplinary way of approaching the problem. We have provided preliminary empirical evidence on the effect of these features. Overall, our experiments suggest that entity-based features are meaningful and worth to be considered for Text Summarization. The improvements are modest but we think there is room for further enhancement. Observe that we did not adapt the saliency models to these DUC collections (we simply used the models learned on Wikinews) but, still, the results suggest that *SEL* can lead to improved summarizers (particularly for multi-document summarization).

For single-document summarization, we only found modest improvements on ROUGE-SU4. In the future, we will further experiment with single-document summarization collections and we will try to confirm the effectiveness (or lack of) of entity-based features under different circumstances.

Observe also that this was a preliminary series of experiments and the aim of this evaluation was not to design a state-of-the-art summarizer. This would require combining evidence and features from multiple studies and summarization approaches. Instead, we focused on a well-known summarization system whose modular architecture permits to incorporate new features. These experiments allowed us to draw some initial conclusions about entity-based features in combination with some standard summarization features. But, of course, the role of entity-based features in enhancing state-of-the-art extractive summarizers requires further investigation.

4.6 Elianto - Entity Linking Annotation Tool

Knowing the set of entities mentioned in a document is often not enough for a plethora of Information Extraction tasks. When the linked entities are used to provide some high-level representation of a document, e.g., for clustering purposes, being able to promote salient entities is crucial. Also the evaluation of *Entity Linking* algorithms should takes into account the importance of the entities in evaluating the performance of the system (e.g., an algorithm that fails to annotate a very important entities should be penalized more than an algorithm that fails to annotate a side entity).

Unfortunately, publicly available benchmark datasets that contain accurate supervised knowledge about mentioned entities and their saliency ranking are currently very poor, both in number and quality. The importance of such data is two-fold. On the one hand, as discussed above, they are necessary for a sound comparison of different EL techniques. On the other hand, these datasets can also be used to train machine learning models, in turn used to automatically link and rank entities, as proposed in this Chapter.

For these reasons, we propose ELIANTO[7] (Entity Linking Annotation Tool), an open-source Web framework that crowd-sources the production of publicly available rank-enriched datasets for *Entity Linking* and *Salient Entities* tasks with the goal of involving the research community in producing such datasets.

The framework allows to annotate collections of documents and provides tools for driving multiple users annotations (e.g., minimum number of annotators per

[7]The source code is available at https://github.com/dexter/elianto

document), for inspecting user annotations and for analyzing collection status. Its back-end allows the ingestion of documents collection trough a command line program, supports semi-structured documents and gives the possibility to specify an HTML template describing how documents must be displayed.

In order to simplify the linking task and improve the response time of the tool, ELIANTO pre-compute candidate spots and entities for each document. DEXTER could be used for this task, but it could be replaced with other EL system if needed. The framework offers a Web interface for annotator users. When annotator logs in to the system, an introduction to the Entity Linking task and a guideline explaining how to annotate documents are presented. Thanks to the login mechanism, we can monitor users activity as well as analyzing participation and annotation quality on a per user basis. The annotation of a document is organized in a guided two-step process, as described in the following. Note that, such two-step process implicitly forces the user to evaluate twice the entities she annotated, and correct them if needed.

Step 1: Mention detection and linking

In the first step (Figure 4.3), the document is presented to the annotator on the left side of the page: if the system has candidate mentions for the document, these are displayed in red. If the annotator clicks on a mention the list of the candidates entities is displayed on the right side of the page. The annotator can decide to: i) select one entity from the list ii) add an other candidate entity inserting its Wikipedia url in a form iii) delete the mention if it is wrong or not relevant.

The annotator can also decide to create a new mention just highlighting a piece of text and selecting the option *Create Spot* from the contextual menu. If the annotator generates a new mention the system can automatically provide a list of candidates entities: this is performed calling a REST service that given a mention returns a list of candidate entities. The mentions that the annotator links to entities are highlighted in green, while the currently selected mention is highlighted in yellow. In the contextual menu we provide an option that allows the annotator to extend an annotation to all the occurrences of the same mention in the document. The interface requires to the annotator to annotate (or delete) all the mentions before moving to the step 2. It is worth to note that we also added the possibility to skip a document if the user thinks that it is not worth to annotate e.g., a web document containing only noisy text, or a tweet with no linkable entities.

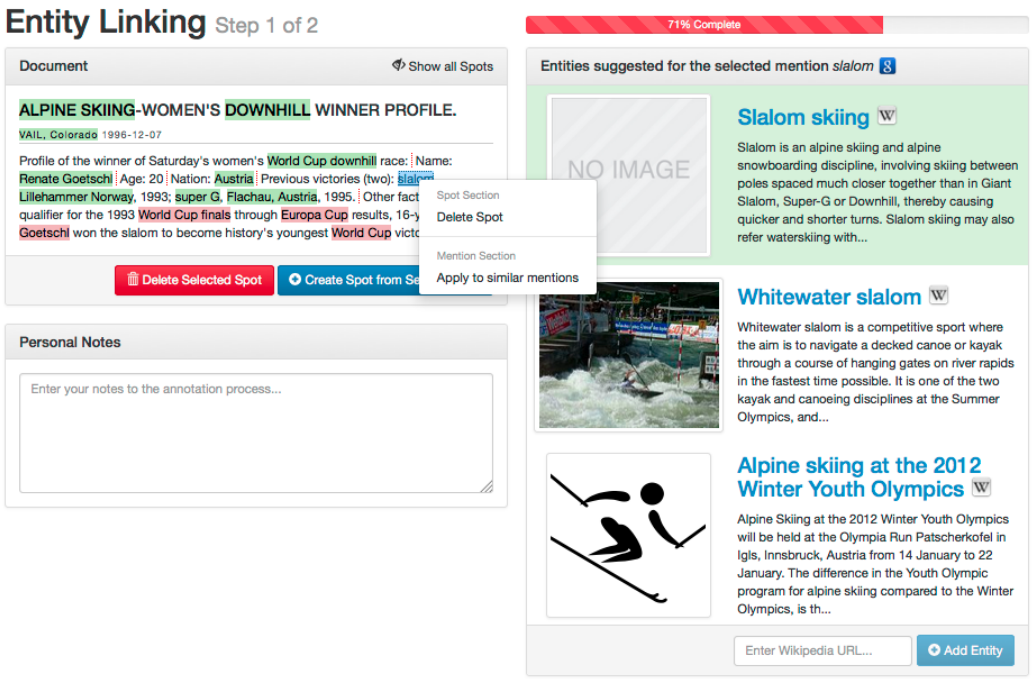

Figure 4.3: ELIANTO step 1: Mention detection and Linking

Step 2: Rank entities by aboutness

In the second step (Figure 4.4), the system still presents the document on the left, and on the right the list of the distinct entities associated to the mentions in the previous step. The annotator is asked to rate the entities that she selected in the previous step, according to how much they are central to the document story. We defined 4 different ratings:

Top Relevant (3 Stars) if the entity tells you what a document is about, i.e., the main topics or the leading characters. We suggested the user to annotate about 3 entities per document in this category;

Highly Relevant (2 Stars) we named them *satellite entities*: they are not essential for understanding the document but provide important facets;

Partially Relevant (1 Star) entities that provide background information about the content of the document;

Not Relevant (0 Star) mentions that the annotator linked to an entity but they are not saying anything about the document.

The system architecture is mainly composed of three layers:

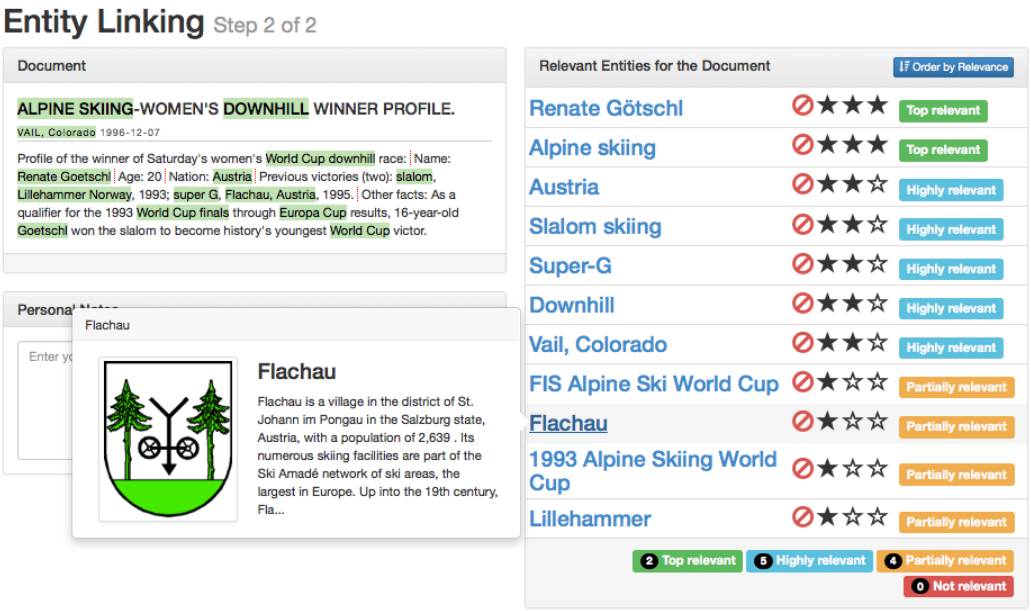

Figure 4.4: ELIANTO step 2: Rank Entities by Saliency

- **The Data Access Object**: allows to store and retrieve the collections and the user annotations, and provides all the object abstractions;
- **The Core**: implements the logic of the application;
- **The Interfaces**: a REST API that allows external applications to retrieve the documents and to submit the user annotations. Some command line programs to perform the indexing and dump the annotations.

We implemented a web interface using the Angular [8] and Bootstrap [9] frameworks, all the actions are performed calling the REST API provided by the ELIANTO server. We also provide a dashboard interface for the user that allows to see the previously annotated documents and to edit them if needed. It is possible to define admin users that can visualize analytics over all the dataset and the user annotations.

4.7 Conclusions

In this chapter we proposed a novel supervised Salient Entity Linking (*SEL*) algorithm that comprehensively addresses Entity Linking and Salient Entities detection problems. Besides improving Entity Linking performance with respect to state-of-the-art competitors, *SEL* predicts also the saliency of the linked entities. The algorithm exploits

[8] https://angularjs.org
[9] http://getbootstrap.com

a two-step machine-learnt process: first a *Candidate Pruning* step aimed at filtering out irrelevant candidate entities is performed, thus obtaining good precision figures without hindering recall; then, a *Saliency Linking* step effectively chooses the entities that are likely to be actually mentioned in the document and predicts their saliency.

The experiments conducted on two different datasets confirmed that the proposed solution outperforms state-of-the-art competitor algorithms in the Entity Linking task. In particular improvements in terms of F_1 of 6% w.r.t. Aida and 18% w.r.t. Tagme were measured. Moreover, *SEL* significantly outperforms the same competitors in the Salient Entities detection task of up to 18% and 139% in terms of NDCG and F_1^{top}, respectively. The latter analysis has been made possible thanks to the creation of a novel dataset of news manually annotated with entities and their saliency, hereinafter publicly available to the research community.

We believe that our comprehensive Entity Linking and Salient Entities detection approach constitutes a remarkable contribution to the field, since entity saliency detection is an important aspect of the whole document annotation pipeline and impacts on information extraction from text in a broader sense.

To experimentally assess this impact on a real use case, we investigated the usage of *SEL* to feed novel text summarization techniques. We thus exploited the entity saliency score predicted by *SEL* to design novel extractive summarizers boosting document sentences mentioning the most salient entities. The experiments conducted on several well-known summarization datasets provided the empirical evidence of the positive effect of including saliency-derived features in the summarization process. In particular we observed improvements in terms of ROUGE-SU4 of up to 5% on single-document datasets and up to 12% on multi-document datasets w.r.t. the Standard MEAD summarizer do not using saliency information. Overall, our results open a plenty of possibilities for solving many information extraction tasks making use of entity and saliency based information.

Chapter 5

Embedding Tree Pruning and Re-Weighting in Learning to Rank

Learning-to-Rank (LtR) solutions are commonly used in large-scale information retrieval systems such as Web search engines where high quality documents needs to be returned in response to a user query within a fraction of a second. The most effective LtR algorithms, e.g., λ-MART, adopt a gradient boosting approach to build an additive ensemble of weighted regression trees. Since the required ranking effectiveness is achieved with very large ensembles, the impact on response time and query throughput of these solutions is not negligible.

In this chapter we propose X-CLEaVER, an iterative meta-algorithm able to build more efficient and effective ranking ensembles. X-CLEaVER interleaves the iterations of a given ensemble learning algorithm with pruning and re-weighting phases. First, redundant trees are removed from the ensemble generated, then the weights of the remaining trees are fine-tuned by optimizing the desired ranking loss function. We propose and analyse several pruning strategies and assess their benefits showing that interleaving pruning and re-weighting phases during learning is more effective than applying a single post-learning optimization step. Experiments conducted using two publicly available LtR datasets show that X-CLEaVER is very effective in optimizing λ-MART models both in terms of effectiveness and efficiency.

5.1 Introduction

The problem of ranking items in response to a given query is of general interest and it is of paramount importance for most information retrieval systems. A challenging example of that regards answering queries submitted to a Web Search Engine (WSE), where a small and relevant set of documents must be retrieved in fractions of a second

from a huge collection. The state of the art in ranking exploits supervised machine learning techniques based on Learning-to-Rank (LtR) algorithms [95]. Ranking models are in this case learnt from *ground truth* datasets composed of labeled training examples. The ranking function obtained allows to measure the relevance of each candidate document with respect to the user query.

While effectiveness of LtR methods has been always considered of primary importance, the efficiency requirements concerning the application of the learnt models in real production environments, characterized by strict time constraints, attracted the interest of the scientific community only recently [26, 30, 100, 7, 6, 175]. Ranking models deployed in large-scale information systems must instead feature very low latency as well as high throughput, due to the high rate of incoming user queries.

In this work we tackle the problem of improving efficiency and effectiveness of LtR models based on forests of additive regression trees, such as the ones learnt by the state-of-the-art λ-MART algorithm [178]. We move from the simple observation that the cost of applying such models is linear in the number of their trees, and that large ensembles composed of thousands of trees, despite being more accurate, are very expensive when exploited for ranking large sets of candidate documents [102]. We thus propose a meta-algorithm, named X-CLEAVER, which interleaves two novel steps of tree pruning and re-weighting within the usual iterative ensemble learning process: the pruning step aims at reducing the number of trees in the ensemble to improve its efficiency at scoring time, while tree re-weighting is an optimization process that aims to maximize the ranking quality of the pruned ensemble by tuning the weights associated with each tree. Our approach is totally agnostic with respect to a specific ensemble learning algorithm: it can exploit any given boosting algorithm as a black box as long as it produces a weighted ensemble of predictors.

X-CLEAVER stems from the CLEAVER algorithm [98], which applies similar optimizations *after* the completion of the learning phase to reduce the size of a given ensemble without affecting its quality. X-CLEAVER improves some of the strategies proposed in [98] and, more importantly, it shows that embedding such optimizations steps within the boosting LtR algorithm is a profitable strategy to achieve more compact and effective ranking models. It is worth remarking that other approaches were proposed to produce simpler and faster tree-based ensembles [6, 175]. However, these proposals aimed at finding a trade-off between efficiency and effectiveness during the learning phase, while X-CLEAVER aims at improving ranking quality and decreasing scoring cost at the same time. This twofold opportunity is justified by two observations that we illustrate below.

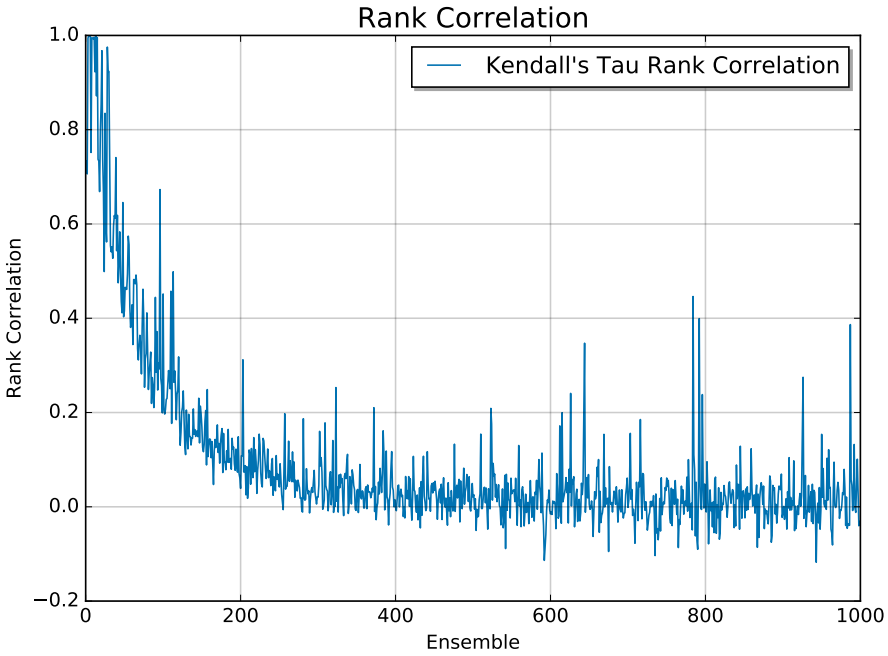

Figure 5.1: Average rank correlation (Kendall τ) between the ranked lists produced by consecutive trees of an ensemble.

The first observation is that tree forests generated by state-of-the-art LtR algorithms encompass a high level of redundancy. To support this claim, we trained by using optimal training parameters [30] a λ-MART forest of $1,000$ trees on a publicly available dataset[1]. We estimate the redundancy in the ensemble by measuring the similarity among the rankings generated by its trees. For every query in the test set, we thus computed the document ranking induced by each tree in isolation. Then, we use the Kendall's τ rank correlation coefficient[2] to estimate the similarity between the rankings obtained by each pair of consecutive trees. Figure 5.1 plots τ coefficient averaged over all the queries in the test set. As it can be seen, the correlation is particularly high for the first trees in the forest, while it tends to be positive but very irregular for the other trees. This high correlation suggests that some trees are redundant, and they can possibly be removed to make the model smaller and faster.

Pruning the ensemble obviously affects the accuracy of predicted scores. We thus propose to optimize the accuracy of the trees survived to the pruning step by fine-tuning their weights with an optimization heuristic driven by a ranking quality metrics. A second observation makes this weight optimization step particularly interesting.

[1]Dataset MSLR-WEB30K-F1, detailed in Section 5.4.

[2]We chose this metric because we are interested to the ranking problem, but a similar behavior can be observed by considering other correlation measures.

λ-MART, the state-of-the-art algorithm used in this work as base LtR solution is a list-wise algorithm aimed at minimizing a rank-based loss function. It is well-known that most rank-based measures are non-differentiable functions, and therefore simple techniques such as gradient descent cannot be applied [40]. The λ-MART algorithm thus casts the ranking problem into a regression problem solved through gradient boosting and by using the λ ranks as proxies of the gradient of the rank-based loss function [178]. This can result in sub-optimal choices performed during the learning process. Thus, our weight optimization step fulfills two objectives: on the one hand it counterbalances the effects of pruning, and, on the other hand, it guarantees the optimization of the actual measure of ranking quality considered, e.g., Normalized Discounted Cumulative Gain (*NDCG*).

By pushing the pruning and tree re-weighting phases within the learning process of λ-MART, X-CLEAVER builds smaller models outperforming λ-MART both in terms of efficiency and effectiveness due to the explicit optimization of the actual quality metrics.

In summary, the main contributions of this work are the following:

- we propose X-CLEAVER, a novel LtR meta-algorithm aimed at optimizing state-of-the-art LtR algorithms generating forests of weighted regression trees by interleaving tree pruning and re-weighting within the iterative learning process;
- we propose several tree pruning strategies aimed at obtaining a faster ranking model and a heuristic optimization for tuning tree weights. Experiments show that the proposed strategies allow the pruning of up to 80% of trees in a λ-MART ensemble without impacting its performance;
- we conduct a comprehensive evaluation of X-CLEAVER on two publicly available datasets. Experiments show that our technique remarkably outperforms, in both ranking quality and scoring efficiency, the reference λ-MART algorithm.

The paper is organized as follows. In Section 5.2 we discuss the research works that are related to LtR models optimization. In Section 5.3 we detail the proposed LtR algorithm showing the proposed pruning strategies and the greedy optimization process used to re-weight the trees in the ensamble. In Section 5.4 we first separately investigate the impact of the pruning and optimization processes, then we assess the performance of the novel X-CLEAVER algorithm as a whole. Finally, in Section 5.6 we draw the final conclusions and show some ideas we plan to investigate in the future.

5.2 Related Work

Ranging from large scale search systems to company-wide search systems using open-source solutions, LtR is one of the main tools used to improve the quality of the results delivered to search users. Many LtR strategies have been proposed in literature to learn ranking models. One pillar in the field has been the work by Herbrich*et al.* [76], where authors proposed to learn such models by adopting a pair-wise approach in place of a point-wise. Since that work, there has been a huge amount of works published with the goal of improving the quality of the generated models. Chapter 2.2.2 provides an in-depth overview about LtR.

Efficiency, which is the primary goal of the research presented in this chapter, has been widely addressed in the past in the machine learning community by works dealing with model pruning and feature selection.

Research in tree pruning dates back to 80's [132] and most of the recent works focus on pruning trees in an embedding. One of the most influential work in this area is that of Mehta *et al.* [107] where minimum description length principle is used to devise a novel strategy that instead of minimizing the length of the class sequence in the training sample together with the length of the decision tree it introduces a new length criterion to capture the intuitive idea of reducing the rate of misclassifications. One of the most recent works on pruning a random forest of decision trees is that of Ren *et al.* [138]. In their work authors propose two techniques that exploits the global knowledge derived by having the whole model already trained and available. Global refinement jointly relearns the leaf nodes of all trees under a global objective function so that the complementary information between multiple trees is well exploited. Global pruning has the double goal of reducing the model size and at the same time they aim at reducing the overfitting risk. Experiments they performed on real-world data show that the model obtained by the pruning method has a smaller footprint and higher effectiveness. The main difference between this work and ours is that the main task with which they deal is regression and classification and not LtR. In particular, our main goal is to improve *NDCG* which is considerably different from minimizing other metrics such as those used in regression and classification. Another recent development in ensembles pruning is the work of Qian *et al.* [130] where authors, differently from previous papers, try to solve the trade-off between effectiveness (maximizing the generalization performance) and efficiency (minimizing the number of weak learners) at the same time using a bi-objective problem formulation whose solution is found through an evolutionary Pareto optimization method combined with a local search

step. Empirical results show that this method is very effective in reducing the size of the ensemble while at the same time increasing the effectiveness of the method. The goal of our method is different in that we aim at reducing the size of the ensemble up to an apriori fixed number of weak learners while keeping the same quality level. In addition, our method is specifically tailored to LtR methods. From the same group of researchers, a Pareto optimization method is also used but with a different goal, i.e., reducing the number of features used while keeping the quality as high as possible [129]. Again, as in the case of ensemble of trees, the goal is reached through optimizing two objectives at the same time. Empirical results show that the feature selection method developed in their research is very effective when compared against seven state of the art feature selection methods. Feature selection is a very related method to what we do in our research. In fact, it is a technique that is orthogonal to ensemble pruning as it can be used in addition to it to reduce the number of features used by the model. Another recent result on pruning an ensemble model is that of Nan *et al.* [118] where they propose pruning of a random forest as an 0-1 integer program with linear constraints that encourages feature re-use. They aim at reducing the size of a random forest while trying also to reduce the number of variables used in the ensemble. It can be considered as a sort of pruning and feature selection mechanism. The method is empirically tested and provides very good performance improvements with respect to the state of the art. The main differences between this work and ours is, again, that the method is not explicitly targeting ranking problems and it is not clear if it can be extended straightforwardly to other ensemble methods like λ-MART or MART.

The research presented in this chapter is mostly concerned with how efficiently an already built LtR model can be deployed in a production system. Efficiency in LtR has been investigated in the past by following, mainly, two different research directions. The first direction includes proposals of solutions that trade efficiency off for effectiveness during the model building phase. Asadi and Lin observe that compact, shallow, and balanced trees often yield to produce predictions quicker [6]. The technique they propose to build ensembles of decision trees incorporate a notion of execution cost that penalizes the generation of trees that will result in low scoring performance. Authors propose two strategies for accomplishing this: i) by directly modifying the node splitting criterion during tree induction, and ii) by stage-wise tree pruning. Experiments on a standard LtR datasets show that the pruning approach is the best. Comparing to our work, we also exploit a pruning strategy but our goal is not to affect the shape of the trees but rather to eliminate from the ensembles

trees who do not effectively contribute to the final rankings. Wang *et al.* present a unified framework for jointly optimizing effectiveness and efficiency [175]. Authors propose new metrics that capture the trade-off between these two competing forces and devise a strategy for automatically learning models that directly optimize the trade-off metrics. Experiments show that models learnt by directly optimizing the effectiveness efficiency trade-off produce, in fact, ranking systems able to retrieve high quality results at a considerable reduced inference cost. Authors also show that their approach naturally leads to a reduction in the variance of query execution times, which is an important aspect leading to a better exploitation of the search infrastructure, load balancing and user satisfaction.

The second research line considers low-level optimizations to the inference phase by speeding-up the traversal of a given tree ensemble. Asadi *et al.* [7] propose to rearrange the code visiting the ensemble by transforming *control hazards* into less expensive *data hazards*, i.e., data dependencies introduced when one instruction requires the result of another. Moreover, to reduce *data hazards* the same work proposes to *vectorize* the scoring algorithm by interweaving the evaluation of a small set of candidate documents. Lucchese *et al.* [100, 55] propose QuickScorer a scoring algorithm, which adopts a new representation of the tree ensemble based on bit-vectors. The tree traversal, aimed to detect the leaves contributing to the final scoring of a document, is performed feature by feature, over the whole tree ensemble, through efficient logical bitwise operations. In addition, the traversal is not performed one tree after another, as one would expect, but it is interleaved, feature by feature, over the whole tree ensemble. The two research directions above are orthogonal to ours as we aim at producing a faster model that does not lose its effectiveness and that can by integrated in any scoring algorithm. Moreover, our methodology is totally agnostic with respect to both the LtR algorithm. It can be applied to all LtR algorithms producing ensemble models.

5.3 Growing and Pruning Tree Ensembles

Gradient boosting is an iterative technique used for regression or classification problems that learns from a training set an additive ensemble $\mathcal{E}$ of weighted *weak learners* minimizing a given loss function $L(\mathcal{E})$. In this work we focus on ensemble-based ranking models where the weak learners are decision trees. Large ensembles encompassing thousands of trees are usually required to achieve the high ranking quality required by WSEs [30]. Since the cost of evaluating an ensemble $\mathcal{E}$ is linear in its size $|\mathcal{E}|$, the ability of learning compact ensembles without compromising accuracy is a very desirable

property. Reducing document scoring time allows in fact to cope with time budget constraints and improves the overall throughput of large search infrastructures [54, 26].

Let $\mathcal{E} = \{(t_1, w_1), \ldots, (t_{|\mathcal{E}|}, w_{|\mathcal{E}|})\}$ be an additive ensemble of decision trees, where each tree t_i is weighted by w_i, $w_i \in \mathbb{R}$. Given a query q and a document d, the document relevance score predicted by each tree t_i is denoted by $s_i(q, d)$, and the global ensemble prediction $S(q, d)$ is computed as:

$$S(q, d) = \sum_{i=1}^{|\mathcal{E}|} w_i \cdot s_i(q, d) \tag{5.1}$$

Gradient boosting algorithms learn trees and weights iteratively, so that each pair (t_i, w_i) is generated at iteration i to improve the predictions of the ensemble built so far. Indeed, each tree t_i approximates the gradient of a loss function L with respect to the model parameters.

Model $\mathcal{E}$ is learnt from a *ground truth* dataset containing query-document pairs (q, d) labeled with multi-graded relevance levels (usually ranging in $\{0, 1, 2, 3, 4\}$). The goal of the LtR algorithm is learning a model that can produce a document ranking agreeing with the relevance ordering of the ground truth. Indeed, ensemble predictions $S(q, d)$ are aimed at establishing the correct ranking of the list of candidate documents, rather than to predict their exact relevance labels. The quality of a ranking model is in fact measured by means of rank-based metrics such as Normalize Discounted Cumulative Gain (*NDCG*) or Expected Reciprocal Rank (*ERR*), which consider both the relevance and the position of each document in the ranked list. Unfortunately, loss functions based on such rank-based measures are not differentiable. Therefore LtR algorithms cannot apply directly gradient boosting to optimize these loss functions. To overcome this issue λ-MART, the state-of-the-art list-wise LtR algorithm, tries to optimize rank-based metrics by using an approximation of loss function's gradient, named λ-ranks. At each iteration i, λ-MART trains a tree t_i so as to minimize the squared loss with respect to such λ-ranks. Thus, λ-MART employs a proxy loss function instead of the target rank-based metric. This may generate sub-optimal ranking choices.

In this work we propose X-CLEAVER, a novel LtR meta-algorithm that runs on top of a *base ensemble learning algorithm*. It is able to learn ensembles of additive regression trees that are more compact and effective than the ones created by the given base algorithm. Moreover, our technique provides a direct optimization of rank-based

Algorithm 1 X-CLEAVER.

1: **function** X-CLEAVER($N, n, p, \mathcal{A}, L$)
input:
2: N : *ensemble size*
3: n : *number of regression trees trained per iteration*
4: p : *ratio of pruned trees per iteration*
5: $\mathcal{A}$: *base learning algorithm*
6: L : *loss function*
output:
7: $\mathcal{E}$: *trained ensemble*
8: $\mathcal{E} \leftarrow \emptyset$ ▷ the current ensemble model
9: **while** $|\mathcal{E}| < N$ **do**
10: $\widehat{\mathcal{E}} \leftarrow \mathcal{A}$.GROWMODEL($\mathcal{E}, n$) ▷ build delta model
11: $\widehat{\mathcal{E}}_P \leftarrow$ PRUNE($\widehat{\mathcal{E}}, L, p$)
12: $\widehat{\mathcal{E}}_W \leftarrow$ REWEIGHT($\widehat{\mathcal{E}}_P, L$)
13: **if** $L(\mathcal{E} \cup \widehat{\mathcal{E}}_W) \geq L(\mathcal{E})$ **then** ▷ no gain condition
14: **break**
15: $\mathcal{E} \leftarrow \mathcal{E} \cup \widehat{\mathcal{E}}_W$
16: **return** $\mathcal{E}$

metrics within the iterations of existing ensemble-learning algorithms. X-CLEAVER interleaves three phases during the learning:

i) a *growing phase*, during which the base algorithm is used to learn a set of additional trees, named *delta model*;

ii) a *pruning phase*, which removes the less relevant decision trees from the delta model, thus improving the *model efficiency*;

iii) a *re-weighting phase*, during which the weights associated with the remaining trees of the delta model are fine-tuned, by directly optimizing a loss function based on a rank-based evaluation measure, thus improving the *model effectiveness*.

These three phases are iterated until the desired ensemble size is achieved.

5.3.1 X-CLEAVER Algorithm

The pseudocode of X-CLEAVER is illustrated in Algorithm 1. During its first phase, X-CLEAVER exploits the base algorithm $\mathcal{A}$ to *grow* $\mathcal{E}$, the (initially empty) current model (line 10)[3]. At each iteration the ensemble learning algorithm $\mathcal{A}$ is used to train, on the basis of the current model $\mathcal{E}$, a new set of n weak learners, denoted with $\widehat{\mathcal{E}}$.

[3]In this work we focus on the state-of-the-art λ-MART as base ensemble learning algorithm but, in general, X-CLEAVER can exploit any other ensemble learning technique.

Before adding $\widehat{\mathcal{E}}$ to the current model $\mathcal{E}$, X-CLEAVER applies the *pruning* and *re-weighting* phases. During the pruning phase (line 11), a fraction p of rankers belonging to $\widehat{\mathcal{E}}$ is removed. Pruning is naturally used to reduce the scoring time by limiting the ensemble size. It exploits the fact that some weak rankers are highly correlated (see Figure 5.1), and therefore can be removed from the model without impacting significantly the model accuracy. In Section 5.3.2 we explore different pruning strategies aimed at identifying the set of rankers that less impact the overall model prediction power. We denote by $\widehat{\mathcal{E}}_P$ the set of weak learners in $\widehat{\mathcal{E}}$ that survive after the application of the pruning strategy.

As the pruning phase may affect the quality of the model, X-CLEAVER applies a *re-weighting* to the non-pruned trees $\widehat{\mathcal{E}}_P$ (line 12). To this end, X-CLEAVER exploits a local optimization strategy based on line search. As discussed in Section 5.3.3, the weights associated with the weak learners of $\widehat{\mathcal{E}}_P$ are fine-tuned by directly optimizing the given loss function L. This allow us to embed into the learning process the direct optimization of a non differentiable loss function L, although the base LtR algorithm $\mathcal{A}$ can only optimize a proxy of L. The final pruned and re-weighted *delta model* is denoted by $\widehat{\mathcal{E}}_W$

If $\widehat{\mathcal{E}}_W$ does not help in improving the quality of the current model $\mathcal{E}$ according to the loss function L, then X-CLEAVER terminates (line 13). Otherwise, $\widehat{\mathcal{E}}_W$ is added to $\mathcal{E}$ and the above three phases are iterated until the desired ensemble size N is achieved.

5.3.2 Pruning Phase

The pruning phase of X-CLEAVER identifies a fraction p of trees to be removed from delta model $\widehat{\mathcal{E}}$. In the following we review the different pruning strategies we propose to identify these trees.

- LAST: this strategy prunes the last fraction p of trees in the ensemble $\widehat{\mathcal{E}}$. The motivation is that trees are progressively built and added to the ensemble to refine the quality of the overall extracted model. The last trees are thus expected to provide a smaller contribution.

- RANDOM: This technique prunes uniformly at random a fraction p of trees from $\widehat{\mathcal{E}}$. The best result out of r pruning rounds is selected, where quality is measured according to the given rank-based loss function L. The main difference between this strategy and a standard random approach, i.e., with a single pruning round, is that we exploit L in choosing the best random-generated subset. In our tests,

we performed $r = 100$ random pruning rounds before selecting the best pruned ensemble.

- SKIP: this strategy removes trees that are uniformly scattered along the sequence of trees in $\widehat{\mathcal{E}}$. One tree every $1/p$ is removed, i.e., trees at position $\lceil i/p \rceil$ with $i = 1, \ldots, \lceil np \rceil$. The rationale is that trees learnt in close iterations are similar, and potentially redundant.

- LOW-WEIGHTS: this strategy removes from $\widehat{\mathcal{E}}$ the fraction p of trees with the lowest weights w_i. The assumption here is that they are less relevant for the final document scoring. For those learning algorithms $\mathcal{A}$ that associate uniform weights with all the trees, a preliminary *re-weigthing phase* is applied to $\widehat{\mathcal{E}}$ (see Sec. 5.3.3) in order to obtain the weights to which this pruning strategy is in turn applied.

- QUALITY-LOSS: this strategy considers the actual impact of a tree in $\widehat{\mathcal{E}}$ to the optimization of the loss function L. The impact of a tree is measured as the loss variation caused by its removal. The tree with the smalled impact is removed, and the impact of the remaining trees in the pruned ensemble is recomputed before pruning the next tree. The procedure is repeated until a fraction p of trees is removed from $\widehat{\mathcal{E}}$. Although the greedy choice of which tree to prune is locally optimal, there is no guarantee about the global optimality of the pruned ensemble $\widehat{\mathcal{E}}_P$. A simplified version of the same strategy, presented in [98], measures the trees' impact only once, and then selects the trees to discard according to this impact-based ranking. Despite being less demanding in terms of computational complexity, it does not take into account trees' dependencies, i.e., two highly redundant trees risk to be removed both.

- SCORE-LOSS: this strategy considers the normalized contribution provided by each tree to the final score $S(q, d)$. The contribution is measured as

$$\frac{w_i \cdot s_i(q, d)}{S(q, d)}$$

and it is averaged over all query-document pairs of the training dataset. Then, the fraction p of trees in $\widehat{\mathcal{E}}$ with the lowest average contribution is pruned.

Eventually, the *pruning phase* produces the pruned ensemble $\widehat{\mathcal{E}}_P \subseteq \widehat{\mathcal{E}}$ (line 11).

5.3.3 Re-weighting phase

The re-weighting phase of X-CLEAVER fine-tunes the pruned model $\widehat{\mathcal{E}}_P$ in order to create a new ensemble $\widehat{\mathcal{E}}_W$ having the same trees of $\widehat{\mathcal{E}}_P$ and a new set of weights such that the loss $L(\mathcal{E} \cup \widehat{\mathcal{E}}_W)$ is minimized.

Let $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{|\widehat{\mathcal{E}}_W|}\}$ be the weights vector of the ensemble $\widehat{\mathcal{E}}_W$, our goal is to solve the following optimization problem:

$$\overline{\Gamma} = \arg\min_{\Gamma \in \mathbb{R}^{|\widehat{\mathcal{E}}_W|}} L(\mathcal{E} \cup \widehat{\mathcal{E}}_W).$$

Finding the optimal $\overline{\Gamma}$ is not feasible. Firstly because the most common loss/gain functions used in ranking problems are not differentiable, and secondly because Γ can have a high number of dimensions. Therefore, standard techniques, such as gradient descent, cannot be exploited.

We address the optimization problem through a local-search heuristic that directly minimizes the given loss function L. Analogously to [162], X-CLEAVER exploits an iterative two-step procedure based on *line search*.

Let Γ^k be the vector of weights found at iteration k of our line search procedure. During the first step, a descent direction originating in Γ^k is identified. Then, in the second step, a new weight vector Γ^{k+1} that improves the loss function L is searched along such direction. Starting from Γ^0 being the weights of $\widehat{\mathcal{E}}_P$, the two steps detailed below are iterated until L, measured on a separate validation set, does not change for a fixed number of iterations:

1. Given the solution Γ^k at iteration k, a line search along each axis of the weight vector is performed independently. For each dimension, the weight γ_i is replaced with a candidate weight computed by testing σ equi-spaced samples in the interval $[\gamma_i - \omega, \gamma_i + \omega]$, while keeping all other γ_j fixed. The parameters σ and ω affect the granularity of the local search[4]. The best among the σ samples, denoted by d_i, is selected by evaluating $L(\mathcal{E} \cup \widehat{\mathcal{E}}_W)$ with the modified set of weights. Eventually, the independent line searches lead to a new point $D = \{d_1, d_2, \ldots, d_{|\widehat{\mathcal{E}}_W|}\}$. Fig. 5.2 exemplifies the line search algorithm on a two-dimensional search space: along each axis, $\sigma = 9$ samples (horizontal and vertical red dots) are evaluated independently in order to choose the best weight updates (circled in blue) along the directions γ_0 and γ_1.

[4]The value of ω is reduced by a shrinking factor η at each iteration to favor a fine-grained optimization when the algorithm approaches a local minimum.

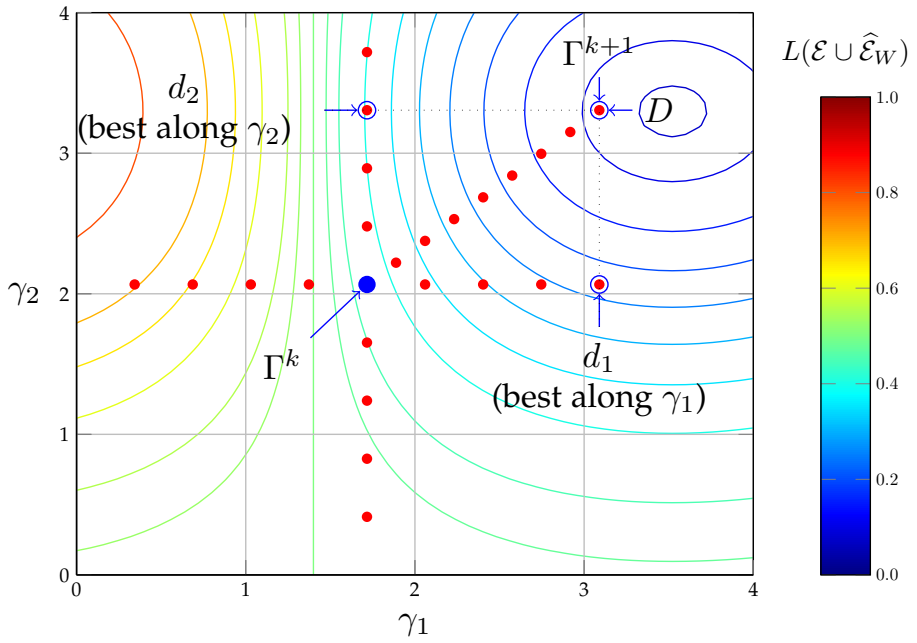

Figure 5.2: Line Search visual interpretation

2. An additional line search is conducted along the promising descent direction identified at the previous step, i.e., along the segment connecting Γ^k to D. Again σ equi-spaced samples are evaluated and the best is chosen so as to minimize $L(\mathcal{E} \cup \widehat{\mathcal{E}}_W)$. The best weight vector found, denoted by Γ^{k+1}, is used as the starting point for the next iteration. In Fig. 5.2, the best among the σ samples is exactly point D.

In order to reduce the overall computational cost of the above search process, we exploit the following optimizations. First, thread-level parallelism is used in order to explore the different search directions during the first step, i.e., to find the various d_i, and to evaluate the σ samples during the second step. Finally, we avoid visiting the whole tree ensemble for scoring documents after each weight update. The ensemble of trees is in fact visited only once and tree predictions $s_i(q, d)$ for all the documents of the training dataset are stored in memory, thus allowing a fast access to these predictions when the single weight γ_i is updated.

Table 5.1: Properties of the MSLR-WEB30K-F1 and Istella-S datasets.

Dataset	MSLR-WEB30K-F1	Istella-S
# queries	31,351	33,018
# query-doc pairs	3,771,125	3,408,630
# features	136	220
avg. # docs/query	120	103
# pos. query-doc pairs	1,830,173 (48.53%)	388,224 (11.39%)
# neg. query-doc pairs	1,940,952 (51.47%)	3,020,406 (88.61%)

5.4 Experimental Evaluation

We conduct experiments by employing two public LtR datasets: i) the MSLR-WEB30K-F1[5] (Fold 1) dataset and ii) the Istella-S[6] dataset [98]. The MSLR-WEB30K-F1 dataset contains 3,771,125 query-document pairs referred to 31,351 queries, while the Istella-S dataset includes 33,018 queries and 3,408,630 query-document pairs. Query-document pairs are represented in the two datasets with 136 and 220 features, respectively. The characteristics of the datasets are listed in Table 5.1.

The query-document pairs in both datasets are labeled by relevance judgments that are natural numbers ranging from 0 (irrelevant) to 4 (perfectly relevant). While the size of the two datasets is comparable, they show a different proportion between positive and negative examples. Indeed, the Istella-S dataset contains a lower number of positive query-doc pairs than MSLR-WEB30K-F1 (11.39% versus 48.53% of the examples). Moreover, as reported in Table 5.2, the distributions of positive labels in the two datasets are different with MSLR-WEB30K-F1 showing a skewed distribution that is not observable in Istella-S. MSLR-WEB30K-F1 comes with a number of low relevance labels of 66.98% of the total set of positive examples. In the Istella-S dataset the low relevance class accounts instead for only 21.42%, while the medium and the high relevance classes account for 35.03% and 24.20%, respectively. MSLR-WEB30K-F1 shows a small number of perfectly relevant query-document pairs (i.e., 1.66% of the total), while in the Istella-S dataset the same class accounts for 19.35%.

We split each dataset in train, validation and test sets according to a 60%-20%-20% schema. Then, we use training and validation datasets to train a *reference model* with λ-MART, a *list-wise* LtR algorithm using *NDCG* in its loss function [178]. To this end, we exploit the open-source implementation of λ-MART provided by [29]. We fine-tune the hyper parameters of the algorithm by sweeping them to maximize *NDCG*@10.

[5] http://research.microsoft.com/en-us/projects/mslr/
[6] http://blog.istella.it/istella-learning-to-rank-dataset/

Table 5.2: Distribution of positive labels in the MSLR-WEB30K-F1 and Istella-S datasets.

# pos. query-doc pairs per label	MSLR-WEB30K-F1	Istella-S
# 1s (low relevance)	1,225,770 (66.98%)	83,167 (21.42%)
# 2s (medium relevance)	504,958 (27.59%)	135,989 (35.03%)
# 3s (high relevance)	69,010 (3.77%)	93,957 (24.20%)
# 4s (perfect relevance)	30,435 (1.66%)	75,111 (19.35%)
Total # pos. examples	1,830,173	388,224

Different cut-off values do not exhibit appreciable differences. We vary the maximum number of leaves in each tree in the set $\{5, 10, 25, 50\}$, while the learning rate in $\{0.05, 0.1, 0.5, 1.0\}$. To avoid overfitting, we allow the algorithm to train up to 1,500 trees unless there is no improvement in *NDCG*@10 on the validation set during the last 100 iterations. We obtain the best results in terms of *NDCG*@10 for both datasets using trees with 50 leaves at maximum, and a learning rate equal to 0.05 (the same settings are reported as optimal in [30]). The resulting forests are composed of 1,199 and 1,497 trees for MSLR-WEB30K-F1 and Istella-S, respectively. It is worth noting that the experiments discussed here were conducted also using Gradient Boosting Regression Trees (MART) a different LtR algorithm [65][7]. However, the performance of MART resulted to be worse than those of λ-MART in all the tests conducted. Therefore, in the following we employ λ-MART as reference learning algorithm for X-CLEAVER.

The tests presented in this chapter are performed on a machine equipped with a dual CPU AMD Opteron 6276, a 16 cores NUMA processor clocked at 2.30GHz, with 16 MB of cache L3 and 32GB of DDR3 RAM. To facilitate the reproducibility of the results, we release our implementation of X-CLEAVER as part of the QuickRank C++ library for Learning to Rank [29][8].

5.4.1 Effectiveness of pruning strategies

We first investigate the effectiveness of X-CLEAVER pruning strategies when applied to ranking models of varying sizes previously trained with λ-MART. This corresponds to run a single iteration of our X-CLEAVER meta-algorithm. A similar analysis is reported in [98]. However, the experiments discussed here are novel as two pruning

[7] MART is a *point-wise* algorithm that uses the root mean squared error as loss function, resulting in a predictor of relevance labels.

[8] The source code of QuickRank can be downloaded from: http://quickrank.isti.cnr.it

strategies (Random and Quality-Loss) have been improved, while the QuickRank library was further optimized for both effectiveness and efficiency.

Table 5.3 reports the values of *NDCG*@10 measured on the test sets for the original λ-MART models and the ones obtained by applying a single iteration of X-CLEaVER. Specifically, the three tables reports the results obtained on both datasets starting from models consisting of 100, 500, and 1,000 trees, respectively. The cell reporting the *NDCG*@10 achieved by the reference λ-MART model is highlighted in each table using a dark-gray background. In the same row of each table we report the performance of the intermediate λ-MART models (trained by the same learning process) having a number of trees ranging from 90% to 10% of the reference one. Exactly below the dark-gray cell, in the first column of the row referred to strategy Last, we report the performance of the ensemble obtained from the reference model without performing any pruning but only tree re-weighting with our line search procedure.

We are interested here in comparing the performance of the reference λ-MART models with those obtained with X-CLEaVER for different pruning levels. In particular we want to assess if our pruning and optimization strategies together are able to produce ranking models that are smaller and at least as effective as the reference model. As an example, the λ-MART reference model of 100 trees reaches on MSLR-WEB30K-F1 a *NDCG*@10 of 0.4540. The intermediate λ-MART model with 50 trees scores only 0.4333. The X-CLEaVER model of 50 trees, obtained from the MSLR-WEB30K-F1 100-tree model using the Quality-Loss strategy, achieves instead a *NDCG*@10 of 0.4643.

For each pruning strategy assessed in a different row of the tables, light-gray cells highlight where we achieved performances greater than or equals to those obtained using the reference λ-MART model. The values in bold highlight instead the most aggressive pruning rate preserving the ranking quality of the reference model. Moreover, values labeled with * identify the performance of the smallest models which proved to be statistically equivalent to the reference λ-MART model. Randomization test with 10,000 permutations and $p - value \leq 0.05$ is performed to assess if the *NDCG*@10 differences between the reference and the pruned models are statistically significant or not [153].

The reported results confirm the validity of the proposed strategies. The first table, referred to ensembles generated from a reference model of 100 trees, shows that X-CLEaVER is able to improve the reference model by using any pruning strategies. For the least aggressive pruning rates, we can observe also remarkable gains in terms

Table 5.3: Values of *NDCG*@10 measured for λ-MART reference models of different size and X-CLEAVER ones obtained at various pruning levels with the different pruning strategies. The * symbol labels the most aggressive pruning rate for each strategy resulting in a model statistically equivalent to the reference one.

100 trees

	MSLR-WEB30K-F1										Istella-S									
	Pruned Model Size										Pruned Model Size									
Strategy	100	90	80	70	60	50	40	30	20	10	100	90	80	70	60	50	40	30	20	10
λ-MART	.4540	.4507	.4477	.4432	.4386	.4333	.4260	.4179	.4085	.3983	.6988	.6946	.6897	.6835	.6766	.6671	.6571	.6441	.6266	.6137
LAST	.4648	.4620	.4605	**.4573***	.4516	.4482	.4407	.4300	.4107	.4003	.7121	.7094	.7052	**.7001***	.6906	.6833	.6727	.6629	.6454	.6201
RANDOM	-	.4644	.4635	.4629	.4624	.4606	.4602	.4599	**.4553***	.4455	-	.7131	.7129	.7118	.7086	.7122	.7089	.7053	**.7015***	.6893
SKIP	-	.4645	.4646	.4630	.4632	.4611	.4597	**.4596**	.4523*	.4465	-	.7124	.7117	.7111	.7104	.7086	.7076	.7041	**.7012***	.6906
LOW-WEIGHTS	-	.4646	.4647	.4653	.4633	.4622	**.4544***	.4411	.4188	.3888	-	.7122	.7120	.7115	.7104	**.7054***	.6919	.6674	.6404	.6204
QUALITY-LOSS	-	.4646	.4651	.4648	.4644	.4643	.4641	.4630	**.4580***	.4486	-	.7128	.7127	.7126	.7127	.7130	.7132	.7116	**.7098**	.6979*
SCORE-LOSS	-	.4645	.4630	.4627	.4616	.4609	.4586	**.4554***	.4472	.4330	-	.7104	.7091	.7108	.7110	.7068	.7052	.6937	**.7003***	.6852

500 trees

	MSLR-WEB30K-F1										Istella-S									
	Pruned Model Size										Pruned Model Size									
Strategy	500	450	400	250	300	250	200	150	100	50	500	450	400	250	300	250	200	150	100	50
λ-MART	.4766	.4758	.4752	.4752	.4745	.4722	.4694	.4644	.4540	.4333	.7433	.7419	.7397	.7379	.7343	.7302	.7239	.7146	.6988	.6671
LAST	.4782	.4780	**.4776**	.4765	.4765*	.4751	.4730	.4708	.4648	.4482	.7473	.7460	.7455	**.7439***	.7414	.7379	.7344	.7247	.7121	.6833
RANDOM	-	**.4772**	.4760*	.4744	.4748	.4743	.4741	.4698	.4679	.4615	-	.7469	.7468	.7459	.7459	**.7450**	.7420*	.7374	.7348	.7249
SKIP	-	**.4773**	.4759	.4769	.4758*	.4746	.4741	.4724	.4698	.4622	-	.7463	.7473	.7459	.7452	**.7448**	.7430*	.7396	.7367	.7255
LOW-WEIGHTS	-	.4759	.4754*	.4367	.4281	.4129	.4065	.3981	.3923	.3554	-	.7470	**.7438***	.7390	.7147	.7022	.5963	.5789	.5551	.4969
QUALITY-LOSS	-	.4781	.4781	.4777	.4780	.4787	.4760	**.4774***	.4740	.4686	-	.7475	.7470	.7464	.7457	.7449	.7447	**.7433***	.7390	.7309
SCORE-LOSS	-	.4753	.4758	.4753	.4750	.4753*	.4734	.4736	.4696	.4546	-	.7461	.7458	.7455	**.7435**	.7423*	.7376	.7320	.7248	.6999

1000 trees

	MSLR-WEB30K-F1										Istella-S									
	Pruned Model Size										Pruned Model Size									
Strategy	1000	900	800	700	600	500	400	300	200	100	1000	900	800	700	600	500	400	300	200	100
λ-MART	.4789	.4785	.4784	.4779	.4774	.4766	.4752	.4745	.4694	.4540	.7495	.7491	.7478	.7467	.7451	.7433	.7397	.7343	.7239	.6988
LAST	.4789	.4785	.4776	.4790	.4789	.4783*	.4776	.4765	.4730	.4646	.7509	.7510	.7499	**.7498**	.7485*	.7473	.7454	.7417	.7342	.7121
RANDOM	-	.4788	.4764	.4783*	.4754	.4768	.4720	.4721	.4670	.4580	-	.7505	.7502	**.7506**	.7490*	.7464	.7446	.7430	.7387	.7297
SKIP	-	.4784	.4778	.4782*	.4771	.4771	.4754	.4734	.4700	.4620	-	.7506	.7495	**.7500**	.7485	.7493*	.7459	.7422	.7384	.7259
LOW-WEIGHTS	-	.4788	.4780	.4788	.4788	.4783*	.4738	.4605	.4337	.4304	-	.7489*	.7359	.6507	.6362	.6222	.5995	.5831	.5452	.5031
QUALITY-LOSS	-	.4793	.4797	.4798	.4797	.4795	**.4804**	.4783*	.4758	.4722	-	.7514	.7509	.7514	.7519	**.7507**	.7488*	.7462	.7431	.7371
SCORE-LOSS	-	.4768	.4761	.4751	.4709	.4760	.4765	.4753	.4732	.4688	-	.7509	.7493	**.7498**	.7486*	.7479	.7460	.7453	.7365	.7196

of *NDCG*@10. On the MSLR-WEB30K-F1 dataset, Random and Quality-Loss perform the best in granting equivalent ranking quality with very aggressive pruning rate. In these cases X-CLEaVER is able to prune up to 80% of the original ensemble without loosing quality. On the Istella-S dataset, Random, Skip, Quality-Loss, and Score Loss strategies can prune up to 80% of the trees and obtain *NDCG*@10 figures greater or equal to that of the reference 100-tree model.

When the size of the reference λ-MART model is increased to 500 trees, the best performing pruning strategy results to be Quality-Loss. This strategy is able to prune up to 70% of the trees and provide models with accuracy equivalent or higher than the reference ensemble. The superiority of Quality-Loss in pruning the ensemble is even more evident when models of $1,000$ trees are considered. Here, Quality-Loss is the only pruning strategy allowing to improve the performance of the reference λ-MART ensemble with pruning rates of up to 60% on MSLR-WEB30K-F1 and to 50% on Istella-S.

From the three tables we can observe that the number of light-gray cells decreases as the size of the reference model increases. The possible explanation of this phenomenon is twofold. First, large models are more effective than small ones and the space left for improvements is thus very little. Second, our line search optimization process is not particularly effective on ensembles having a large number of trees since it is more likely to converge to a local optimum. We lean towards this second hypothesis that will be supported by some of the experiments discussed in the following.

We conclude that Quality-Loss is the best performing pruning strategy among the proposed ones, as it provides the smallest models in all experiments conducted. Indeed, it is the only strategy that exploits the ranking metric being optimized. The overall benefit of our proposal is remarkable since on both the datasets X-CLEaVER can exploit Quality-Loss to remove from 50% to 90% of the trees in the reference ensemble without hindering ranking quality.

5.4.2 Qualitative analysis of pruning strategies

We analyze here the behavior of the pruning strategies embedded in X-CLEaVER by evaluating the distribution of the trees removed from the ensemble. The analysis is conducted by applying a pruning rate $p = 50\%$ to a λ-MART model with $1,000$ trees trained on the Istella-S dataset.

The histograms reported in Figure 5.3 show the percentage of pruned trees across buckets of 100 consecutive trees of the input ensemble. As an example, the Last strategy (Fig. 5.3 upper-right) straightforwardly remove all the trees in the last five

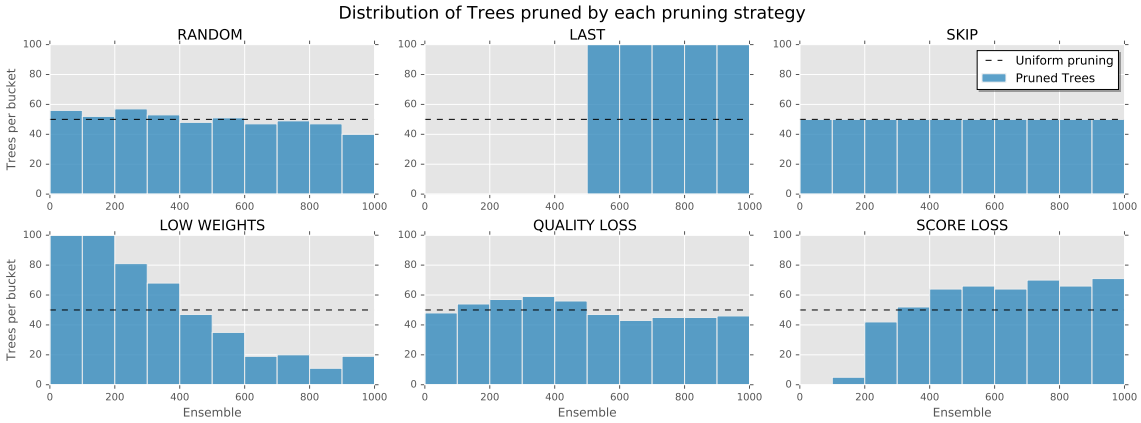

Figure 5.3: Distributions of removed trees for each pruning strategy (with $p = 50\%$) applied to a λ-MART model with 1,000 trees trained on Istella-S. Values are averaged over buckets of 100 consecutive trees.

buckets, for a total of 500 trees out of $1,000$. Not surprisingly, strategy Skip performs a uniform pruning as it removes equidistant trees from the input forest, and Random behaves similarly, despite in this case the pruning is not perfectly uniform due to the smart selection of the best performing pruned ensemble subset.

Interestingly, strategy Low-Weights removes most of the trees from the first buckets of the ensemble. A possible explanation is that initial trees are the most redundant, while the last ones try to fine-tune document scores. Strategy Score-Loss, which considers the relative contribution of each tree to document score, behaves symmetrically to Low-Weights and tends to prune trees from the last buckets of the ensemble. Boosting learning algorithms in fact assigns larger scores to the leaves of the initial trees which accounts for the largest part of the final score, and smaller scores to the last trees which are responsible for fine-tuning.

Quality-Loss, the best performing strategy shows a smoother pruning behavior. Pruned trees are distributed almost uniformly, except for those in the range 100 to 500 where the pruning is more aggressive. This suggests that evaluating the contribution of each tree to the given metric function (as opposed to the final document score) leads to a more balanced pruning, which in turn results in a more accurate model.

Finally, we analyze how tree weights are modified by our ReWeight procedure aimed at optimizing the *NDCG* score. To this end we measure the variations to per-bucket average weights after the Quality-Loss pruning strategy halved the size of a λ-MART model of 1,000 trees trained on Istella-S. Figure 5.4 shows the results of this analysis where the dashed horizontal line represents the uniform weights of the reference model, rescaled to the unit. Interestingly, the average weights in the first

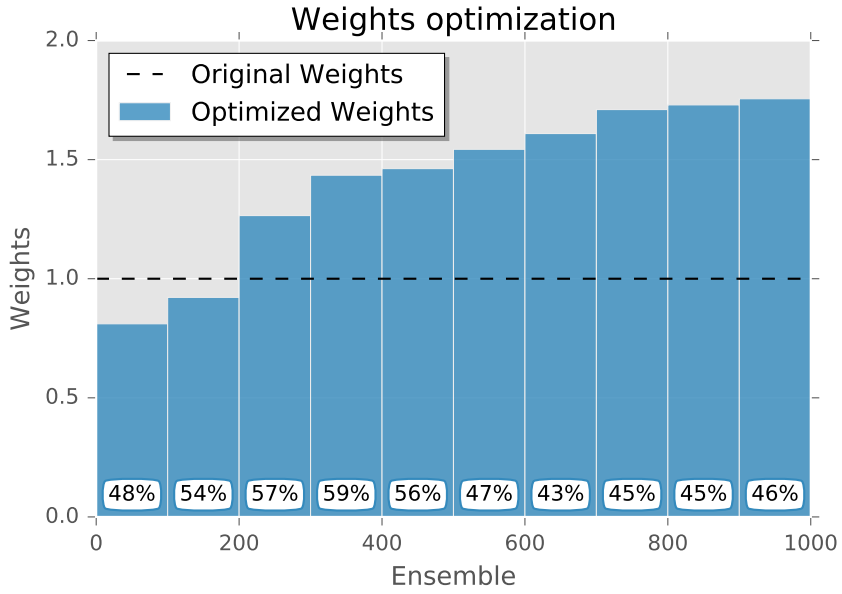

Figure 5.4: Average per-bucket weights of the optimized model. The dashed line corresponds to the uniform weight used in the reference model. The value reported inside each bar measures the percentage of pruned trees in the bucket.

two buckets result to be lower than in the original model, while the weights in the last buckets are boosted. Moreover, the weights increase monotonically, suggesting that the last trees of the ensemble are very important, and that the original learning algorithm tends to underestimate their importance. The analysis reveals that, while the pruning step removes redundancy, the line search algorithm fine-tunes the weights by giving more importance to the last part of the forest.

5.4.3 X-CLEAVER analysis

In this section we discuss the experiments aimed at assessing if X-CLEAVER can train smaller and more accurate models than the reference LtR algorithm. We use as a reference the models trained with λ-MART by using the best training parameters discussed before (50 leaves and a learning rate equal to 0.05). The resulting ensembles have 1,199 and 1,497 trees for MSLR-WEB30K-F1 and Istella-S, respectively. Given the previous experimental results and to make the discussion clearer, we limit the analysis to the best performing pruning strategy QUALITY-LOSS.

As shown in Table 5.4, the reference λ-MART models achieve 0.4791 and 0.7530 in terms of *NDCG*@10 on MSLR-WEB30K-F1 and Istella-S, respectively. To provide a consistent baseline for each pruned X-CLEAVER model, we also report *NDCG*@10 values of the incremental λ-MART models generated every 100 iterations. As in the

previous table, light-gray cells highlight performance greater or equals than the ones of the reference λ-MART model, while bold results evidence the best performing model for each ensemble size. Finally, values labeled with symbol * identify the performance of the smallest models which proved to be statistically equivalent to the reference one.

To first investigate the contribution of line search, we apply the re-weighting strategy to the incremental λ-MART models produced during the learning of the reference model (see line λ-MART + *LineSearch*). On both datasets, the larger benefit is achieved on small models. On the MSLR-WEB30K-F1 dataset line search boosts the 600-trees model to the same performance of the 1,000-trees λ-MART. Similarly, on Istella-S, the 700-trees models after line search provides a better *NDCG* than the original 1,000-trees λ-MART. With larger models, line search results to be less beneficial. For example on the tests conducted with the MSLR-WEB30K-F1 dataset it does not provide any improvement from models with 800 and more trees.

To show that, even when applied only once, our optimization strategies still works well, we run a single iteration of the X-CLEAVER algorithm and report the results of the tests conducted in the row labeled X-CLEAVER$_{iter=1}$. In this case we set X-CLEAVER to apply a $p = 50\%$ pruning rate to a λ-MART model being double in size than the final one. We first observe that X-CLEAVER$_{iter=1}$ always achieves better results than line search only, for every model size tested. Second, we highlight that X-CLEAVER$_{iter=1}$ is able to generate models statistically equivalent to the reference λ-MART one, having a much smaller number of trees: 300 trees (-75%) for MSLR-WEB30K-F1 and 800 trees (-47%) for Istella-S. This means that learning a larger number of trees and then pruning them with our QUALITY-LOSS strategy provide the subsequent line search procedure with a set of effective trees to be optimized. Pruning and re-weighting are thus able to boost each other in building compact and effective ranking models.

Finally, we address the main research question of this work, i.e., whether it is effective to embed pruning and re-weighting strategies within an ensemble learning algorithm. To answer this question we measure the performance of different X-CLEAVER models obtained by varying its hyper-parameters, i.e., the step size n, ranging in $\{200, 400\}$, and the pruning rate, ranging in $\{50\%, 75\%\}$. To provide some additional insights, we also evaluate X-CLEAVER with $n = 100$ and no pruning.

When no pruning is performed and line search applied after the generation of every bunch of 100 trees, no significant improvement is observed. In particular, for larger models, it is better to learn a λ-MART model and eventually apply line search just once. This again confirms the benefit of blending pruning and re-weighting altogether.

Table 5.4: Comparison in terms of *NDCG*@10 among λ-MART and the different X-CLEAVER models, by varying p, n and N. Values in bold highlight the best performing model for each ensemble size, while light-gray cells highlight models performing equivalently or better than the reference λ-MART model. The * symbol labels the smallest model for each X-CLEAVER setting resulting to be statistically equivalent to the reference one.

MSLR-WEB30K-F1

Strategy	Trees per Iter (n)	Pruning rate (p)	Pruned Model Size (N) 100	200	300	400	500	600	700	800	900	1000	1199	Training Time
λ-MART	-	-	.4540	.4694	.4745	.4752	.4766	.4774	.4779	.4784	.4785	.4789	.4791	18h 12m
λ-MART + LineSearch	-	-	.4646	.4730	.4765	.4776	.4783*	.4789	.4790	.4776	.4785	.4789	-	
X-CLEAVER$_{iter=1}$	$2N$	50%	.4733	.4767	.4792*	.4786	.4795	.4798	-	-	-	-	-	
X-CLEAVER	100	0%	.4643	.4734	.4750	.4770	.4775	.4777*	.4782	.4784	.4785	.4785	-	17h 42m
	200	50%	.4741	.4783*	.4797	.4801	.4807	.4803	.4807	.4810	.4809	.4810	-	34h 45m
	200	75%	**.4762**	.4781*	.4799	.4808	.4815	**.4825**	**.4829**	**.4828**	.4830	.4830	-	66h 57m
	400	50%	-	.4770	-	.4809	-	.4818	-	.4822	-	.4828	-	31h 01m
	400	75%	.4745	**.4790***	**.4809**	**.4810**	**.4822**	.4823	.4828	.4831	**.4834**	**.4841**	-	62h 23m
X-CLEAVER$_G$	400	75%	.4745	.4773	.4798*	.4799	.4804	.4801	.4811	.4818	.4829	.4816	-	69h 55m

Istella-S

Strategy	Trees per Iter (n)	Pruning rate (p)	Pruned Model Size (N) 100	200	300	400	500	600	700	800	900	1000	1497	Training Time
λ-MART	-	-	.6988	.7239	.7343	.7397	.7433	.7451	.7467	.7478	.7491	.7495	.7530	8h 21m
λ-MART + LineSearch	-	-	.7121	.7342	.7417	.7454	.7473	.7485	.7498	.7499	.7510	.7509	-	
X-CLEAVER$_{iter=1}$	$2N$	50%	.7332	.7439	.7469	.7480	.7507	.7514	.7515	.7533*	-	-	-	
X-CLEAVER	100	0%	.7123	.7371	.7403	.7464	.7463	.7474	.7479	.7491	.7491	.7493	-	8h 47m
	200	50%	.7340	.7418	.7493	.7512	.7523*	.7545	.7545	.7551	.7550	.7551	-	16h 06m
	200	75%	.7375	.7464	**.7513**	**.7542***	**.7553**	**.7549**	.7544	.7550	.7552	.7551	-	37h 10m
	400	50%	-	.7438	-	.7504	-	.7545	-	.7556	-	.7575	-	13h 20m
	400	75%	**.7399**	**.7469**	.7497	.7514	.7530*	.7541	**.7546**	**.7557**	**.7566**	**.7577**	-	26h 11m
X-CLEAVER$_G$	400	75%	**.7399**	.7455	.7480	.7504	.7527*	.7545	.7548	.7563	-	-	-	31h 26m

When pruning is applied with a larger step size, results improve dramatically. X-CLEAVER always outperforms λ-MART models of the same size, even after line search, and it is able to provide better models than the reference λ-MART model for several pruning rates highlighted in the table by light-gray cells. In particular, X-CLEAVER achieves a *NDCG*@10 of 0.4790 on MSLR-WEB30K-F1 with only 200 trees, 83% less than the full reference model, and of 0.7542 on Istella-S with 400 trees, 73% less then the reference model. The best performing models are usually obtained with a pruning rate $p = 75\%$. This is in accordance with the experiments shown in Section 5.4.1, where QUALITY-LOSS is able to prune up to 70% of the original ensemble made up of 500 trees without loosing effectiveness. Using this pruning rate, the X-CLEAVER best models outperform the λ-MART ones with a *NDCG*@10 of 0.4841 (compared to 0.4791) on MSLR-WEB30K-F1 and 0.07577 (compared to 0.7530) on Istella-S.

Finally, we also test a variant of X-CLEAVER, employing the weight optimization step in a global fashion. We call it X-CLEAVER$_G$. Unlike the original X-CLEAVER, which applies line search and pruning only to the new trees trained at each iteration, X-CLEAVER$_G$ works by pruning and re-weighting the entire model at each iteration. Results show that this strategy achieve worse results than X-CLEAVER on both MSLR-WEB30K-F1 and Istella-S, proving the benefits of the local optimization strategy over a global one.

In summary, Figure 5.5 reports the *NDCG*@10 performance of the best performing X-CLEAVER model on the Istella-S dataset (i.e., $N = 1000$, $p = 75\%$ and $n = 400$), compared to that of the reference λ-MART model. The *NDCG*@10 values achieved by the two models is measured at every 100 trees. X-CLEAVER outperforms λ-MART for every ensemble size and the results indicate that the gap between the two models is considerable. Despite providing an increased effectiveness with respect to the reference model, the efficiency is as well increased, since X-CLEAVER is able to achieve the same performance of λ-MART with a reduced number of trees.

To conclude, Table 5.5 compares the actual per-document scoring time of different X-CLEAVER and λ-MART models. The scoring time is measured by exploiting QUICKSCORER which is the state-of-the-art algorithm for the evaluation of regression tree forests [100, 55]. As expected, the speed-up provided by X-CLEAVER compact models is proportional to the reduction in their size. On the MSLR-WEB30K-F1 dataset, the reference λ-MART models has $1,199$ trees and requires about 22.33 μs to score a document, while X-CLEAVER can produce a model of only 200 trees with the same effectiveness but being four times faster at scoring time. Similar results

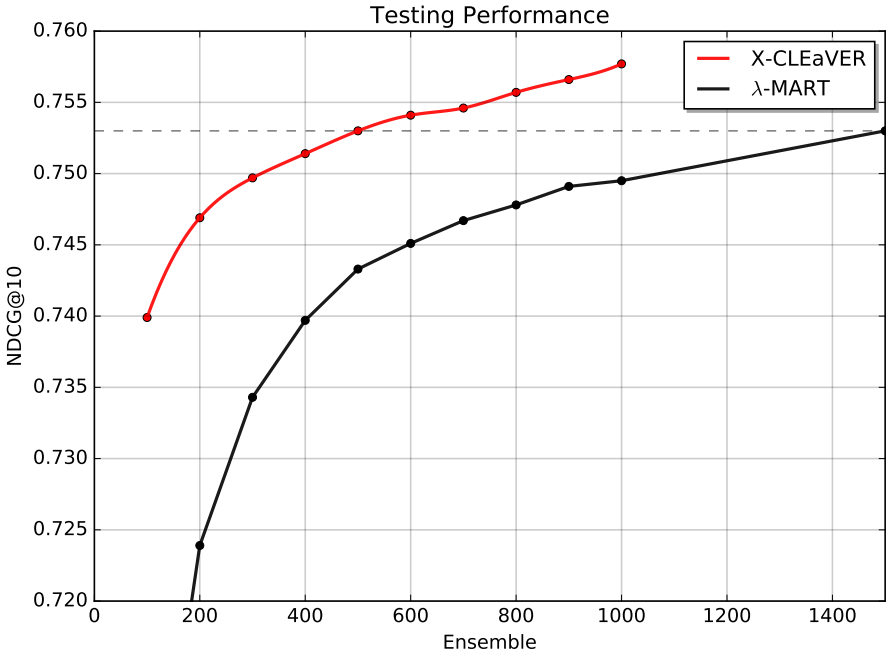

Figure 5.5: Comparison of X-CLEaVER and λ-MART effectiveness in terms of *NDCG*@10 on the Istella-S dataset. The hyper-parameters adopted are $p = 75\%$ and $n = 400$.

are achieved on the Istella-S dataset with a 3.1 speed-up. When considering the X-CLEaVER model of $1,000$ trees, despite showing a higher effectiveness in terms of *NDCG* with respect to the reference model, it proves to be also more efficient by a speed-up factor of 1.1 and 1.5 on MSLR-WEB30K-F1 and Istella-S, respectively.

5.4.4 Training behavior

We further analyze the training behavior of X-CLEaVER across multiple iterations thus investigating the impact of the pruning and re-weighting strategies. In Figure 5.6 we report the *NDCG*@10 achieved by X-CLEaVER on the training set as a function of the number of trees generated. In detail, the solid lines show the performance of λ-MART in performing the GrowModel phase devoted at learning a delta model $\widehat{\mathcal{E}}$, composed of n *weak rankers*. This phase ends at circle blue points, when the learning of a given delta model $\widehat{\mathcal{E}}$ ends, annotated by blue numbers that identify the X-CLEaVER iteration number. Then, X-CLEaVER executes the Prune and ReWeight phases, which lead to a model $\widehat{\mathcal{E}}_W$, corresponding to square red points labeled with the same iteration number. For instance, during the first iteration, X-CLEaVER uses λ-MART to train 400 trees from scratch, and achieves a *NDCG*@10

Table 5.5: Per document scoring time of λ-MART and X-CLEAVER ($n = 400$,$p = 75\%$) models.

MSLR-WEB30K-F1

Algorithm	# Trees	*NDCG*@10	Time (μs.)	Speed-up
λ-MART	1,199	0.4791	22.33	–
X-CLEAVER	1,000	0.4841	20.35	1.1x
	200	0.4790	4.92	4.1x

Istella-S

Algorithm	# Trees	*NDCG*@10	Time (μs.)	Speed-up
λ-MART	1,497	0.7530	48.33	–
X-CLEAVER	1,000	0.7577	31.87	1.5x
	500	0.7530	15.64	3.1x

of 0.7724. This point is identified by the blue number 1. Then, the pruning strategy discards 300 out of the 400 trees, and line search re-weights the remaining ones, leading to a *NDCG*@10 of 0.7821 identified by the red number 1. The gaps between the corresponding blue and red numbers (0.0097 in this example) highlights the ability of X-CLEAVER to remove the less relevant trees and optimize the metric measure *NDCG*@10. Red numbers identify the performance of the X-CLEAVER model at the end of every iteration, thus describing the performance of the intermediate models. Each subsequent iteration starts from the pruned and re-weighted model obtained at the end of the previous iteration, i.e., $\mathcal{E} = \mathcal{E} \cup \widehat{\mathcal{E}}_W$, shown as a new solid line of a different color.

Lastly, the dashed black line identifies the performance achieved by the reference λ-MART algorithm. Since both X-CLEAVER and λ-MART start from scratch at the beginning, they initially train the same 400 trees. The horizontal dashed black line, instead, represents the performance of the reference λ-MART model on the training set.

An interesting effect deserving attention is the performance of the λ-MART algorithm when it restarts the training of a previously optimized model to produce a new delta model $\widehat{\mathcal{E}}$. Indeed, the first trees of the delta model, added to the optimized model $\mathcal{E}$ learnt so far, cause a significant drop of the performance on the training set. This is reflected by the descending direction of the training curve, at the beginning of each iteration (except the first one). The reason of this unexpected behavior can be explained by one of the motivation behind this work, i.e., λ-MART does not directly

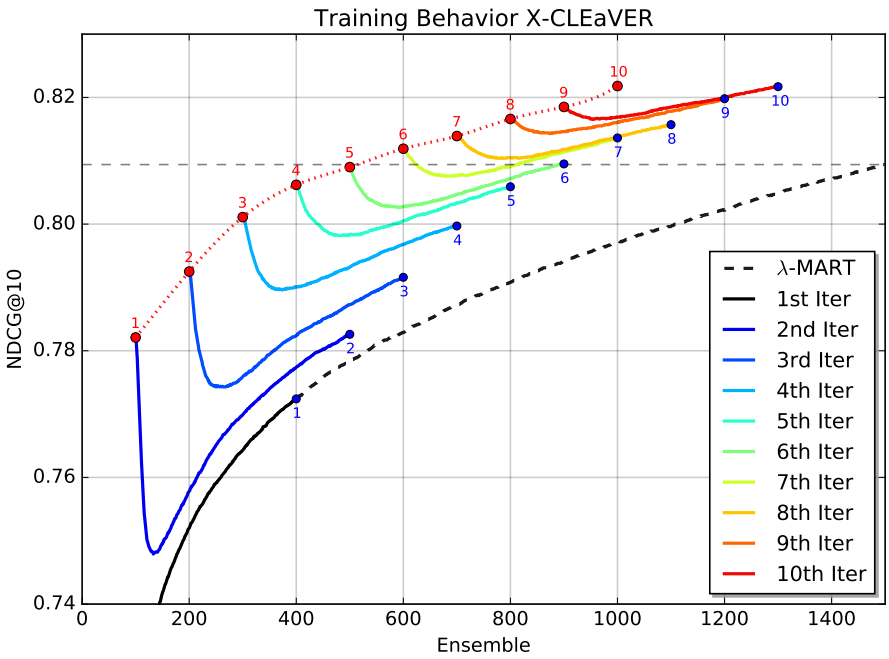

Figure 5.6: Values of *NDCG*@10 measured on the training dataset (Istella-S) across multiple iterations of X-CLEAVER. The hyper-parameters adopted are $p = 75\%$ and $n = 400$.

optimize the given ranking measure, e.g., *NDCG*, but can only indirectly optimize it by using the proxy λ-ranks. Apparently, the models produced at the end of each X-CLEAVER iteration contrast with the learning direction of λ-MART.

However, the performance drop never falls under the training performance of the previous X-CLEAVER iteration when considering same-sized models. In fact, each X-CLEAVER iteration brings significant improvements both before and after the local optimizations. This behavior deserves future investigations as it may shed light on novel optimization strategies for rank-based loss functions. As preliminary conclusion, we believe that the X-CLEAVER local optimizations have the effect of "teleporting" λ-MART on a different regions of the gradient boosting search space that would not be explored otherwise. This allows to generate novel trees, some of them apparently introducing a performance drop and some of them providing a significant improvement with respect to the reference λ-MART. Eventually, the local optimizations of X-CLEAVER select the most relevant trees and re-weight them properly. We thus believe that X-CLEAVER benefits from a wider exploration of the search space.

5.4.5 Training cost analysis

The last column of Table 5.4 reports the training times of the various strategies, referred to the largest model trained. We recall that the training time is an off-line cost: it is worthwhile to pay an extra cost at training time if better or more efficient models can be exploited at prediction time. Overall, the best performing model of X-CLEAVER shows a training time up to 3.5 times slower than the reference λ-MART model, i.e., 62 hours against 18 for MSLR-WEB30K-F1, and 26 hours against 8 for Istella-S. The X-CLEAVER cost is mainly due to the larger number of trees trained: in order to build a model of $N = 1,000$ trees with a pruning rate of $p = 75\%$, the number of trees that we have to actually generate is 4,000. However, X-CLEAVER rewards this additional training cost with more accurate ranking predictions and more faster models.

We further investigate the training time of X-CLEAVER by considering the four main phases of the algorithm: i) training a new delta model $\widehat{\mathcal{E}}$ of n weak rankers using the λ-MART model; ii) computing score predictions $s_i(q, d)$ for every document in the training dataset and for every weak ranker in $\widehat{\mathcal{E}}$, thus allowing a more efficient implementation of the pruning and re-weighting processes; iii) pruning a fraction p of weak rankers according to the QUALITY-LOSS strategy; iv) re-weighting the remaining trees via line search. We measure the fraction time spent by X-CLEAVER across those phases in order to train the best performing model on the Istella-S dataset (i.e., $N = 1,000$, $p = 75\%$, and $n = 400$).

As shown in Figure 5.7, the λ-MART training accounts for about 2/3 of the total time. Learning the weak learners is the most demanding phase. Recall that in order to train 1,000 trees, a total of 4,000 trees are generated across the 10 iterations of the algorithm. The scoring phase has a non-trivial cost of about 15%. This is due to the large number of documents and trees generated during each iteration. Note that X-CLEAVER exploits λ-MART as a black box algorithm, and therefore every generated tree is re-evaluated. Otherwise, the process could be optimized by storing the tree predictions as they are learnt. Pruning has a similar cost of about 13%. Note that QUALITY-LOSS is the most expensive pruning strategy proposed. Nevertheless, the overall cost is relatively small and well balances the higher quality provided in comparison to the other pruning strategies. Finally, the cost of line search optimization is limited to about 8%. Such limited cost is achieved thanks to several optimizations, including multi-threading exploitation.

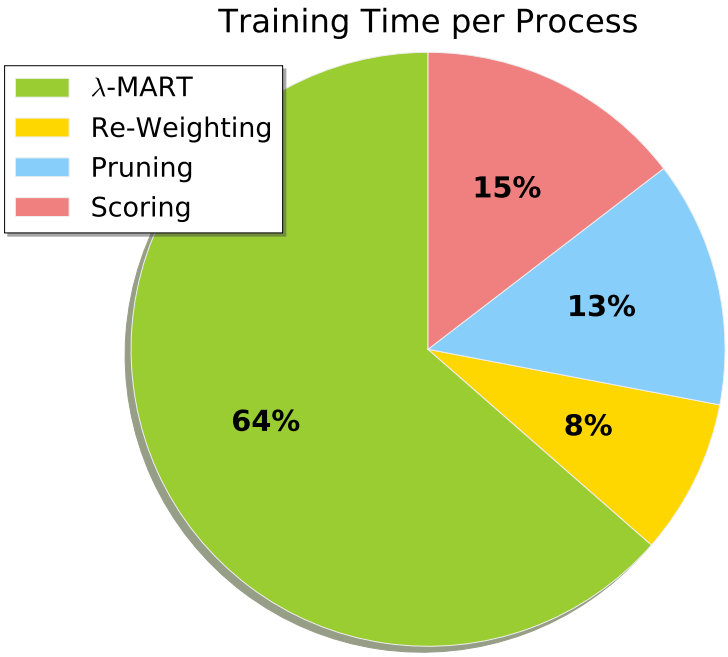

Figure 5.7: X-CLEAVER computational cost breakdown. The hyper-parameters adopted are $p = 75\%$ and $n = 400$ on the Istella-S dataset.

The additional cost introduced by X-CLEAVER is not small. The local optimizations cover 1/3 of the total cost, and a much larger number of trees is generated to build the final model.

However, if we focus on models with similar ranking quality, we observe that the total training time of X-CLEAVER is comparable to that of λ-MART. When considering the X-CLEAVER algorithm with parameters $p = 75\%$ and $n = 400$, it is sufficient to train 200 trees on MSLR-WEB30K-F1 to achieve similar performance to the reference λ-MART model with 1,199 trees, and 500 trees are sufficient on Istella-S to equal the reference λ-MART model with 1,497 trees. In these cases, the training times of X-CLEAVER are of about 11 hours on both datasets, while λ-MART requires 18 hours on MSLR-WEB30K-F1 and 8 on Istella-S.

We thus conclude that X-CLEAVER has a training cost comparable with the state-of-the-art λ-MART algorithm. In addition, it is able to *i)* create much more compact models providing the same ranking quality, and *ii)* learn models with better ranking quality at the cost of some additional computation at training time.

5.5 QuickRank - Learning to Rank Framework

In the latest years a number of machine learning algorithms have been proposed to automatically learn high-quality ranking functions from labeled training sets. This task is particularly challenging for large-scale Web collections, since it requires to meet strict requirements both in terms of effectiveness and efficiency. The former requirement is valued in terms of ranking quality, while the latter is somehow more complex. Indeed traditional Web Search Engine (*WSE*) update their index very often, in order to include fresh and novel documents. Consequently the training time of an LtR model is an important factor to consider. On the other hand scoring time (i.e., the time for producing a prediction) is as much as important, since thousand of documents need to be scored in sub-second time when a *SE* has to produce the result page for a user query.

For these reasons, we propose QuickRank[9], a modular and extensible LtR framework addressing both the learning and the scoring processes. The framework is written in C++ (using C++11) and it exploits OpenMP to provide multithreaded implementation of several state-of-the-art LtR algorithms: GBRT (also known as MART) [65], λ-MART [178], and O-λ-MART [150], as well as the *Line-Search* [162] gready optimization process (for the tree weights), CLEAVER [98] and X-CLEAVER. QuickRank takes the dataset in the SVM-light format, and save the machine-learnt ranking model in a XML file. Regarding the scoring process, the framework supports various strategies. The first is named If-Then-Else, and aims at translate each decision tree in a sequence of C++ if-then-else blocks, trying to take advantage of compiler optimization strategies. The second implements the Vpred [7] strategy, and aims at reducing the control hazards of the previous approach caused by the large number of conditional branches. This strategy translate the ranking model in the format required by the Vpred implementation. The third strategy applies only to O-λ-MART, and exploit the property of these trees to be balanced to use a bitmask for the comparison.

5.6 Conclusion

In this chapter, we proposed X-CLEAVER, a novel meta-learning algorithm that embed pruning and weight optimization strategies within an ensemble learning algorithm providing compact models with higher quality than state-of-the-art solutions. We

[9]The source code is available at https://github.com/hpclab/quickrank

proposed several pruning strategies to identify the less relevant trees to remove at each iteration, and we formalized a greedy optimization process devoted at maximizing a given ranking loss function such as *NDCG*, for which it is not possible to adopt the classical gradient descent strategy since it is not derivable. The experiments conducted on two public LtR dataset confirmed the validity of the pruning and weights optimization strategies, at first when tested in isolation, and then when nested in the proposed iterative X-CLEAVER meta-algorithm. The resulting model exhibits improvements in terms of both efficiency and effectiveness. Indeed, compared to the reference λ-MART model, it allowed the training of smaller models showing similar effectiveness or more effective models fixing the ensemble size. We believe this is due to the somehow heuristic nature of learning algorithms such as λ-MART. As ranking measure cannot be optimized directly, proxy measures are used (e.g., lambda gradients). X-CLEAVER allows i) to select a subset of relevant tree from a large candidate set (pruning) and ii) to fine-tune theirs weights (re-weighting), with both steps being driven by the optimization of the target ranking function. At each iteration, X-CLEAVER provides a solid set of ranker from which the λ-MART algorithm can continue its optimization.

Chapter 6

Conclusions and Future Work

Web search represents the most advanced Information Retrieval application to date. Besides the efficiency challenges, caused by the need to manage the exponential growth in the number of web pages and the huge, ever increasing, number of users, WSEs are expected to return precise and accurate results for any query to provide a very high quality of experience to their users. To this end, one of the biggest problem they need to face derives from the ambiguity of queries due, for example to polysemy and synonymy. Indeed, both documents and queries are expressed natural language, with the former written by a multitude of heterogeneous users with different writing styles and cultures, and the latter being inherently ambiguous due to their short length.

Several document and query understanding techniques have been thus developed to achieve a finer comprehension of the main topics and the leading characters covered. The most promising of such techniques is represented by *Entity Linking*, a task aimed at identifying the short fragments of text referring to an entity that is listed in a given knowledge base. The annotation process leads to a richer representation of both queries and documents, which are enriched with links to the semantic concepts they refer to. The net result is a paradigm shift, moving from unstructured to semi-structured queries and documents, that simplifies understanding and query-document matching, thus enabling the possibility of deriving insights from the data.

The statement of this thesis is that "*a semantically enriched learning-to-rank strategy can simultaneously improve both efficiency and effectiveness of document understanding and retrieval tasks*". To this end, in Chapter 3 we proposed a novel entity relatedness measure by formalizing entity-relatedness as a learning-to-rank problem. In Chapter 4 we formalized a novel algorithm comprehensively addressing both Entity Linking and Saliency detection. The two works together enhanced the effectiveness of document understanding with respect to state-of-the-art annotation systems. The improvements were achieved with the usage of supervised machine learning techniques

that produce additive ensembles of decision trees. These techniques received a lot of attention by the IR community in the latest years due to the high quality of the ranking models ensemble-based LtR techniques allow to learn from labeled datasets. Thus, Chapter 5 focuses on ensemble-based models and proposes a novel meta-learning LtR algorithm that embeds tree pruning and re-weighting in order to obtain more efficient and effective ranking models.

The remainder of this chapter is organized as follows. In Section 6.1 we summarize the main contributions of this thesis, while in Section 6.2 we critical discuss several limitations of this dissertation, explaining how important each of these limitation is, justifying the choices made during the research progress and possibly suggesting how to overcome such limitations in future. Finally, in Section 6.3 we present the list of publications produced during this PhD programme.

6.1 Thesis Contributions and Future Work

In the following we summarize the main contributions of this thesis and discuss possible directions for future research.

Learning Relatedness Measures for Entity Linking In Chapter 3 we investigated *entity relatedness*, a measure aimed at providing the degree of similarity between two entities belonging to a given Knowledge Base. This measure is probably the most important feature adopted by state-of-the-art *Entity Linking* systems in performing the disambiguation, i.e., in selecting among all the candidate entities identified by the spotting for a given fragment of text, the most relevant one. To this end, a common approach is to adopt a global optimization strategy, i.e., to maximize the coherence among the selected entities. The definition of an effective relatedness function is thus a crucial point in any entity-linking algorithm. We addressed the problem proposing a machine learning approach aimed at discovering the entity relatedness function. In Section 3.2 we formalized the problem of learning entity relatedness as a learning-to-rank problem. We then proposed in Section 3.4 a methodology to create reference datasets on the basis of manually annotated data and discussed the features adopted. The experiments showed that our machine-learnt entity relatedness function performs better than other relatedness functions previously proposed. Finally, in Section 3.5 we proved that we can remarkably improve the overall performance of state-of-the-art *Entity Linking* algorithms by sinlpy embedding the proposed relatedness measure within them. The proposed solution opens up a wide spectrum of improvement opportunities. In particular, it could be interesting to investigate the trade-off between

feature computational cost and benefit provided, and the introduction of novel and more powerful features (e.g., entity popularity, categories, similarity between the description of the two entities, i.e., the text of their Wikipedia pages. Moreover, it could be worth investigating the usage of popular word embedding techniques, trained on the Wikipedia corpus in such a way to consider not only the text but also the existing links between the entities. This technique could be adopted in place of our proposed relatedness measure, or as a feature.

SEL: A Unified Algorithm for Entity Linking and Saliency Detection. In Chapter 4 we suggested to go beyond the traditional definition of *Entity Linking*. Indeed, it is straightforward that not all the entities mentioned in a document have the same relevance and utility in understanding the topics being discussed. Thus, the related problem of identifying the most relevant entities present in a document, also known as Salient Entities, is of fundamental importance for a fine-grained document understanding. To this purpose, in Section 4.3 we gave a formal definition of the Salient Entity Linking problem and we proposed SEL, a novel supervised two-step algorithm comprehensively addressing both entity linking and saliency detection. The first step is based on a classifier aimed at identifying a set of candidate entities that are likely to be mentioned in the document, thus maximizing the precision of the method without hindering its recall. The second step is still based on machine learning, and aims at choosing from the previous set the entities that actually occur in the document. Indeed, we tested two different versions of the second step, one aimed at solving only the entity linking task, and the other that, besides detecting linked entities, also scores them according to their saliency. In Section 4.4, we presented the methodology for creating a novel dataset of news manually annotated with entities and their saliency. Experiments conducted on two different datasets show that the proposed algorithm outperforms state-of-the-art competitors, and is able to detect salient entities with high accuracy. Furthermore, we investigated the usage of Entity Saliency for the document summarization problem, where exploiting the saliency of the entities is crucial for providing a high-level summary of the document. The resulting summarizers outperform well-known summarization systems, proving the importance of using the Salient Entities information.

To the best of our knowledge, this is the first work addressing the Salient Entity detection problem in conjunction with the Entity Linking. For this reason, several research directions remain open. One of this future directions is to exploits the information about the saliency of the entity in evaluating the effectiveness of an *Entity Linking* system. The idea is that a system that fails to annotate one of the

most salient entities should score lower that a system that fails to annotate satellite concepts. Alternative directions include studying how Entity Saliency impacts on several information extraction tasks, i.e., text summarization or document clustering. In these application scenario, the capability of weighting entities on the basis of their saliency is crucial, as empirically proved by our preliminary experiments on the former task. To this end, further investigation is required to study the role of entity-based features in enhancing state-of-the-art extractive summarizers.

Embedding Tree Pruning and Re-Weighting in Learning to Rank. In Chapter 5 we approached the problem of improving efficiency and effectivess of ensemble-based LtR models. These models are trained using a boosting strategy, i.e., an iterative approach that combines many weighted weak tree-based rankers into a single model. At each iteration the tree that minimizes a given loss function is added to the ensemble. By analyzing the models generated by such state-of-the-art algorithms, we observed some degree of similarity between the trees, i.e., consecutive trees tend to provide similar predictions. We thus proposed several pruning strategies that, starting from a given ensemble model, remove the less relevant trees with the goal of minimizing the effectiveness drop but increasing the model efficiency. The efficiency of ensemble-based models, indeed, is proportional to the number of trees in the ensemble, thus reducing their number results in a lower scoring time. While pruning the ensemble provide evident benefits in terms of efficiency, on the other hand optimizing tree weights leads to improve the effectiveness. Indeed, state-of-the-art algorithms do not consider this aspect and use a uniform weighting schema. We instead proposed a greedy optimization algorithm aimed at finding the best weights assignment as to maximize a given loss function, i.e., *NDCG*. Finally, we combined the pruning and weight optimization strategies in a single meta-algorithm, named X-CLEAVER, which is able to iteratively prune the less useful part of the ensemble and re-weight the remaining part accordingly to the loss function adopted, in such a way to optimize both efficiency and effectiveness with respect to the original model. The experiments conducted on two public LtR datasets confirmed that the pruning and weights optimization strategies in isolation were responsible for producing more compact models than state-of-the-art method, although exhibiting a similar ranking quality. Further experiments executed on the iterative meta-algorithm proved the validity of the interleaved approach, resulting in the training of more efficient and more effective models.

To the best of our knowledge, this is the first work addressing the learning of ranking models from both the perspectives at the same time: efficiency and effectiveness. To this regards, several research directions remains to investigate. On the one side, the

usage of different LtR algorithms, other than λ-MART, within the proposed approach. Indeed, X-CLEAVER is designed to work in principle with every LtR algorithm, since the training algorithm is used as a black box. For instance, it could be interesting to adopts a MART model, i.e., a point-wise regression algorithm quite popular in the LtR field, and study the joint effects of minimizing the Root Mean Squared Error (RMSE) (MART) and the *NDCG* on the full list (the X-CLEAVER optimization strategies). On the other side, it could be interesting to investigate the application of the proposed methodology to tasks different from the ranking, like regression and classification. Indeed, several state-of-the-art regression and classification algorithms adopts the same ensemble-based models, built in an additive way. To this regards, optimizing a loss function different from the ranking loss (e.g., the regression loss or the classification loss) could lead to interesting results, since these functions are usually derivable unlike *NDCG*. Lastly, it could be interesting to adopt the weight optimization process on a per tree basis, differently from the X-CLEAVER approach where the greedy optimization is done on a set of trees (thus increasing the difficulty of the local search, due to an higher number of parameters, but on the contrary allowing more space for improvements, since the local search can explore a bigger space).

Open-Source Frameworks developed as product of the research activity. In this section we list several open source frameworks, mainly developed for validating the novel solutions proposed in this thesis and then made publicly available to the research community. In Section 3.6 we presented DEXTER, a framework addressing the *Entity Linking* problem and providing out of the box the implementation of several state-of-the-art algorithms. It has been designed optimizing the modularity of its components (e.g., single modules for spotting, disambiguation and filtering) and the resource requirements, such that it can run also on commodity hardware. DEXTER constituted the building block of all the experiments presented in this thesis, and has been actively used also in other research activities (e.g., European and national projects). In Section 4.6 we presented ELIANTO, a framework aimed at crowd-sourcing the production of manually annotated datasets for the Entity Linking and Saliency Detection tasks. ELIANTO has been mainly developed for creating the dataset for the Entity Saliency work, since the missing of public datasets with saliency annotations. As DEXTER, also this framework has been extensively used in parallel research activities. Lastly, Section 5.5 introduce QUICKRANK, a framework designed for efficiently addressing the learning to rank task as well as the scoring process, using a modular and extensible architecture. Despite the framework has been originally developed by other researchers, in the scope of this thesis it has been

extensively expanded and reshaped for incorporating new ideas and optimizations. The X-CLEAVER approach is now completely integrated within QUICKRANK.

6.2 Research Limitations

Various limitations may exist in this study. The first aspect we want to discuss is the lack of an empirical demonstration that improving the effectiveness of document understanding will lead to an improvement of the retrieval quality in Web search. Several works in literature treated about semantic search, approaching the problem from a twofold angles: supporting the semantic retrieval of documents and enriching web queries with concepts of a Knowledge Base (KB). Section 2.1.3 provides some details on this argument as well as references to various interesting works the reader might follow up. Researchers often face a barrier when working on this topic due to the lack of a standard large-scale benchmark collection of annotated documents to use for testing the effectiveness of the proposed solutions. Indeed, most past studies use small datasets and small corpus and thus it is unclear whether similar conclusions would be reached using a full-stack architecture. Interestingly, researchers from Yahoo observed, on their commercial web search engine using a full-web corpus, a significant boost to the relevance of search results by enriching queries with concepts [97]. Due to the aforementioned considerations, we state that such a demonstration is hard to reach without a direct connection with a commercial web search engine and thus it is beyond the scope of this thesis.

A second aspect to discuss is the lack of evidence that the solution proposed in this dissertation for improving the document understanding task are not only effective but also efficient. To this regard, it is useful to recall that each one of these solutions, despite adopting traditional state-of-the-art learning model, have also been analyzed with regard to the feature space. Indeed, a feature selection study has been done to find the best trade-off between efficiency and effectiveness of the resulting model. Moreover, the learning to rank model proposed in Chapter 5, which is designed to learn a much more compact ensemble-based forest of decision trees with respect to state-of-the-art model, is applicable also to the solutions above mentioned. Thus to summarize, the net result is that we attacked the efficiency problem by considering both the feature space and the compactness of the model, with the latter that linearly affect the scoring time of such ensemble-based models.

Several other aspects are worth to be considered, despite being less important. Some of them are strictly related to the Entity Linking task. One aspect for example

is the NIL problem, i.e., the ability of the Entity Linking systems to recognize fragments of text referring to concepts that are not in the *KB*. This problem is of paramount importance when dealing with fresh documents like news, due to an increased probability of dealing with entities not yet in the *KB*. KB freshness, i.e., the capability of staying as much as possible in sync with the *KB*, is another problem particular important in this context. Indeed, the intermediate data structures (e.g., indexes) used by the Entity Linking systems need to be updated usually by a batch process that is time and memory consuming. Due to the effort needed for the update, these systems adopt often information from *KB* old several months and sometimes even years. Neither the NIL problem nor KB Freshness have been tackled in this dissertation, but could significantly enhance the effectiveness of the document understanding task.

Another aspect do not considered in this dissertation is the multi-language dependency, and in particular the ability for the Entity Linking system to adopts on the fly the joint information obtained by exploiting the interconnections between datasets provided in different languages. Indeed Wikipedia and others *KB* have different editions for each language, but they also provide with the connections between related pages in the distinct languages, thus creating a global *KB* that can be fruitfully exploited for improving the knowledge about the entities. Furthermore, this dissertation focus on using the Wikipedia *KB*, but it could be interesting to study the effect of adopting different *KB* with regard to the topic of the documents to annotate. For example, the following research question rise: i) is Wikipedia the best general purpose *KB* ? ii) when dealing with document of a specific topic, is still Wikipedia good enough or using different *KB* could lead to significant improvements to the annotation quality ? iii) how difficult is to joint together two or more *KB* in order to create a unique, huge collection of entities ? These questions are known to the researchers working on this topic and several attempts has been done to provide at least partially answer to them. In particular with regard to the second RQ, we tried a first, rudimentary approach to study the effect of using a subset of the original *KB* to enrich documents of a very specific topic, i.e., philosophical documents [34]. This preliminary work highlighted than the proposed solution provides higher effectiveness with respect to the usage of the full *KB*. However the experiments where done on a small dataset manually annotated by some experts, thus it is difficult to generalize the outcome of this work.

6.3 List of Publications

In this section we lists all the references to papers – published or still under revision – produced in the course of this PhD programme. The list of the papers is organized accordingly to the chapters of this thesis, in order to reflect the contributions presented on a topic basis. Moreover, an additional paper is included in the list, covering a preliminary work aimed at studying the effectiveness of an Entity Linking system in enriching topic-based documents, i.e., philosophical texts. In this work, the focused on restricting the original Knowledge Base (Wikipedia) in such a way to include only topic-related entities, using different strategies. The experiments performed on a small set of manually assessed documents proved the benefits of this approach, thus confirming that a general purpose Knowledge Base is problematic when the task is to annotate topic-specific texts.

Chapter 3 - Learning relatedness measures for entity linking

Learning relatedness measures for entity linking Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. *In Proceedings of the 22nd ACM international conference on Information & Knowledge Management, pp. 139-148. (ACM CIKM), Burlingame, CA, USA October 27th - November 1st, 2013.*

Chapter 4 - SEL: A Unified Algorithm for Entity Linking and Saliency Detection

SEL: A Unified Algorithm for Entity Linking and Saliency Detection[1] Salvatore Trani, Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. *In Proceedings of the 2016 ACM Symposium on Document Engineering, pp. 85-94. (ACM DocEng)* Vienna, Austria September 13 - September 16, 2016.

SEL: A Unified Algorithm for Entity Linking and Saliency Detection Diego Ceccarelli, David E. Losada, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. *Submitted for review to an International Journal*;

[1] **Awarded with the ACM DocEng best student paper.**

Chapter 5 - Embedding Tree Pruning and Re-Weighting in Learning to Rank

Improve Ranking Efficiency by Optimizing Tree Ensembles Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. *Extended abstract in 7th Italian Information Retrieval Workshop (IIR)* Venice, Italy May 20 - May 31, 2016.

Post-learning optimization of tree ensembles for efficient ranking Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. *In Proceedings of the 39th International ACM conference on Research and Development in Information Retrieval, pp. 949-952. (ACM SIGIR)* Pisa, Italy July 17 - July 21, 2016.

X-CLEAVER**: Learning Ranking Ensembles by Growing and Pruning Trees** Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. *Submitted for review to a top tier International Journal*;

Frameworks

Dexter: an open source framework for entity linking Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego and Salvatore Trani. *In Proceedings of the 6th International Workshop on Exploiting Semantic Annotations in Information Retrieval, pp. 17-20. (ESAIR)* San Francisco, CA, USA October 28, 2013.

Dexter 2.0: an open source tool for semantically enriching data Salvatore Trani, Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. *In Proceedings of the 13th International Semantic Web Conference Posters & Demonstrations Track, p. 417-420. (ISWC-PD)* Riva del Garda, Italy October 19 - October 23, 2014.

Manual annotation of semi-structured documents for entity-linking Salvatore Trani, Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, and Raffaele Perego *In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 2075-2077. (ACM CIKM)* Shanghai, China November 03 - November 7, 2014.

Other Publications

Entity Linking on Philosophical Documents Diego Ceccarelli, Alberto De Francesco, Raffaele Perego, Marco Segala, Nicola Tonellotto and Salvatore Trani *6th Italian Information Retrieval Workshop (IIR)* Cagliari, Italy May 25 - May 26, 2015.

Bibliography

[1] Shivani Agarwal. Ranking on graph data. In *Proceedings of the 23rd international conference on Machine learning*, pages 25–32. ACM, 2006. 25

[2] Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26. ACM, 2006. 20, 30

[3] Marco Aiello, Christof Monz, Leon Todoran, and Marcel Worring. Document understanding for a broad class of documents. *International Journal on Document Analysis and Recognition*, 5(1):1–16, 2002. 15

[4] Omar Alonso, Daniel E Rose, and Benjamin Stewart. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum*, volume 42, pages 9–15. ACM, 2008. 29

[5] Jesse Alpert and Nissan Hajaj. We knew the web was big. *The official Google blog*, 25, 2008. 1

[6] Nima Asadi and Jimmy Lin. Training efficient tree-based models for document ranking. In *Advances in Information Retrieval*, pages 146–157. Springer, 2013. 98, 102

[7] Nima Asadi, Jimmy Lin, and Arjen P. de Vries. Runtime optimizations for tree-based machine learning models. *IEEE Trans. Knowl. Data Eng.*, 26(9):2281–2292, 2014. 98, 103, 125

[8] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007. 12

[9] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *International Conference on Extending Database Technology*, pages 588–596. Springer, 2004. 2

[10] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999. 1, 2, 17, 19

[11] Gordon Bell, Tony Hey, and Alex Szalay. Beyond the data deluge. *Science*, 323(5919):1297–1298, 2009. 1

[12] Michael Bendersky, W Bruce Croft, and Yanlei Diao. Quality-biased ranking of web documents. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 95–104. ACM, 2011. 20

[13] Tim Berners-Lee. Linked data-design issues (2006). *URL http://www. w3. org/DesignIssues/LinkedData. html*, 2011. 10, 11

[14] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001. 3, 10

[15] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009. 3

[16] Roi Blanco, Harry Halpin, Daniel M Herzig, Peter Mika, Jeffrey Pound, Henry S Thompson, and Thanh Tran Duc. Repeatable and reliable search system evaluation using crowdsourcing. In *Proceedings of SIGIR*, pages 923–932. ACM, 2011. 75

[17] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008. 13

[18] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984. 24

[19] Marc Bron, Krisztian Balog, and Maarten de Rijke. Ranking related entities: components and analyses. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 1079–1088, 2010. 43

[20] Jake Brutlag. Speed matters for google web search. *Google. June*, 2009. 2, 9

[21] Michael Buckland and Fredric Gey. The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12, 1994. 31

[22] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005. 24, 25

[23] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581, 2010. 25

[24] Christopher JC Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *NIPS*, volume 6, pages 193–200, 2006. 25

[25] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002. 17

[26] Berkant Barla Cambazoglu and Ricardo A. Baeza-Yates. *Scalability Challenges in Web Search Engines*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2015. 2, 98, 104

[27] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193. ACM, 2006. 24

[28] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007. 25

[29] Gabriele Capannini, Domenico Dato, Claudio Lucchese, Monica Mori, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. QuickRank: a C++ suite of learning to rank algorithms (http://quickrank.isti.cnr.it/). *IIR '15: 6th Italian Information Retrieval Workshop*, 2015. 110, 111

[30] Gabriele Capannini, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. Quality versus efficiency in document scoring with learning-to-rank models. *Information Processing & Management*, 2016. 7, 98, 99, 103, 111

[31] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336, New York, NY, USA, 1998. ACM. 85

[32] S Jeromy Carrière and Rick Kazman. Webquery: Searching and visualizing the web through connectivity. *Computer Networks and ISDN Systems*, 29(8):1257–1267, 1997. 20

[33] Vitor R Carvalho, Matthew Lease, and Emine Yilmaz. Crowdsourcing for search evaluation. In *ACM Sigir forum*, volume 44, pages 17–22. ACM, 2011. 29

[34] Diego Ceccarelli, Alberto De Francesco, Raffaele Perego, Marco Segala, Nicola Tonellotto, and Salvatore Trani. Entity linking on philosophical documents. In *IIR*, 2015. 133

[35] Diego Ceccarelli, Sergiu Gordea, Claudio Lucchese, Franco Maria Nardini, and Raffale Perego. When entities meet query recommender systems: semantic search shortcuts. In *Proceedings of SAC*, 2013. 43

[36] Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. Dexter: an open source framework for entity linking. In *ESAIR'13, Proceedings of the Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval, co-located with CIKM 2013, San Francisco, CA, USA, October 28, 2013*, pages 17–20, 2013. 7, 52

[37] Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. Learning relatedness measures for entity linking. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 139–148. ACM, 2013. 6

[38] Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. Dexter 2.0 - an open source tool for semantically enriching data. In *Proceedings of the 13th International Semantic Web Conference, Posters & Demonstrations Track (ISWC-PD)*, pages 417–420, 2014. 7, 52

[39] Soumen Chakrabarti, Sasidhar Kasturi, Bharath Balakrishnan, Ganesh Ramakrishnan, and Rohit Saraf. Compressed data structures for annotated web search. In *Proceedings of WWW*, 2012. 16

[40] Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, NIPS'09, pages 315–323, 2009. 100

[41] Gong Cheng and Yuzhong Qu. Searching linked objects with falcons: Approach, implementation and evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):49–70, 2009. 14

[42] Xiao Cheng and Dan Roth. Relational inference for wikification. *Urbana*, 51:61801, 2013. 16, 63

[43] Rudi Cilibrasi and Paul M. B. Vitányi. The google similarity distance. *IEEE Trans. Knowl. Data Eng.*, 19(3):370–383, 2007. 42

[44] Charles LA Clarke, Nick Craswell, and Ian Soboroff. Overview of the trec 2004 terabyte track. In *TREC*, volume 4, page 74, 2004. 27

[45] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666. ACM, 2008. 2

[46] Cyril Cleverdon. The cranfield tests on index language devices. In *Aslib proceedings*, volume 19, pages 173–194. MCB UP Ltd, 1967. 29

[47] Cyril Cleverdon. On the inverse relationship of recall and precision. *Journal of documentation*, 28(3):195–201, 1972. 31

[48] Gordon V. Cormack, Mark D. Smucker, and Charles L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, 14(5):441–465, 2011. 17

[49] David Cossock and Tong Zhang. Statistical analysis of bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008. 26

[50] Koby Crammer, Yoram Singer, et al. Pranking with ranking. In *Nips*, volume 1, pages 641–647, 2001. 24

[51] Nick Craswell, David Hawking, Ross Wilkinson, and Mingfang Wu. Overview of the trec 2003 web track. In *TREC*, volume 3, page 12th, 2003. 27

[52] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 708–716, 2007. 16, 34, 43

[53] Matt Cutts. Spotlight keynote. *Proceedings of Search Engine Strategies*, 2012. 1

[54] Van Dang, Michael Bendersky, and W Bruce Croft. Two-stage learning to rank for information retrieval. In *Advances in Information Retrieval*, pages 423–434. Springer, 2013. 104

[55] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM Transactions on Information Systems*, 2016. 27, 103, 119

[56] Maurice de Kunder. Worldwidewebsize.com - the size of the world wide web (the internet), 2015. 1

[57] Ofer Dekel, Christopher D Manning, and Yoram Singer. Log-linear models for label ranking. In *NIPS*, volume 16, 2003. 25

[58] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. Entity disambiguation for knowledge base population. In *Proceedings of COLING*, 2010. 16, 43

[59] Jesse Dunietz and Dan Gillick. A new entity salience task with millions of training examples. *EACL 2014*, page 205, 2014. 64, 75

[60] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.(JAIR)*, 22(1):457–479, 2004. 73

[61] Paolo Ferragina and Ugo Scaiella. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario,*

Canada, October 26-30, 2010, pages 1625–1628, 2010. 16, 34, 37, 40, 42, 43, 57, 62, 67

[62] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. Technical report, 1999. 11

[63] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003. 24, 25

[64] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995. 25

[65] Jerome Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001. 24, 111, 125

[66] Jerome Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 2001. 47

[67] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of IJCAI*, 2007. 43

[68] Michael Gamon, Tae Yano, Xinying Song, Johnson Apacible, and Patrick Pantel. Identifying salient entities in web pages. In *Proceedings of CIKM*, pages 2375–2380. ACM, 2013. 6, 60, 63

[69] Xiubo Geng, Tie-Yan Liu, Tao Qin, and Hang Li. Feature selection for ranking. In *Proceedings of SIGIR 2007*, 2007. 48

[70] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Inferring web communities from link topology. In *Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space—structure in hypermedia systems: links, objects, time and space—structure in hypermedia systems*, pages 225–234. ACM, 1998. 10

[71] Michael Gordon and Manfred Kochen. Recall-precision trade-off: A derivation. *Journal of the American Society for Information Science*, 40(3):145, 1989. 31

[72] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R Curran. Evaluating entity linking with wikipedia. *Artificial intelligence*, 2013. 54

[73] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of SIGIR*, 2011. 16, 37, 40, 41, 42, 43, 57

[74] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011. 3

[75] MH Heine. Inverse relationship of precision and recall in terms of swets model, 1973. 31

[76] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. *Advances in neural information processing systems*, pages 115–132, 1999. 23, 25, 101

[77] Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of CIKM*, 2012. 34, 43

[78] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 782–792, 2011. 16, 37, 45, 63, 67, 73

[79] Ian Jacobs and Norman Walsh. Architecture of the world wide web. 2004. 11

[80] Bernard J Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. Real life information retrieval: A study of user queries on the web. In *ACM SIGIR Forum*, volume 32, pages 5–17. ACM, 1998. 2

[81] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 2002. 32, 39

[82] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD*, 2002. 39

[83] Thorsten Joachims et al. Evaluating retrieval performance using clickthrough data., 2003. 20, 30

[84] Sparck Jones. Report on the need for and provision of an" ideal" information retrieval test collection. 1975. 29

[85] Allen Kent, Madeline M Berry, Fred U Luehrs, and James W Perry. Machine literature searching viii. operational criteria for designing information retrieval systems. *American documentation*, 6(2):93–101, 1955. 30

[86] Rohit Khare and Tantek Çelik. Microformats: a pragmatic path to the semantic web. In *Proceedings of the 15th international conference on World Wide Web*, pages 865–866. ACM, 2006. 13, 15

[87] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49–79, 2004. 14

[88] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999. 10, 20

[89] Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006. 11

[90] Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2007. 24, 26

[91] Yanhong Li. Toward a qualitative search engine. *IEEE Internet Computing*, 2(4):24, 1998. 20

[92] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. 87

[93] Thomas Lin, Oren Etzioni, et al. No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 893–903. Association for Computational Linguistics, 2012. 16

[94] Chao Liu, Ryen W White, and Susan Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 379–386. ACM, 2010. 20, 30

[95] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. 22, 23, 98

[96] David E. Losada and Javier Parapar. Injecting multiple psychological features into standard text summarisers. In *Proceedings of the 4th Spanish Conference on Information Retrieval*, CERI '16, pages 3:1–3:8, New York, NY, USA, 2016. ACM. 88

[97] Yumao Lu, Fuchun Peng, Gilad Mishne, Xing Wei, and Benoit Dumoulin. Improving web search relevance with semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 648–657. Association for Computational Linguistics, 2009. 132

[98] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. Post-learning optimization of tree ensembles for efficient ranking. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 949–952, 2016. 7, 27, 98, 107, 110, 111, 125

[99] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Salvatore Trani. X-cleaver: a new learning to rank algorithm for efficient ranking. *Manuscript submitted for review to a Top Tier Journal*, 2017. 7

[100] Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. Quickscorer: A fast algorithm to rank documents with additive ensembles of regression trees. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 73–82. ACM, 2015. 98, 103, 119

[101] Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957. 18

[102] Craig Macdonald, Rodrygo LT Santos, and Iadh Ounis. The whens and hows of learning to rank for web search. *Information Retrieval*, 16(5):584–628, 2013. 98

[103] Christoph Mangold. A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, 2(1):23–34, 2007. 14

[104] Inderjeet Mani. *Automatic Summarization*. John Benjamins Publishing Company, 2001. 83

[105] Larry Masinter, Tim Berners-Lee, and Roy T Fielding. Uniform resource identifier (uri): Generic syntax. 2005. 11

[106] Olena Medelyan, Ian H Witten, and David Milne. Topic indexing with wikipedia. In *Proceedings of the AAAI WikiAI workshop*, volume 1, pages 19–24, 2008. 14

[107] Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. Mdl-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, KDD'95, pages 216–221. AAAI Press, 1995. 101

[108] Edgar Meij, Marc Bron, Laura Hollink, Bouke Huurnink, and Maarten De Rijke. Learning semantic query suggestions. In *International Semantic Web Conference*, pages 424–440. Springer, 2009. 14

[109] Edgar Meij, Marc Bron, Laura Hollink, Bouke Huurnink, and Maarten de Rijke. Mapping queries to the linking open data cloud: A case study using dbpedia. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):418–433, 2011. 14

[110] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM, 2011. 16, 62

[111] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 233–242, 2007. 15, 36, 61, 62

[112] Peter Mika, Edgar Meij, and Hugo Zaragoza. Investigating the semantic gap through query log analysis. In *International Semantic Web Conference*, pages 441–455. Springer, 2009. 14

[113] Andrei Mikheev, Marc Moens, and Claire Grover. Named entity recognition without gazetteers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 1–8. Association for Computational Linguistics, 1999. 16

[114] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 13

[115] David Milne and Ian H. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *In Proceedings of AAAI*, 2008. 6, 37, 43, 45

[116] David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of CIKM*, 2008. 6, 16, 34, 37, 40, 41, 42, 43, 46, 57, 61, 62, 67, 72, 86

[117] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007. 16

[118] Feng Nan, Joseph Wang, and Venkatesh Saligrama. Pruning random forests for prediction on a budget. *Advances in neural information processing systems*, 2016. 102

[119] Ani Nenkova. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *AAAI*, volume 5, pages 1436–1441, 2005. 15

[120] Ani Nenkova and Kathleen McKeown. A survey of text summarization techniques. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 43–76. Springer, 2012. 83

[121] NIST. The trec nist website. *URL http://trec.nist.gov*. 29

[122] Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. Detecting spam web pages through content analysis. In *Proceedings of the 15th international conference on World Wide Web*, pages 83–92. ACM, 2006. 20

[123] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello. Sindice. com: a document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1):37–52, 2008. 4

[124] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999. 10, 21, 41

[125] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008. 17

[126] Patrick Pantel and Ariel Fuxman. Jigs and lures: Associating web queries with structured entities. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 83–92, 2011. 3

[127] Deepa Paranjpe. Learning document aboutness from implicit user feedback and document structure. In *Proceedings of CIKM*, 2009. 6, 60, 63, 75

[128] Francesco Piccinno and Paolo Ferragina. From tagme to wat: a new entity annotator. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 55–62. ACM, 2014. 16, 63

[129] Chao Qian, Jing-Cheng Shi, Yang Yu, Ke Tang, and Zhi-Hua Zhou. Parallel pareto optimization for subset selection. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, USA*, pages 1939–1945, 2016. 102

[130] Chao Qian, Yang Yu, and Zhi-Hua Zhou. Pareto ensemble pruning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2935–2941. AAAI Press, 2015. 101

[131] Tao Qin, Xu-Dong Zhang, De-Sheng Wang, Tie-Yan Liu, Wei Lai, and Hang Li. Ranking with multiple hyperplanes. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 279–286. ACM, 2007. 27

[132] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986. 101

[133] Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. MEAD – A platform for multidocument multilingual text summarization. In *Conference on Language Resources and Evaluation (LREC)*, Lisbon, Portugal, 2004. 84

[134] Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248. ACM, 2005. 26

[135] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 43–52. ACM, 2008. 20, 30

[136] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, 2009. 16

[137] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics, 2011. 16, 63

[138] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. Global refinement of random forest. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 101

[139] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520, 2004. 18

[140] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49. ACM, 2004. 19

[141] Stephen E Robertson. The probability ranking principle in ir. *Journal of documentation*, 33(4):294–304, 1977. 19, 29

[142] Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976. 19

[143] Stephen E Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at trec. In *Text retrieval conference*, pages 21–30, 1993. 19

[144] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, 109:109, 1995. 19

[145] Henning Rode, Pavel Serdyukov, Djoerd Hiemstra, and Hugo Zaragoza. Entity ranking on graphs: Studies on expert finding. 2007. 63, 64

[146] Cynthia Rudin. Ranking with a p-norm push. In *International Conference on Computational Learning Theory*, pages 589–604. Springer, 2006. 26

[147] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. 18

[148] Mark Sanderson. *Test collection based evaluation of information retrieval systems*. Now Publishers Inc, 2010. 29, 30

[149] William W Cohen Robert E Schapire and Yoram Singer. Learning to order things. *Advances in Neural Information Processing Systems*, 10:451, 1998. 24

[150] Ilya Segalovich. Machine learning in search quality at yandex. *Invited Talk, SIGIR*, 2010. 125

[151] Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *Knowledge and Data Engineering, IEEE Transactions on*, 27(2):443–460, 2015. 16, 63

[152] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of WWW*, 2012. 34, 43

[153] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proc. CIKM '07*. ACM, 2007. 112

[154] Ruihua Song, Zhenxiao Luo, Ji-Rong Wen, Yong Yu, and Hsiao-Wuen Hon. Identifying ambiguous queries in web search. In *Proceedings of the 16th international conference on World Wide Web*, pages 1169–1170. ACM, 2007. 2

[155] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972. 18

[156] Karen Sparck Jones and Cornelis Joost van Rijsbergen. Information retrieval test collections. *Journal of documentation*, 32(1):59–75, 1976. 29

[157] Wolfgang G Stock. Concepts and semantic relations in information science. *Journal of the American Society for Information Science and Technology*, 61(10):1951–1969, 2010. 14

[158] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007. 13

[159] Yizhou Sun, Yintao Yu, and Jiawei Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 797–806. ACM, 2009. 17

[160] Yuan Yan Tang, Seong-Whan Lee, and Ching Y. Suen. Automatic document processing: A survey. *Pattern Recognition*, 29(12):1931–1952, 1996. 15

[161] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 77–86. ACM, 2008. 25

[162] Michael Taylor, Hugo Zaragoza, Nick Craswell, Stephen Robertson, and Chris Burges. Optimisation methods for ranking functions with multiple parameters. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 585–593. ACM, 2006. 108, 125

[163] Salvatore Trani, Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. Manual annotation of semi-structured documents for entity-linking. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 2075–2077. ACM, 2014. 7, 74

[164] Salvatore Trani, Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. Sel: A unified algorithm for entity linking and saliency detection. In *Proceedings of the 2016 ACM Symposium on Document Engineering*, pages 85–94. ACM, 2016. 6

[165] Salvatore Trani, Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. Sel: A unified algorithm for entity linking and saliency detection. *Manuscript submitted for review to a Top Tier Journal*, 2017. 6

[166] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru, and Stefan Decker. Sig. ma: Live views on the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):355–364, 2010. 14

[167] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. Learning to rank by optimizing ndcg measure. In *Advances in neural information processing systems*, pages 1883–1891, 2009. 25

[168] Cornelis Joost van Rijsbergen. Information retrieval. *The Information Retrieval Group,¡ http://dsc. glasgow. ac. uk/preface. html*, 1979. 31

[169] Roelof van Zwol, Lluis Garcia Pueyo, Mridul Muralidharan, and Börkur Sigurbjörnsson. Ranking entity facets based on user click feedback. In *Proceedings of the 4th IEEE International Conference on Semantic Computing (ICSC 2010), September 22-24, 2010, Carnegie Mellon University, Pittsburgh, PA, USA*, pages 192–199, 2010. 43

[170] Luc Vincent. Google book search: Document understanding on a massive scale. In *icdar*, pages 819–823, 2007. 15

[171] Ellen M Voorhees. Trec: Continuing information retrieval's tradition of experimentation. *Communications of the ACM*, 50(11):51–54, 2007. 29

[172] Ellen M Voorhees et al. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82, 1999. 17, 31

[173] Ellen M Voorhees, Donna K Harman, et al. *TREC: Experiment and evaluation in information retrieval*, volume 1. MIT press Cambridge, 2005. 29

[174] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. 13

[175] Lidan Wang, Jimmy Lin, and Donald Metzler. Learning to efficiently rank. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 138–145, New York, NY, USA, 2010. ACM. 98, 103

[176] Pu Wang, Jian Hu, Hua-Jun Zeng, and Zheng Chen. Using wikipedia knowledge to improve text classification. *Knowledge and Information Systems*, 19(3):265–281, 2009. 14

[177] Gerhard Weikum and Martin Theobald. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 65–76, 2010. 3

[178] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010. 25, 47, 98, 100, 110, 125

[179] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. AIDA: an online tool for accurate disambiguation of named entities in text and tables. *PVLDB*, 4(12):1450–1453, 2011. 16, 42

[180] Yun Zhou and W Bruce Croft. Document quality models for web ad hoc retrieval. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 331–332. ACM, 2005. 20

[181] Mu Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2, 2004. 31

[182] George Kingsley Zipf. Selected studies of the principle of relative frequency in language. 1932. 18