

Faegheh Hasibi

Semantic Search with Knowledge Bases

Thesis for the degree of Philosophiae Doctor

Trondheim, March 2018

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science



Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Computer Science

© 2018 Faegheh Hasibi. All rights reserved

ISBN 978-82-326-2966-4 (printed version)
ISBN 978-82-326-2967-1 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2018:88

Printed by NTNU-trykk

Abstract

Over the past decade, modern search engines have made significant progress towards better understanding searchers’ intents and providing them with more focused answers, a paradigm that is called “semantic search.” Semantic search is a broad area that encompasses a variety of tasks and has a core enabling data component, called the *knowledge base*. In this thesis, we utilize knowledge bases to address three tasks involved in semantic search: (i) query understanding, (ii) entity retrieval, and (iii) entity summarization.

Query understanding is the first step in virtually every semantic search system. We study the problem of identifying entity mentions in queries and linking them to the corresponding entries in a knowledge base. We formulate this as the task of entity linking in queries, propose refinements to evaluation measures, and publish a test collection for training and evaluation purposes. We further establish a baseline method for this task through a reproducibility study, and introduce different methods with the aim to strike a balance between efficiency and effectiveness.

Next, we turn to using the obtained annotations for answering the queries. Here, our focus is on the entity retrieval task: answering search queries by returning a ranked list of entities. We introduce a general feature-based model based on Markov Random Fields, and show improvements over existing baseline methods. We find that the largest gains are achieved for complex natural language queries.

Having generated an answer to the query (from the entity retrieval step), we move on to presentation aspects of the results. We introduce and address the novel problem of dynamic entity summarization for entity cards, by breaking it into two subtasks, fact ranking and summary generation. We perform an extensive evaluation of our method using crowdsourcing, and show that our supervised fact ranking method brings substantial improvements over the most comparable baselines.

In this thesis, we take the reproducibility of our research very seriously. Therefore, all resources developed within the course of this work are made publicly available. We further make two major software and resource contributions: (i) the Nordlys toolkit, which implements a range of methods for semantic search, and (ii) the extended DBpedia-Entity test collection.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for partial fulfillment of the requirements for the degree Philosophiae Doctor (PhD). The work has been carried out at the Department of Computer Science, NTNU, Trondheim. The doctoral program has been supervised by Professor Svein Erik Bratsberg, Professor Krisztian Balog (University of Stavanger), and Dr. Øystein Torbjørnsen (Microsoft Development Center Norway). During the process, the candidate has been supported by the Information Access Disruptions (iAd) project, funded by the Research Council of Norway and NTNU.

Acknowledgements

This thesis is the end result of a long journey, started back on the first day of my school when I was seven. Many people had affected me during this journey, the most influential ones being: my parents, who raised me with love, care, and a questioning mind; and my lovely husband, the greatest friend of mine, who was emotionally next to me in every step of this thesis and provided me with unconditional support. **This thesis is dedicated to them.**

A number of individuals supported me during the past few years, whom I would like to thank.

To my advisors ...

My first and foremost acknowledgement goes to my advisor, Prof. Svein Erik Bratsberg, who always offered wise advices and gave me the freedom to pursue my research interests. Thanks for always being there and for providing me with incredible support and encouragement. I extend my appreciation to Dr. Øystein Torbjørnsen for his guidance and inspirational meetings during the early stages of my PhD.

My special gratitude goes to Prof. Krisztian Balog, as I was fortunate to work under his eminent guidance since July 2014. Thanks for everything you have done for me. You taught me how to do research, from idea to presentation, and provided me with valuable comments and advices about every aspect of my research. Not only that, your encouragement and friendship was always a source of energy during these years. Thanks for your hospitality during my Stavanger visits, making all the fun moments at the IAI group, giving vision and advice about after-PhD career, and of course teaching me a bit of skiing. Words fall short to express my thanks; I summarize it by saying “you were a great friend and manager who was hard to find, difficult to leave, and impossible to forget.”

To my PhD dissertation committee ...

I am honored and grateful to have Prof. James Allan, Prof. Arjen P. de Vries, and Prof. Kjetil Nørvåg serving as my committee members. I would like to thank them for reviewing this dissertation and for providing me with detailed comments and suggestions.

To my colleagues and collaborators ...

I am thankful to colleagues at the Department of Computer Science, both PhD students, and technical and administrative staff. The help and support from the former PhD graduates of the group, Massimiliano, Naimdjon, Orestis, Robert, and Simon was a gift. In particular, I thank Simon Lia-Jonassen for the enlightening advices in the early stages of my PhD and for providing me with constructive feedback on my first paper. The informal chats and group outings with Dirk, Eliezer, Hoda, Jan, Malene, Shweta, Soudabeh, Stella, Tomasz, and Tárík made my stay at NTNU a memorable experience. Thank you Jan for creating a fun, creative office environment, and for capturing our moments through your wonderful lens.

I further extend my gratitude to the members of the IAI group at the University of Stavanger for the nice company during our group activities and conference trips. I particularly thank Darío and Shuo, who were also my co-authors and collaborators. My acknowledgments also goes to my co-authors from across the ocean for all that I have learned during our discussions.

To my internship hosts ...

I would like to express my deepest appreciation to Dr. B. Barla Cambazoglu for hosting me at the Yahoo! research lab and introducing me to the world of machine learning and entity search. Thanks Ioannis for providing me with useful instructions and many fruitful discussions.

To my friends and family ...

I owe my warm gratitude to my dearest friends, Maryam and Hamid, for their lovely heart and home, which were always open for me during my Stavanger trips. Thanks Maryam for your delicious exotic foods and for our memorable chats during tea-drinking moments.

Most importantly, I convey my heartfelt appreciation to my sisters and brother, who were my childhood friends, and later my great teachers. Thanks for helping me to find my path and sending me your love and care over miles of distance. The traces of your love and support are embedded in this thesis.

Contents

1	Introduction	1
1.1	Knowledge Bases	2
1.2	Research Outline and Questions	3
1.2.1	Query Understanding	4
1.2.2	Entity Retrieval	6
1.2.3	Entity Summarization	7
1.3	Contributions	8
1.4	Thesis Overview	10
1.5	Origins	11
2	Background	15
2.1	Conventional Information Retrieval	15
2.1.1	Document Processing	16
2.1.2	Query Processing	17
2.1.3	Document Retrieval	18
2.1.4	Evaluation	24
2.2	Semantic Search	26
2.2.1	Knowledge Bases	27
2.2.2	Document Processing	29
2.2.3	Query Understanding	31
2.2.4	Retrieval	32
2.2.5	Result presentation	35
3	Query Understanding via Entity Linking	37
3.1	Task Definitions	40
3.1.1	Entity Linking (in Documents)	40
3.1.2	Entity Linking in Queries	40

3.1.3	Semantic Linking	43
3.2	Evaluation Methodology	44
3.3	Test Collections	45
3.3.1	YSQLE	45
3.3.2	ERD	47
3.3.3	Y-ERD	47
3.4	Approaches	50
3.4.1	Mention Detection	51
3.4.2	Candidate Entity Ranking	52
3.4.3	Disambiguation	54
3.5	Experiments	56
3.5.1	Experimental Setup	56
3.5.2	Mention Detection	57
3.5.3	Semantic Linking	58
3.5.4	Entity Linking in Queries	58
3.6	Discussion	59
3.7	Summary	61
4	Establishing a Baseline for Entity Linking in Queries	63
4.1	Overview of TAGME	65
4.1.1	Approach	65
4.1.2	Test Collections	68
4.1.3	Evaluation Measures	69
4.2	Repeatability	70
4.3	Reproducibility	70
4.3.1	Implementation	71
4.3.2	Results	72
4.4	Generalizability	74
4.4.1	Experimental Setup	75
4.4.2	Results	75
4.5	Discussion	76
4.5.1	Further Investigations	76
4.5.2	Lessons Learned	77
4.6	Summary	78
5	Methods for Entity Linking in Queries	79
5.1	Entity Linking in Queries	81
5.1.1	Candidate Entity Ranking	83
5.1.2	Disambiguation	84

5.2	Experimental Setup	86
5.2.1	Data	86
5.2.2	Methods	87
5.2.3	Evaluation	88
5.3	Results and Analysis	88
5.3.1	Results	88
5.3.2	Feature Analysis	92
5.4	Summary	93
6	Exploiting Entity Linking in Queries for Entity Retrieval	95
6.1	Background	98
6.1.1	The Markov Random Field Model	98
6.1.2	Sequential Dependence Model	99
6.1.3	Fielded Sequential Dependence Model	101
6.2	The ELR Approach	101
6.2.1	Model	102
6.2.2	Feature Functions	105
6.2.3	Fielded Representation of Entities	107
6.3	Experimental Setup	108
6.3.1	Data	108
6.3.2	Field Selection	110
6.3.3	Baseline Models	110
6.3.4	Parameter Setting	111
6.3.5	Entity Linking	112
6.4	Results and Analysis	112
6.4.1	Overall Performance	112
6.4.2	Breakdown by Query Subsets	113
6.4.3	Parameter Settings	115
6.4.4	Impact of Entity Linking	117
6.5	Summary	118
7	Dynamic Factual Summaries for Entity Cards	119
7.1	Problem Statement	122
7.2	Approach	124
7.2.1	Fact Ranking	125
7.2.2	Summary Generation	129
7.3	Establishing a Benchmark	131
7.3.1	Data sources	131
7.3.2	Selecting Entity-Query Pairs	132

7.3.3	Fact Ranking Test Set	133
7.4	Fact Ranking Results	134
7.4.1	Settings	134
7.4.2	Experiments	135
7.4.3	Results	136
7.4.4	Feature Analysis	137
7.5	Summary Generation Results	138
7.5.1	Settings	139
7.5.2	Experiments	139
7.5.3	Results	139
7.5.4	Analysis	140
7.6	Summary	142
8	The DBpedia-Entity v2 Test Collection	143
8.1	The Test Collection	144
8.1.1	Test Queries	144
8.1.2	Knowledge Base	145
8.2	The Entity Retrieval Subcollection	145
8.2.1	Constructing the Collection	146
8.2.2	Baseline Methods	149
8.2.3	Results and Analysis	152
8.3	The Target Type Identification Subcollection	153
8.3.1	Constructing the Collection	153
8.3.2	Baseline Methods	155
8.3.3	Results and Analysis	156
8.4	Summary	157
9	Nordlys: A Semantic Search Toolkit	159
9.1	Functionality	160
9.1.1	Entity Catalog	160
9.1.2	Entity Retrieval	161
9.1.3	Entity Linking in Queries	162
9.1.4	Target Type Identification	163
9.2	The Nordlys Toolkit	164
9.2.1	Architecture	164
9.2.2	Usage Modes	165
9.3	Summary	166

10 Conclusions	167
10.1 Main Findings	167
10.2 Future Directions	171
A Resources	173
A.1 Query Understanding via Entity Linking	173
A.2 Establishing a Baseline for Entity Linking in Queries	173
A.3 Methods for Entity Linking in Queries	174
A.4 Exploiting Entity Linking in Queries for Entity Retrieval	174
A.5 Dynamic Factual Summaries for Entity Cards	174
References	175

Chapter 1

Introduction

Over the past decade, web search has undergone a major change, from “ten blue links” towards understanding searchers’ intent and providing them with more focused responses, a paradigm that is referred to as *semantic search*. Google, for instance, announced its attempt of becoming a “knowledge engine” rather than of an “information engine” in 2012 [2]. It currently serves direct answers to a large portion of queries that are centered around entities; an example is shown in Figure 1.1. Apple Siri, Google Now, Microsoft Cortana, and Amazon Alexa are other examples of semantic search systems, where search tasks are performed through a natural language interface.

Semantic search is a broad area that encompasses a variety of tasks and techniques. It has a core enabling data component, called the *knowledge base*. A knowledge base is a structured repository of *entities* (such as people, organizations, and locations), containing information about their attributes and their relationships to other entities. Specifically, in this thesis, we utilize knowledge bases to address three tasks involved in semantic search: (i) query understanding, (ii) entity retrieval, and (iii) entity summarization. Overall, this thesis presents methods, tools, and datasets for performing semantic search over knowledge bases. In the remainder of this chapter, we provide a brief introduction to knowledge bases, followed by a research outline and thesis overview.

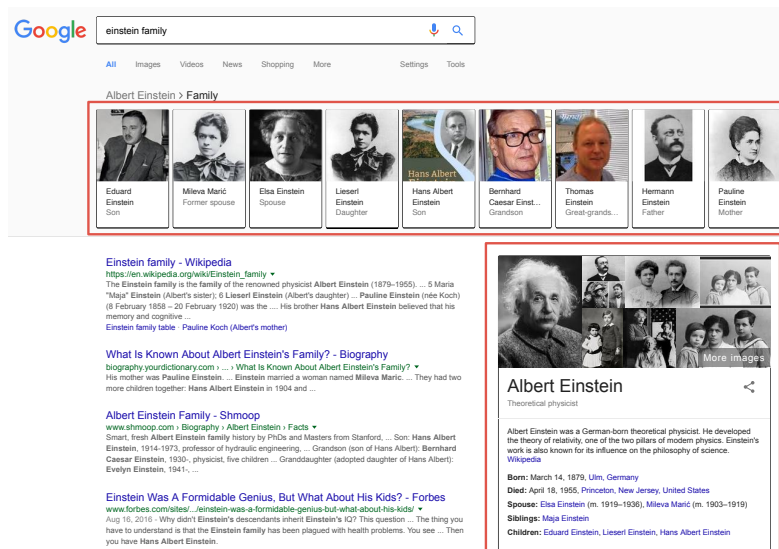


Figure 1.1: Google search results for the query “einstein family.” Besides the ranked list of documents, two other components (extracted from a knowledge base) are presented: (i) *direct answer* (top marked area), and (ii) *entity card*, (bottom marked area).

1.1 Knowledge Bases

Knowledge bases (KBs) organize information about entities in a machine-readable form. Entities are “uniquely identifiable objects or things (such as persons, organizations, and places), characterized by their types, attributes, and relationships to other entities” [10]. Each entity in the knowledge base is represented by a unique identifier, and the facts and relationships of entities are stored in a standard form. This structured form of knowledge is the underlying idea behind the Semantic Web [24]. KBs also enable answering semantically rich queries. Take, for example, the query “female german politicians,” which searches for entities of *type* politician, with *gender* attribute of female, and *birth place* of Germany.

Search systems such as Google [63], Bing,¹ Facebook,² and LinkedIn³ build and use their own knowledge bases to answer such queries.

General purpose knowledge bases (often built based on encyclopedic knowledge) got their boost in the late 2000s; prominent examples include Freebase [30], Wikidata [191], YAGO [180], and DBpedia [6]. Freebase, launched in 2007, was the largest publicly available knowledge base, and was built based on the data collected from Wikipedia as well as from community contributions. With the shut down of Freebase in 2014, the content of Freebase was set to be transferred to Wikidata, which is another collaborative knowledge base by the Wikimedia Foundation [155]. YAGO, on the other hand, is an automatically generated knowledge base, which combines information from Wikipedia, WordNet and GeoNames.⁴ DBpedia is the most popular and prominent knowledge base and is considered as a central interlinking hub in the Linked Open Data (LOD)⁵ cloud [6]. It is built based on a framework that extracts structured information from Wikipedia articles and turns it into a knowledge base.

Each entity in a knowledge base is associated with a number of facts. The entity facts are often stored as RDF triples `<subject, predicate, object>`, where the `object` is a literal or a URI (link to another entity); see for example Figure 1.2. This structured representation of entities has various benefits for semantic search systems. It enables processing and answering of complex queries, such as “founder of intel” or “physicists of the 20th century.” It can also be used to generate succinct summaries of entities, including key facts and related entities, as often seen in entity cards (see Figure 1.1). In the course of this thesis, we incorporate DBpedia as our main knowledge base and use (same-as) links to Freebase whenever needed. As a notational convenience, we shall typeset entities in small caps, e.g., ALBERT EINSTEIN.

1.2 Research Outline and Questions

The central research theme governing this thesis is addressing semantic search tasks through knowledge bases. To this end, we first focus on understanding users’ information needs by semantically annotating search queries. We then

¹<https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/>

²<http://www.insidefacebook.com/2013/01/14/facebook-builds-knowledge-graph-with-info-modules-on-community-pages/>

³<https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph>

⁴<http://www.geonames.org/>

⁵<http://linkeddata.org/>


 ALBERT EINSTEIN	
foaf:name	Albert Einstein
rdfs:comment	Albert Einstein was a German-born theoretical physicist who developed the general theory of relativity, one of the two pillars of modern physics.[...]
dbo:birthDate	1879-03-14
dbo:birthPlace	dbpedia:Ulm dbpedia:German_Empire dbpedia:Kingdom_of_Württemberg
dbo:deathDate	1955-04-18
dbo:deathPlace	dbpedia:Princeton,_New_Jersey dbpedia:United_States
dbp:shortDescription	dbpedia:Physicist
dbp:doctoralAdvisor	dbpedia:Alfred_Kleiner
dbp:ethnicity	Jewish
dbp:residence	Germany Italy Switzerland Austria Belgium United states

Figure 1.2: An excerpt from the DBpedia knowledge base for the entity ALBERT EINSTEIN; i.e., predicate-object pairs for triples, where the subject is `dbpedia:Albert_Einstein`.

incorporate these semantic annotations to improve result ranking. Finally, we concentrate on result presentation, and specifically on summarizing entity facts for “entity cards,” the knowledge panels that are typically presented at the right column of search engine result pages. Below we detail each of these tasks, together with the research questions we address.

1.2.1 Query Understanding

In many search systems, users express their information needs using free text, i.e., keyword or natural language queries. It is then the search engine’s task to go beyond the query terms and identify the underlying intent of the queries, the process that is referred to as *query understanding* [55]. This process is the first step in virtually every search scenario, and is of great importance in semantic search systems. We therefore dedicated a great deal of attention to this specific task.

The general problem of query understanding has been approached from a number of different angles, just to mention a few: segmenting queries into meaningful phrases [23, 84, 181], recognizing named entities in queries [81], identifying the target type(s) of queries [11], and annotating queries with specific

entities [43]. Among these, entity annotation of queries has received significant attention recently [75], and has been incorporated in various information retrieval tasks [58, 175, 197, 205].

Identifying entity mentions in text and linking them to the corresponding entries in a reference knowledge base is referred to as the problem of *entity linking in queries* (ELQ). While there is a large body of work on entity linking in documents, entity linking in queries poses new challenges due to the limited context the query provides, coupled with the efficiency requirements of an online setting. A fundamental difference between these two tasks is that entity mentions in queries cannot always be disambiguated, and a mention can possibly be linked to multiple entities. Consider, for example, the query “france world cup 98,” in which the term “france,” can be linked to two entities: FRANCE and FRANCE NATIONAL FOOTBALL TEAM. This calls for a methodological departure from the traditional entity linking task. We start by asking the following research question:

RQ1 How can the inherent ambiguity of entity annotations in queries be handled and evaluated?

We discuss different ways to deal with query ambiguity and establish entity linking in queries as the task of finding entity linking interpretation(s). We propose refinements in the evaluation measures and introduce a test collection for this task. We then move on to establishing a baseline for this task. Our next research question is:

RQ2 How does a state-of-the-art entity linking approach perform on the ELQ task?

We select TAGME [69] as a state-of-the-art entity linking system and show systematically that it can be generalized to the task of entity linking in queries. After establishing the task, the evaluation measures, the test collection, and the baseline, we turn to developing methods specifically for the ELQ task. We ask:

RQ3 How should entity linking in queries be performed to achieve high efficiency and effectiveness?

We divide the ELQ task into two main steps: (i) candidate entity ranking and (ii) disambiguation, and explore both unsupervised and supervised alternatives for each step. The resulting four combinations of the methods are then compared to each other with respect to efficiency and effectiveness. Based on an extensive analysis, we provide general recommendations for effective and efficient entity linking in queries.

1.2.2 Entity Retrieval

Once we shed some light on how to understand queries via entity annotations, we turn to using these annotations for answering the queries. The task that we focus on here is *entity retrieval*: answering search queries by returning a ranked list of entities. The basic premise of entity retrieval is that some information needs are better answered by returning specific entities instead of just any type of documents [137]. Take for example the queries “eiffel,” “vietnam war movies,” or “Apollo astronauts who walked on the Moon,” which clearly look for specific entities. Figure 1.1 illustrates how entity retrieval is used in a commercial search engine to provide direct answers; see the top marked area.

There is an extensive body of work on entity retrieval, with a broad variety of models and algorithms for matching query terms against entities; see e.g., [27, 47, 111, 147, 148, 157, 210]. We postulate that entity retrieval can be improved by incorporating semantic annotations of the queries into the retrieval model. Some prior work performed in the context of the INEX-XER [59] evaluation campaign lends credence to our assumption [15, 107, 162]; these studies use target entity types of queries, provided by searchers as part of the definitions of information needs. We, on the other hand, aim to leverage automatically extracted entity annotations of queries into the retrieval model. This is a particularly challenging task due to the inherent uncertainty involved with entity annotations, and leads us to the following research question:

RQ4 How to exploit entity annotations of queries in entity retrieval?

The answer to this question should be a model that can handle different types of queries, including named entity queries (e.g., “Madrid”), keyword queries (e.g., “Szechwan dish food cuisine”), list queries (e.g., “Formula one races in Europe”), and natural language queries (e.g., “Which organizations were founded in 1950?”). Entity annotations may contribute differently in answering different types of queries: while named entity queries seek a particular entity that is mentioned in the query, other types of queries are often about some specific relation or attribute of the mentioned entity. We develop a generative model that can be applied on top of several term-based models and can deal with the heterogeneity of queries.

1.2.3 Entity Summarization

As the last research topic of this thesis, we move on to presentation aspects of search results. Specifically, we focus on generating content for entity cards, which are being used frequently in modern web search engines (see Figure 1.1). Entity cards are presented in response to an entity-bearing query and offer a concise overview of an entity directly on the results page. It has been shown that entity cards, regardless of their topic, can increase searcher engagement with organic web search results [33]. When these cards are relevant to the query, users can more easily find the answers they were looking for and can accomplish their tasks faster [116].

Entity cards are composed of various elements, one of them being the entity summary: a selection of facts describing the entity from an underlying knowledge base. Our goal is to generate these summaries, the task that is referred to as *entity summarization*. Summaries on entity cards play a dual role: (i) they provide a succinct overview of entity facts from the knowledge base, and (ii) they directly address users’ information needs. Consider for example, the summary of ALBERT EINSTEIN for two queries “einstein family” and “einstein awards”: one should be about Albert Einstein’s awards and education, and the other about his family. A particular challenge in generating such summaries is selecting both important and relevant facts. A summary of only important facts is *diverse*, but may not reflect users’ information need. On the other hand, a summary containing only relevant facts to the query is *biased* and fails in providing a concise overview of the entity. This calls for striking a balance between query bias and fact diversity in generating the entity summaries. Additionally, rendering the summary on an entity card is more involved than simply displaying a ranked list of facts, and some presentation aspects need to be addressed. Our leading research question in this area is:

RQ5 How to generate and evaluate factual summaries for entity cards?

Addressing this research question involves a precise formulation of the problem, followed by developing methods for generating summaries and for evaluating them. We formulate two subtasks of *fact ranking* and *summary generation*, where the former ranks facts with respect to importance and/or relevance, and the latter renders ranked facts as a summary to be displayed on the entity card. The evaluation methodology of the fact ranking subtask includes generating a test collection via crowdsourcing, and comparing it to ranked entity facts generated by our proposed approach (using standard rank-based measures). For evaluating the summary generation subtask, we perform side-by-side evaluation of generated summaries via crowdsourcing and assess their quality by counting user preferences.

1.3 Contributions

We now summarize the main contributions of this thesis, which fall into two categories: (i) theoretical and empirical contributions, and (ii) software and resource contributions.

Theoretical and Empirical Contributions

C1 Revisiting the task definition, evaluation measures, and test collections for entity linking in queries.

We differentiate between the two seemingly similar tasks of *semantic linking*, and *entity linking in queries*, discuss current evaluation methodology, and propose refinements. We examine publicly available datasets for these tasks and introduce a new manually curated dataset for entity linking in queries. To further deepen the understanding of task differences, we present a set of approaches for effectively addressing these tasks, and report on experimental results.

C2 Establishing a baseline for entity linking in queries from a state-of-the-art entity linking approach.

We study the TAGME [69] entity linking system through a reproducibility study and provide insights on how it can be used as a baseline for entity linking in queries. Within this study, we also formulate lessons learned

about reproducibility, which were employed in the rest of the research in this thesis.

C3 Presenting methods for efficient and effective entity linking in queries.

We propose four different methods for entity linking in queries, and compare them with respect to both efficiency and effectiveness. Based on an extensive analysis, we show that the best overall performance (in terms of efficiency and effectiveness) can be achieved by employing supervised learning for ranking entities, and further tackling the entity disambiguation by an unsupervised algorithm.

C4 Introducing a theoretical framework for incorporating entity annotations of queries into entity retrieval models.

We present a framework that brings entity retrieval and entity linking together and formulate a new probabilistic component that can be applied on top of any term-based retrieval model that can be instantiated as a Markov Random Field model. Our approach outperforms existing state-of-the-art ad hoc entity retrieval models by over 6%, and is especially beneficial for complex and heterogeneous queries, improving performance by up to 16% in terms of MAP (all improvements are relative).

C5 Presenting methods to generate and evaluate dynamic factual summaries for entity cards.

We introduce and formalize the task of dynamic entity summarization for entity cards, and break it down to two specific subtasks: fact ranking and summary generation. Our proposed method for fact ranking brings over 16% relative improvement over the most comparable state-of-the-art baseline in terms of NDCG@10. We perform an extensive user preference evaluations and show that users prefer dynamic summaries (containing both important and relevant facts) over static importance- or relevance-based summaries. We further show that presentation aspects have a major impact on the perceived quality of summaries and thus these deserve further attention.

Software and Resource Contributions

C6 DBpedia-Entity test collection.

We create and publish the DBpedia-Entity v2 test collection. This data set consists of around 500 heterogeneous queries, which were previously

synthesized by Balog and Neumayer [12] from different evaluation campaigns (DBpedia-Entity v1). Relevance judgments are collected for three semantic search tasks: entity retrieval, entity summarization, and target type identification.

C7 Nordlys toolkit.

We develop and introduce *Nordlys*, a toolkit that implements a range of methods for entity-oriented and semantic search, including the ones presented in this thesis. It is available as a Python library, as a command line tool, as a RESTful API, and as a graphical web user interface.

1.4 Thesis Overview

This section presents the outline of the thesis, followed by reading direction. As illustrated by Figure 1.3, the thesis starts with the introduction and background chapters, followed by seven technical chapters (Chapters 3–9), and concluding remarks in Chapter 10.

Chapters 1–2 provide introduction and background for the rest of the thesis. Chapter 1 includes motivation, research questions, contributions, and origins of the thesis. Chapter 2 reviews the fundamental concepts of information retrieval and the state-of-the-art on semantic search.

Chapters 3–5 cover our work related to query understanding. Chapter 3 discusses the task of entity linking in queries and evaluation measures. Chapter 4 establishes a baseline for the task of entity linking in queries, and Chapter 5 describes the proposed methods for addressing this task.

Chapter 6 presents a method for incorporating entity annotations of queries into the retrieval process.

Chapter 7 investigates generating query-dependent summaries for entity cards.

Chapters 8–9 describe the resources developed in the course of this research. Chapter 8 presents the DBpedia-Entity v2 test collection, while Chapter 9 details Nordlys, our semantic search toolkit.

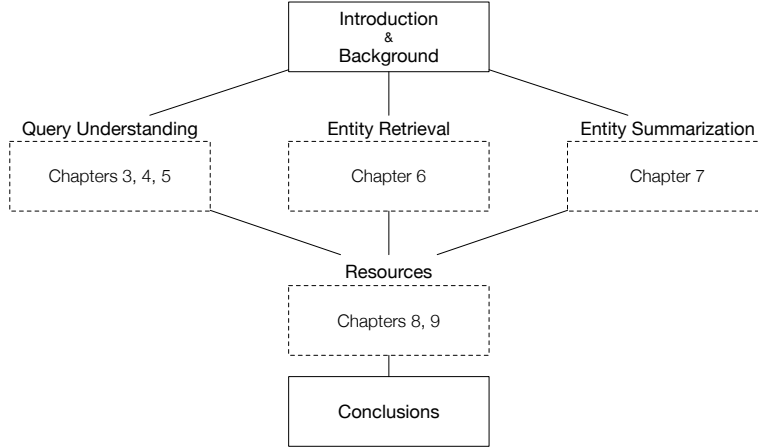


Figure 1.3: Overview of the thesis organization.

Chapter 10 revisits the research questions addressed in this thesis and provides an outlook on future research.

Readers familiar with information retrieval and semantic search may skip Chapter 2. The query understanding (Chapters 3-5), entity retrieval (Chapter 6), and entity summarization (Chapter 7) parts can be read independently of others. The resources part (Chapters 8-9) considers Chapters 5-7 as prerequisite.

1.5 Origins

The content of this thesis builds upon a number of publications. Some of these publications are directly included [76, 91–98], while some others indirectly contributed to advancing the thesis [88–90]. Below we list these publications and their relevance to the thesis.

- P1 Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. **Entity Linking in Queries: Tasks and Evaluation**, In proceedings of ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '15), pages 171-180, 2015.

[Related to RQ1 and contribution C1; included in Chapter 3.]

- P2 Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. **On the reproducibility of the TAGME Entity Linking System**, In proceedings of 38th European Conference on Information Retrieval (ECIR '16), pages 436-449, 2016.
[Related to RQ2 and contribution C2; included in Chapter 4.]
- P3 Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. **A Greedy Algorithm for Finding Sets of Entity Linking Interpretations in Queries**, In proceedings of SIGIR 2014 workshop on Entity Recognition and Disambiguation Challenge (ERD), pages 75-78, 2014.
[Related to RQ3 and contribution C3; included in Chapter 5.]
- P4 Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. **Entity Linking in Queries: Efficiency vs. Effectiveness**, In proceedings of 39th European Conference on Information Retrieval (ECIR '17), pages 40-53, 2017.
[Related to RQ3 and contribution C3; included in Chapter 5.]
- P5 Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. **Exploiting Entity Linking in Queries for Entity Retrieval**, In proceedings of ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '16), pages 209-218, 2016.
[Related to RQ4 and contribution C4; included in Chapter 6.]
- P6 Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. **Dynamic Factual Summaries for Entity Cards**, In proceedings of 40th ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '17), pages 773-782, 2017.
[Related to RQ5 and contribution C5; included in Chapter 7.]
- P7 Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. **DBpedia-Entity v2: A Test Collection for Entity Search**, In proceedings of 40th ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '17), pages 1265-1268, 2017.
[Related to contribution C6; included in Chapter 8.]
- P8 Faegheh Hasibi, Krisztian Balog, Dario Garigliotti, and Shuo Zhang. **Nordlys: A Toolkit for Entity-Oriented and Semantic Search**, In proceedings of 40th ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '17), pages 1289-1292, 2017.
[Related to contribution C7; included in Chapter 9.]

The following publication is related to the contribution C6 and is partly included in Chapter 8:

- P9 Dario Garigliotti, Faegheh Hasibi, and Krisztian Balog. **Target Type Identification for Entity-Bearing Queries**, In proceedings of 40th ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '17), pages 845-848, 2017.

These papers are not directly related to this thesis, but brought insights in the broader research area of semi-structured data and knowledge bases.

- P10 Faegheh Hasibi. **Indexing and Querying of Overlapping Structures**, In proceedings of 36th ACM SIGIR conference on Research and development in Information Retrieval (SIGIR '13), pages 1144, 2013.
- P11 Faegheh Hasibi and Svein Erik Bratsberg. **Non-hierarchical Structures: How to Model and Index Overlaps?**, In Balisage-The Markup Conference 2014.
- P12 Faegheh Hasibi, Dario Garigliotti, Shuo Zhang, Krisztian Balog. **Supervised Ranking of Triples for Type-Like Relations (The Cress Triple Scorer at WSDM Cup 2017)**, In WSDM Cup 2017.

Chapter 2

Background

In this chapter we provide an overview of conventional and modern information retrieval and set the stage for the rest of the research chapters in this thesis. We start by describing the fundamental steps and techniques used for conventional information retrieval in Section 2.1. We then move on to modern information retrieval and review knowledge bases and semantic search techniques in Section 2.2.

2.1 Conventional Information Retrieval

Information Retrieval (IR) is a vast research area that has been around since 1950, when the term was coined by Calvin Mooers [144]. A number of definitions has been proposed for information retrieval; perhaps the most commonly used one, that is still accurate today, was presented by Gerard Salton in 1968 [173]: “Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information.” This definition, in layman words, regards any computer-based search application as information retrieval; examples include desktop search, multimedia search, question answering systems, or web search engines, which are the most widely used application of information retrieval.

For a large part of the past 50 years, the primary focus of information retrieval has been on *documents*; the common scenario being that a user submits a query through a search box and a search system returns an ordered list of documents that contain the answer to the sought information needs. Identifying

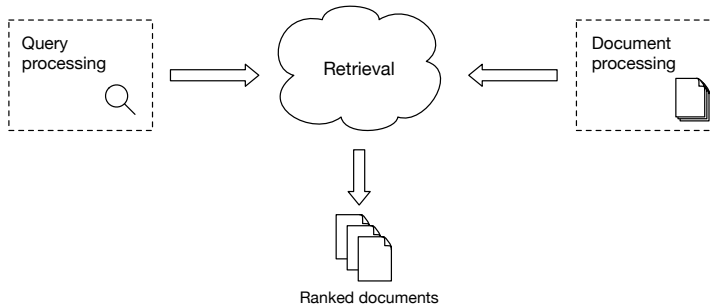


Figure 2.1: High-level building blocks of a conventional IR system

these documents, among all available ones, requires a mechanism that can match the queries against documents and assign each document a relevance score. The conventional and successful matching mechanism is based on *lexical matching* of query words against documents; i.e., modeling queries and documents based on their terms. The term “bag of words” is used when a text is represented by an unordered set of words, without considering term relations or meaning. Some other advanced models take phrases and term proximities into consideration. In this chapter, we use the term “conventional information retrieval” to refer to IR systems that consider term-based representations to model query-document relevance.¹ These systems are composed of some basic building blocks such as document processing (Section 2.1.1), query processing (Section 2.1.2), and retrieval modeling (Section 2.1.3); see Figure 2.1 for illustration. Another core aspect of IR is *evaluation*, which we describe in Section 2.1.4.

2.1.1 Document Processing

Document processing is a major component of an IR system. It is responsible for collecting documents and converting them to a structure that enables efficient search and lookup. There are three steps involved in document processing: (i) text acquisition, (ii) text transformation, and (iii) index creation [54]. Below we describe each of them.

¹ *Relevance* is a fundamental, yet loose concept in information retrieval. According to Croft et al. [54] “a *relevant* document contains the information that a person was looking for when she submitted a query to a search engine.”

Text acquisition. In many cases, text acquisition is only about using some existing collection. In most real world scenarios, however, this involves discovering new documents and updating existing ones; in web search, this process is known as *web crawling*. Basically speaking, a web crawler starts from a set of seed pages and recursively visits new pages following the links of visited pages. While seemingly simple, this is a significantly challenging process and there are certain desiderata that must be fulfilled by a crawler, such as robustness (being resilient to link traps), freshness (operating in continuous mode), politeness (respecting visiting rate policies), and efficiency (making efficient use of system resources) [133].

Text transformation. Once the documents are acquired, they need to be transformed to basic indexing units (a.k.a. index terms). This process consists of several linguistic transformations of the text, which are as follows:

Tokenization converts the input text into a sequence of tokens. A tokenizer is a language-specific component that deals with word separators and special characters. In English, for example, it handles capital characters, hyphens, apostrophes, and periods. Tokenization may also benefit from some syntactic information, obtained from part-of-speech tagging.

Stopping eliminates common words that contribute very little in matching documents against queries; examples include *the*, *a*, *for*, and *to*.

Stemming aims to derive the stem of the words; e.g., “laugh” for “laughter,” “laughing,” and “laughed.” This process may bring little improvements to retrieval effectiveness. If it is done aggressively, it can even hurt retrieval performance. Therefore, stemming may be ignored for some search applications.

Index creation. In the final stage of text processing, terms and other information about the documents will be stored in an *index*, a time and space efficient structure that enables storing and updating documents, as well as looking up information about terms and documents. The statistics that are stored in the index are used in the retrieval component.

2.1.2 Query Processing

Queries are the users’ means of interacting with a search engine and expressing their information needs. The simplest query processing step is to perform

tokenization, stopping, and stemming. These operations, however, should be performed in similar vein as for documents, so that the generated terms can be compared against the ones from the documents.

Spell checking is another query processing step, and has major effect on retrieval performance. Around 10–15% of web search queries contain spelling errors [57]; some can be easily captured by a standard spelling dictionary (e.g., “mashroom picking”), but others are related to the specific content of the Web, such as people’s names, websites, or products (e.g., “stephen roberson”) [54]. Depending on the available data sources and application, spell corrections can be performed using query logs, document collections, and trusted dictionaries.

Queries, in many cases, are poor representations of users’ information needs. A concept may be referred to by different terms (e.g., “car” and “automobile”), or a single term may match different concepts (e.g., “Jaguar” can refer to a car or an animal). Several methods have been proposed to overcome such issues, a large body of them fall under the area of *query expansion* techniques.

There are two main categories of *query expansion* techniques: global and local methods. Using global query expansion techniques, each term is expanded with related words from a thesaurus. The searchers can manually select the terms and phrases from a controlled vocabulary (e.g., in the medical domain), or the search engine itself expands the query terms and offers them to the searchers to receive their feedback. Since general thesauri contain words about different aspects of a term, automatic global query expansion techniques are shown to be less helpful [44, 189]. Local query expansion techniques, on the other hand, focus on the content of the query and choose the words that are related to the topic of the query. The well-known *pseudo-relevance feedback* method expands the query with the words that occur in the top ranked documents, assuming that these documents are relevant to the query [118, 130, 207]. Other techniques use the explicit feedback of the searchers to find relevant documents and extract the expansion terms.

2.1.3 Document Retrieval

At the core of any IR system, there is a ranking algorithm that ranks documents with respect to a query based on a *retrieval model*. Many retrieval models have been proposed over the past 50 years. In this section, we focus on the “conventional” retrieval models in which the term-based representations of documents and queries are used for ranking. Specifically, we discuss three classical retrieval models: the Vector Space Model, BM25, and Language Models. Some of these

models can be extended to model other features of documents and queries, which are further described in Section 2.2.

Vector Space Model

The Vector Space Model (VSM) [174] is an intuitive model that has been proposed in the early years of IR research. It is based on Luhn's similarity criterion [129], which suggests a statistical approach for searching information. VSM represents queries and documents as n -dimensional vectors in a common vector space. Formally, a document d and a query q are represented as:

$$\begin{aligned}\vec{d} &= (d_1, d_2, \dots, d_n), \\ \vec{q} &= (q_1, q_2, \dots, q_n),\end{aligned}$$

where n is the number terms in the whole collection, d_i and q_i are the weights of i th term for the document and the query, respectively. Various term weighting schemes have been proposed, the most common one being *tf-idf*. Given a term t and document d , the *tf-idf* weight is computed as:

$$tf\text{-}idf_{t,d} = tf_{t,d} \cdot idf_t. \quad (2.1)$$

Here, the $tf_{t,d}$ component represents the frequency of term t in document d , and is usually computed as:

$$tf_{t,d} = \frac{freq(t, d)}{\sum_{i=1}^n freq(t_i, d)}, \quad (2.2)$$

where the denominator is a document length normalization factor. The *idf* component in Eq. 2.1 stands for inverse document frequency and represents the discriminating power of a term in the whole collection. The typical definition of idf_t is:

$$idf_t = \log \frac{N}{df_t}, \quad (2.3)$$

where df_t is number of documents that contain term t (also referred to as document frequency). For a rare term, the *idf* value will be high, reflecting that fact that lots of information are carried by the term.

Once the vector representations of documents and queries are built, the similarity of each document to the query can be computed, for example, using

the cosine similarity:

$$\cos(\vec{d}, \vec{q}) = \frac{\vec{d} \cdot \vec{q}}{\|\vec{d}\| \times \|\vec{q}\|} = \frac{\sum_{i=1}^n d_i \cdot q_i}{\sqrt{\sum_{i=1}^n d_i^2 \cdot \sum_{i=1}^n q_i^2}}. \quad (2.4)$$

The vector space model makes no explicit definition of *relevance*, but implicitly considers it as the similarity between document and query vectors. One of the famous extensions of the vector space model is the *Rocchio algorithm* [171], which improves retrieval effectiveness based on the concept of relevance feedback (cf. Section 2.1.2).

BM25

Against the Vector Space Model that uses the similarity between query and index representations, Robertson [167] made a theoretical statement about relevance based on probability theory, known as *Probability Ranking Principle* (PRP). The PRP criterion encourages ranking documents according to probability of relevance of a document to a query: $P(R = 1|q, d)$. One of the early probabilistic retrieval models that aimed to estimate this probability was the Binary Independence Model (BIM). Despite the strong theoretical basis of BIM, the model performs poorly in empirical settings. It, however, became the foundation of *BM25*, which is an effective and still widely used retrieval model.

Built upon the binary independence model and the notions of term frequency and inverse document frequency,² the BM25 ranking formula for a query q and document d is defined as:

$$\text{score}(q, d) = \sum_{t \in q} \frac{tf_{t,d}}{k_1(1 - b + b \frac{|d|}{avdl}) + tf_{t,d}} idf_t, \quad b \in [0, 1], k_1 \in [0, +\infty) \quad (2.5)$$

where $|d|$ is the document length and $avdl$ is the average document length in the collection. Based on BIM, the idf component is computed as:

$$idf_t = \log \frac{N - df_t + 0.5}{df_t + 0.5}. \quad (2.6)$$

The BM25 model (Eq. 2.5) involves two free parameters, k_1 and b , which control term saturation and document length normalization components, respectively.

²The reader is referred to an IR book [8, 55, 169] for the full theory behind the BM25 model.

Based on a large number of experiments, the value ranges $0.5 < b < 0.8$ and $1.2 < k_1 < 2$ are shown to yield reasonably good performance across various IR tasks [168].

Language Models

Language Models (LM) have a long-standing history in the natural language processing community and have been successfully used for different applications, such as speech recognition and machine translation. In information retrieval, language models provide a sound and straightforward theoretical framework for developing various retrieval models [206], which are shown to provide strong empirical performance across a wide range of retrieval tasks [117]; examples include query likelihood [156], length normalized query likelihood [113], and the Kullback-Leibler (KL) divergence model [115].

One of the popular instances of language modeling is the *query likelihood* model. The basic idea is to estimate the probability of relevance of document d to the query q , i.e., $P(d|q)$. But since documents are of much longer and provide richer representation of vocabulary words than queries, we apply Bayes' rule and rewrite the probability as:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \stackrel{rank}{=} P(q|d)P(d). \quad (2.7)$$

In this transformation, the probability of the query $P(q)$ is the normalization factor, which can be ignored in most cases. The prior probability $P(d)$ can be also ignored, as it is often treated as uniform over all documents. Therefore, the final ranking model boils down to computing the query likelihood $P(q|d)$, which is computed using the unigram language models. Assuming that θ_d is the language model inferred for the document d , the probability of generating query q from the document model θ_d is:

$$P(q|\theta_d) = \prod_{t \in q} P(t|\theta_d)^{tf_{t,q}}, \quad (2.8)$$

where $tf_{t,q}$ denotes the number of times term t appears in query q . The final bit of this model is the probability $P(t|\theta_d)$, which can be simply computed as $tf_{t,d}/|d|$. However, this may result in zero value for probability $P(t|\theta_d)$, thereby turning the whole $P(q|\theta_d)$ to zero. This is not a desired behavior for document ranking, as it makes partial matches impossible. To overcome this issue, the frequency of terms in the whole collection is also taken into account, a technique

that is called smoothing. According to the *Jelinek-Mercer* smoothing method, the probability $P(t|\theta_d)$ is computed as:

$$P(t|\theta_e) = (1 - \lambda)P(t|e) + \lambda P(t|C), \quad (2.9)$$

where $P(t|d)$ and $P(t|C)$ are the probability of term t in the document d and the whole document collection C , respectively. The free parameter λ is in range $[0, 1]$ and is set empirically (often $\lambda = 0.1$).

Another smoothing technique is *Dirichlet prior*, which defines the probability $P(t|\theta_d)$ as:

$$P(t|\theta_d) = \frac{tf_{t,d} + \mu P(t|C)}{|d| + \mu}. \quad (2.10)$$

If we rewrite this equation similar to the Jelinek-Mercer smoothing method, the *lambda* coefficients of Eq. 2.9 are defined as the following in Dirichlet smoothing method:

$$\lambda = \frac{\mu}{|d| + \mu} \quad (1 - \lambda) = \frac{|d|}{|d| + \mu}. \quad (2.11)$$

The μ parameter controls the amount of smoothing, and should be set for each ranking task; the default values 1500 and 2000 are shown to deliver good performance in various experiments.

Semi-structured Retrieval

There are huge amounts of structured and semi-structured data that need to be handled by information retrieval systems; examples include HTML and XML files, patents, blogs, and emails. The term *semi-structured retrieval* refers to the models and techniques that are used for searching over semi-structured data.

Prominent retrieval models, like BM25 and language models, are extended to support multi-field documents. The fielded extension of the BM25 model, BM25F [170], generates a linear combination of term frequencies across all fields and then scores the document using the resulting pseudo term frequencies. The Mixture of Language Models (MLM) [152] is an extension of language models for semi-structured data. The model considers a separate language model for each document field and then takes a linear interpolation of these field-level models. The field weights in these models should be trained or hand-tuned. The Probabilistic Retrieval Model for Semi-structured data (PRMS) [111] is an extension of MLM, where the field weights for each query term are inferred individually, based on collection statistics. The model is shown to deliver superior performance compared to both MLM and BM25F, when there is a distinctive distribution of terms across all fields [93, 112]. In a follow-up study,

Kim and Croft [110] incorporated relevance feedback to compute field weights. The recently developed Fielded Sequential Dependence Model (FSDM) [210] is a feature-based retrieval model, similar to the Sequential Dependence Model (SDM) [139], where the document language model of feature functions are computed using the MLM model.

Learning to Rank

Learning to Rank (LTR) refers to the application of machine learning techniques in information retrieval. LTR models [125] represent the current state-of-the-art for a variety of IR tasks but their performance is largely dependent on the amount of training data available. For document retrieval, LTR models estimate the relevance of documents directly from a set of features, using a model that has been learned from the training data. The features can be of different kinds [131]: textual similarities such as BM25 or Language Model scores, link analysis features like PageRank, number of in- and out-links, and query features such as query length. There is a wide range of machine learning algorithms for learning to rank, grouped into three categories of *pointwise*, *pairwise*, and *listwise* approaches:

Pointwise approaches consider each query-document pair as a single instance, and perform ranking using regression, classification, or ordinal regression loss functions [125]; i.e., predicting a score for each single document, independent of the others. Random Forests (RF) [36] and Gradient Boosted Regression Trees (GBRT) [74] are popular algorithms from this category.

Pairwise approaches take pairs of documents and output the relative order between them (thereby casting the ranking task as a binary classification problem). Examples include GBRank [209], RankNet [39], RankBoost [73], and RankSVM [104].

Listwise approaches take all documents for a query and return a ranked list of documents, obtained by direct optimization of an IR evaluation measure, or optimizing a loss function that is defined for all documents related to a query. Coordinate Ascent (CA) [140], AdaRank [202], ListNet [41], and LambdaMART [196] are well-known examples of listwise models.

In this thesis, we take advantage of RF and GBRT algorithms for ranking tasks. Below, we briefly describe each of them.

Random Forests is a specific type of bagging technique that builds a number of uncorrelated trees using the Classification And Regression Tree (CART) procedure. In each iteration, the model builds a CART from a bootstrap sample of instances and m random features. The final prediction is the mean prediction of the individual trees. The number of iterations and maximum number of features are two free parameters that should be set empirically. Random forest is a very popular algorithm, as it is resilient to overfitting, can be easily parallelized, and most importantly delivers good performance.

Gradient Boosted Regression Trees is a boosting algorithm that sequentially builds a set of low-depth trees and generates the final prediction score by tree averaging. Unlike random forests, the creation of trees here is not random, but each tree in every iteration is built to minimize the residual error of the previous iterations. The model uses stochastic gradient descent to minimize a loss function, e.g., the squared loss. The number of iterations and the depth of the trees are free parameters of this model.

2.1.4 Evaluation

Evaluation is a fundamental aspect of information retrieval. Evaluating the quality of IR systems is most commonly done through experiments and by comparing the results against a standard test collection. These collections contain a number of queries and their corresponding relevance labels, often generated by human annotators. The Text Retrieval Conference (TREC) is the most well-known initiative that provides test collections for different IR tasks. Each TREC workshop consists of a set of tracks; sample tracks of TREC over the recent years include: Web track, entity track, question answering, and open search track. Other initiatives are INEX, CLEF, and NTCIR, which released different test collections for tasks such as XML retrieval, cross language retrieval, personal life log retrieval, mathematical information retrieval, and medical natural language processing.

Beside test collections, there are also a number of evaluation measures that quantify the performance of retrieval systems. Below we review two main categories of evaluation measures, which have been used in the course of this thesis.

Set-based Measures

The most basic measure for evaluating different IR tasks are *precision* and *recall*. Precision is the fraction of retrieved items that are relevant, and recall is the

fraction of relevant items that are retrieved. Specifically, considering \hat{A} as the set of relevant results according to the ground truth and A as the set of retrieved results, precision and recall are computed as:

$$\begin{aligned} P &= \frac{|\hat{A} \cap A|}{|A|} \\ R &= \frac{|\hat{A} \cap A|}{|\hat{A}|}. \end{aligned} \quad (2.12)$$

F-measure combines precision and recall by taking the harmonic mean of the two measures:

$$F1 = \frac{1}{\frac{1}{2}(\frac{1}{P} + \frac{1}{R})} = \frac{2PR}{P + R}. \quad (2.13)$$

These measures do not take the order of the results into account and are rarely considered for ranking problems, but they are widely used for IR-related classification tasks, such as named entity recognition and entity linking.

Rank-based Measures

In a ranked retrieval context, the quality of the results is measured at different positions of the ranked list. A natural extension of precision and recall to the ranking scenarios is to compute them at a given rank position K , denoted by $P@K$ and $R@K$. *Average Precision (AP)* is a single measure that combines precision at different levels of recall.

$$AP = \frac{1}{|\hat{A}|} \sum_{i=1}^{|\hat{A}|} P(A_i), \quad (2.14)$$

where $|\hat{A}|$ is the number of relevant results for the query, and A_i is the set of retrieval results from the top result until the i th relevant document is observed. Considering a set of queries Q , *Mean Average Precision (MAP)* is computed by taking average of AP over all queries. For many applications, especially when the relevance judgments are binary, MAP gives a complete overview of the performance of the system.

For non-binary (or graded) relevance judgments, *normalized Discounted Cumulative Gain (nDCG)* is used [103]. The central concept in computing nDCG is “gain,” which represents how much information is *gained* when a user views

a document. If the user stops viewing documents at rank p , the cumulative information gain can be computed as:

$$CG_p = \sum_{i=1}^p r_i, \quad (2.15)$$

where r_i is the gain (or the relevance level) corresponding to the i th retrieved document. This cumulative gain, however, is deficient in capturing the rank or position of each retrieved document. To capture this position-based penalty, we compute the discounted cumulative gain:

$$DCG_p = r_1 + \sum_{i=2}^p \frac{r_i}{\log_2 i}, \quad (2.16)$$

where $\log_2 i$ is the discounting factor that reduces the document's gain as the rank increases. Here, the gain of the top ranked document does not need to be discounted, as it is assumed that the user always sees this document. To normalize the final value into the range $[0, 1]$, the DCG score at each rank is divided by the DCG score of the ideal ranking, referred to as *IDCG*:

$$nDCG_p = \frac{DCG_p}{IDCG_p}. \quad (2.17)$$

For many applications, nDCG is reported for rank cut-offs 5, 10 or 20; i.e., nDCG@5, nDCG@10 or nDCG@20.

2.2 Semantic Search

Semantic search in a nutshell means “search with meaning” [20]. This stands in contrast with conventional search, where lexical matching of query and document terms is the main technique for answering search queries. Semantic search is a broad field, with many different aspects, ranging from query understanding, to answer retrieval, and result presentation, which can be applied on both text and knowledge bases [20]. In this section, we focus on the techniques that use knowledge bases, and briefly explain this core data component of semantic search. We then draw analogy to the main components of conventional information retrieval systems, and review the techniques related to document processing, query processing, and retrieval. We further shed some light on the result presentation aspect of semantic search, and explore some of the currently trending topics in this area.

2.2.1 Knowledge Bases

While documents and unstructured text are understandable only by humans, knowledge bases are meant to be machine understandable form of knowledge. *Knowledge bases* (KBs) are repositories of records about entities, their attributes, and relations to other entities.³ They are the main data components for many semantic-aware applications, such as semantic search [20], question answering [22, 203], academic, and product search [61, 200]. Over the past decade, several general purpose knowledge bases have been built, such as DBpedia [6, 120], YAGO [179, 180], NELL [42], OpenIE [17], Freebase [30], and Wikidata [68, 191]. The importance of knowledge bases is also seen by commercial search engines; Google’s Knowledge Vault [63], Bing’s Satori,⁴ Facebook’s,⁵ and LinkedIn’s⁶ knowledge graph are only few examples of industrial efforts to constructing general or domain specific knowledge bases.

The increased interest in knowledge bases may to a large extent be attributed to the emergence of the *Semantic Web* in the early 2000s. The Semantic Web, coined by Berners-Lee et al. [24], is the “global Web of machine-readable data” [25], and brings structure to the content of the Web. The term “Linked Data” refers to a means that enables reaching the goal of semantic web, and is in fact the technological framework for building knowledge bases [25]. A large number of knowledge bases have been created based on the Linked Data principles and are interlinked in the Linking Open Data (LOD) cloud.⁷ LOD is a community project⁸ that provides a platform for publishing and linking open knowledge bases. Figure 2.2 presents the LOD cloud, which currently contains 1,139 datasets.

Knowledge bases are often stored as RDF triples. Each RDF triple is in the form of `<subject, predicate, object>`, where the subject is a URI (link to another entity), and the object is a URI or a literal value. The predicate specifies the relation between subject and object and is represented by a URI. Consider for example the following RDF triples about ALBERT EINSTEIN from DBpedia:

³The term “knowledge graph” is exchangeably used to refer to the same concept, with the explicit focus that the records are represented as a graph.

⁴<https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/>

⁵<http://www.insidefacebook.com/2013/01/14/facebook-builds-knowledge-graph-with-info-modules-on-community-pages/>

⁶<https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph>

⁷<http://lod-cloud.net/>

⁸<http://linkeddata.org/>

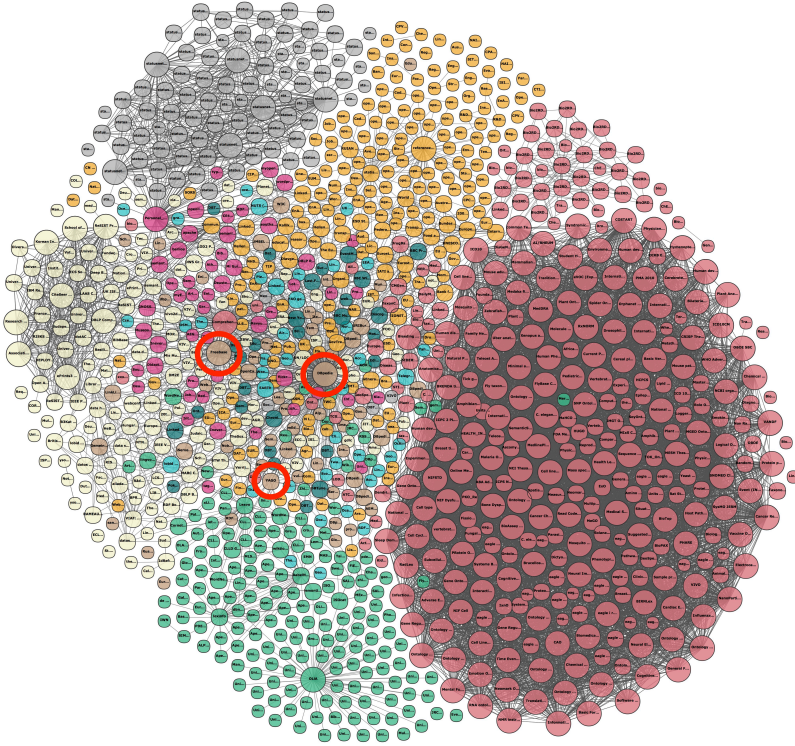


Figure 2.2: The Linking Open Data (LOD) cloud diagram [3] as of June 2017. It presents all the knowledge bases that have been published in Linked Data format, color-coded by their domain. Freebase, DBpedia, and YAGO are marked with red circles.

```
<dbpedia:Albert_Einstein    dbo:birthDate    '1879-03-14'>
<dbpedia:Albert_Einstein    dbo:field          dbpedia:Physics>,
```

where the predicates “Birth data” and “Field” take literal and URI values, respectively.

Knowledge bases can be constructed automatically (e.g., NELL and OpenIE) or by the help of humans (e.g., DBpedia, Freebase, Wikidata, and YAGO). For the first group facts are automatically extracted from text corpora using information extraction techniques. These knowledge bases, albeit being of signifi-

cant accuracy, are still below the quality of man-made knowledge bases [180]. Manually curated knowledge bases are of two types: some are launched as a community project and are directly built by the help of volunteer annotators (e.g., Wikidata and Freebase); others, like YAGO and DBpedia, are built by automatically extracting facts from (semi-structured data in) Wikipedia. These human curated knowledge bases are very popular among researchers, especially DBpedia, which has received increasing attention over the recent years; e.g., [12, 135, 138, 145]. It has been around for several years (unlike Wikidata, which is in its infancy), it is backed by an active community and is regularly updated (unlike Freebase and YAGO), and most importantly covers a wide range of entities, coupled with quality data.

2.2.2 Document Processing

Document processing in semantic search is mainly about extracting meaning from documents. One of the early tasks in this area is *named entity recognition* (NER), which aims to identify named entities in the text and label each of them with the most likely class; these classes often being person (PER), location (LOC), or organization (ORG) [32, 71, 158]. Another task, *named entity disambiguation* (NED), assumes that named entities have already been recognized and seeks to map entity mentions to their corresponding entries in a knowledge base [38, 86]. The *entity linking* (EL) task combines these two tasks and focuses on recognizing entity mentions in the text and linking (disambiguating) them to entities in a knowledge base. Entity linking is an important research problem across different communities (such as natural language processing, information retrieval, and data mining), and has been addressed at different evaluation campaigns, including the Knowledge Base Population track (KBP) of the Text Analysis Conference (TAC) [65, 66], and the Entity Recognition and Disambiguation Challenge [43].

Early work on entity linking relied on the contextual similarity between the document and the candidate referent entities [56, 141]. The Wikify! system [141] performs concept detection by extracting all n-grams that match Wikipedia concepts and then filters them. Their most effective filtering approach utilizes link probabilities obtained from Wikipedia articles. For the entity disambiguation step, they use a combination of knowledge-based and feature-based learning approaches. Cucerzan [56] employs contextual and category information extracted from Wikipedia and calculates the similarity between the document and candidate entities' pages. Later, Milne and Witten [143] introduced the concepts of *commonness* and *relatedness*, which are generally regarded as two of the most

important features for entity linking. Commonness is the probability that mention m points to the entity e , and relatedness captures the semantic similarity between two entities using their incoming and outgoing links in the knowledge base. They employ a machine learning approach, using commonness and relatedness as the main features. In their work, they demonstrate substantial improvements over prior approaches. DBpedia Spotlight [138] is another entity linking system, which uses the Vector Space Model to disambiguate named entities.

In contrast to early systems that disambiguate one entity mention at-a-time, collective entity linking systems exploit the relatedness between entities jointly and disambiguate all entity mentions in the text simultaneously [87, 99, 114, 176]. The system by Kulkarni et al. [114] exploits the interdependence between entity linking decisions in a collective manner. This system influenced TAGME [69, 70], one of the most popular entity linking systems, which has been widely used by researchers as a black-box entity linker. It finds collective agreement for the link targets using a voting scheme for a relatedness score. TAGME was designed specifically for short text snippets such as tweets and queries, but is also used for long texts such as documents. For long texts, TAGME is significantly more effective than the system of Milne and Witten [143] and more efficient than the one from Kulkarni et al. [114], while being on a par with Kulkarni et al.’s system in terms of effectiveness. For short texts, TAGME is compared to the Milne and Witten’s method, and has demonstrated significant improvements. The recently developed system by Pappu et al. [153] employs Conditional Random Fields (CRF), entity embeddings, the forward-backward algorithm [7], and other techniques from [29] to perform fast multilingual entity linking in documents.

Since entity linking is a complex process, several attempts have been made to break it down into standard components and to compare systems in a single framework [45, 46, 83, 184]. Particularly, Hachey et al. [83] reimplemented three prominent entity linking systems [38, 56, 188] in a single framework and found that much of the performance variation between these systems stems from the candidate entity ranking step (called *searcher* in their framework). Ceccarelli et al. [45] developed Dexter, an entity linking framework that provides implementations of three notable systems: Wikiminer [143], TAGME [69], and the system developed by Han et al. [87]. GERBIL [184], an extension of BAT-framework [51], is the other benchmarking framework that has implemented TAGME in addition to other eleven entity annotation systems. In this thesis, we extensively study the TAGME entity linking system, and use it for annotating queries with entities.

2.2.3 Query Understanding

Query understanding refers to the process of “identifying the underlying intent of the queries, based on a particular representation” [55]. One main branch of approaches focuses on determining the “aboutness” of queries by performing a topical classification of the query contents [40, 101, 122, 177]. Most of these methods train a classifier to predict the category of web queries.

Query segmentation is another approach for understanding queries, where the query is divided into phrases, such that each phrase can be considered as an individual concept [23, 84, 102, 106, 165, 181]. An early approach in this area is [165], which considers a segment’s frequency and mutual information between the subsequences of a segment. Mutual information is also used in a number of subsequent studies; e.g., [102, 106]. Bergsma and Wang [23] employ supervised learning for query segmentation, using decision-boundary, context, and dependency features. Later, Hagen et al. [84] proposed a naive, yet fast method for query segmentation, based on n-gram frequencies and Wikipedia titles, and showed that it performs comparable to the state-of-the-art.

Another effort for understanding query intents is *named entity recognition in queries* (NERQ); i.e., identifying named entities in a query and classifying them into predefined classes such as “movie” or “music.” The first study in this area was performed by Guo et al. [81], which employed probabilistic methods together with a weakly supervised learning algorithm (WD-LDA). Alasiry et al. [4] proposed a processing pipeline for entity detection in queries, which involves the following steps: query pre-processing (e.g., spell checking), grammar annotation (POS and ORTH tagging), segmentation, and entity recognition (based on a small set of manually constructed rules). Importantly, these works are limited to detecting mentions of entities and do not perform disambiguation or linking; that follows in the entity linking task for queries.

Entity linking for queries has only recently received attention from the research community. Related work in this area includes the TAGME system [69], which can be used for annotating short texts, such as queries and tweets. A seminal work for semantic linking in short texts was presented by Meij et al. [136]. The main strategy behind their approach is to first obtain high recall and then improve precision by employing machine learning. Their method first extracts all candidate Wikipedia concepts for each n-gram. Then a supervised learning algorithm with an excessive set of features is used to classify relevant concepts. Guo et al. [82] also studied microblog texts and employed a structural SVM algorithm in a single end-to-end task for mention detection and entity disambiguation. The work proposed by Blanco et al. [29] addresses semantic linking

for queries, considering both efficiency and effectiveness, using a probabilistic approach.

Unlike these works, which revolve around ranking entities for query spans, the Entity Recognition and Disambiguation (ERD) Challenge [43] viewed *entity linking in queries* (ELQ) as the problem of finding multiple query interpretations. ELQ advances the conventional entity linking task (as it is known for long texts) and finds set(s) of (semantically related) linked entities, where each set reflects a possible meaning (interpretation) of the query. Even though it was one of the main considerations behind the ERD Challenge to capture multiple query interpretations, only a handful of systems actually attempted to address that [64, 77, 91, 150]. Out of these, [64] performed best and was the third best performing system in overall. The winning system of the ERD challenge, SMAPH [53], “piggybacks” on a web search engine to rank entities, and then disambiguates them using a supervised collective approach. The other contribution of this work is the GERDAQ dataset, which consists of 1000 queries.

In this thesis, we discuss why entity linking in queries should be addressed as an interpretation finding task and how the other tasks studied in the literature are different from it. We further develop a data set for the ELQ task and investigate the efficiency and effectiveness of different ELQ methods.

2.2.4 Retrieval

Unlike conventional document retrieval, where users’ information needs are addressed by a ranked list of documents, semantic search techniques offer more focused responses by returning entities, documents, text snippets, or direct answers. Below we present two aspects of retrieval that are related to the research in this thesis: entity-enhanced document retrieval and entity retrieval.

Entity-enhanced Document Retrieval

Exploiting entity annotations of documents for improving document retrieval has been trending recently [34, 58, 67, 121, 126, 197, 198]. This research direction has been especially boosted by the introduction of the Freebase Annotations of ClueWeb Corpora (FACC) [75] in 2013. The first study that employed FACC annotations for ad hoc document retrieval was performed by Dalton et al. [58]. They leverage entity annotations of both documents and queries in a query expansion technique for improving document retrieval. Their approach takes annotated queries and documents as input and enriches the query with various

features extracted from entities in a knowledge base. EsdRank [197], on the other hand, annotates the query using TAGME and incorporates various features in a machine learning framework to tackle document retrieval. Recently, Raviv et al. [163] and Ensan and Bagheri [67] extended language models to represent entity annotations of document and queries. Other studies in this area either employ query expansion techniques [34, 121, 198] or operate in a latent entity space [126].

Schuhmacher et al. [175] addressed a variant of the ad hoc entity ranking task for informational queries, which is “close in spirit to ad hoc document search” [175]. They leveraged entity annotations of queries in a learning to rank framework by incorporating a set of mention, query-mention, query-entity, and entity features. They also generated two datasets, consisting of 25 and 22 queries for Robust and ClueWeb12 collections, respectively. In a follow-up study, Foley et al. [72] presented an approach that employs minimal linguistic resources and is able to deliver competitive results on the datasets in [175].

Entity Retrieval

Many information needs revolve around entities. This has been observed in different application domains, including question answering [127], enterprise search [14], and web search [157]. One main theme in entity retrieval research concerns the representation of entities; once a term-based representation is created, entities can be ranked using traditional retrieval models, much like documents [9]. Early work, especially in the context of expert search, obtains such representations by considering mentions of the given entity across the document collection [13, 15]. The INEX 2007-2009 Entity Retrieval track (INEX-XER) [59, 60] studies entity retrieval in Wikipedia, while the INEX 2012 Linked Data track goes one step further and considers Wikipedia articles together with RDF properties from the DBpedia and YAGO2 knowledge bases [193]. Much of the recent work represents entities as fielded documents, extracted from a knowledge base [12, 27, 210] or from multiple information sources [78].

Entity retrieval models can be categorized into two main groups: semistructured retrieval models [12, 148, 210] and learning to rank approaches [47, 78]. In the first category of models, the fielded representations of entities are ranked using fielded variations of standard document retrieval models, e.g., BM25F [170] or the mixture of language models [152]. This is the predominant approach for ad hoc entity retrieval [12, 27, 148, 210]. Neumayer et al. [147] apply MLM for the ad hoc entity retrieval task over RDF data, where predicates are folded into

four document fields: name, attributes, in-relations, and out-relations. They also consider a hierarchical variant of the model that organizes predicates into a two-level structure with the four main predicate types on the top level and individual predicates on the bottom level. In a follow-up work, Neumayer et al. [148] report slightly better results using a considerably simpler instantiation of the MLM model that considers only two fields: title and content. Zhiltsov et al. [210] introduced the Fielded Sequential Dependence Model (FSDM), which extends the Sequential Dependence Model [139] to multi-field representation of entities. Later, they introduced a feature-based extension of the FSDM model, coupled with an algorithm that can learn feature weights [149]. Learning to rank approaches for entity retrieval include the study by Chen et al. [47], which employs various term-based similarities as features and ranked them using supervised ranking algorithms. Graus et al. [78] also used LTR algorithms to learn the weights of different entity fields.

Entity retrieval has also been explored in the context of specific tasks, presented by various benchmarking campaigns. The INEX 2007-2009 Entity Retrieval track [59, 60] studied entity retrieval in Wikipedia. Two tasks of *entity ranking* and *list completion* were formulated, where both sought a ranked list of entities as the output. The tasks' input, however, was different: the entity ranking task provided queries with their target entity types, while the list completion task enriched queries with a small set of example entities. The Linked Data track at INEX 2012 also considered entities from Wikipedia, but articles were enriched with RDF properties from both DBpedia and YAGO2, and participants were explicitly encouraged to make use of these RDF facts and employ Semantic Web style reasoning techniques [193]. The aim of this INEX track was to answer keyword queries by Wikipedia articles, using retrieval techniques that combine textual and highly structured data. The TREC 2009-2011 Entity track [14, 16] defined the related entity finding task: return home pages of entities, of a specified type, that engage in a required relationship with a given source entity. In 2010, the Semantic Search Challenge introduced a platform for evaluating ad hoc queries, targeting a particular entity, over a diverse collection of Linked Data [85]. The 2011 edition of the challenge presented a second task, list search, with more complex queries [26]. In a complementary effort, Balog and Neumayer [12] introduced the DBpedia-Entity test collection. They synthesized a large number of queries from these benchmarking campaigns and maps the relevant results to DBpedia. We use this collection in Chapters 6 and 7, and further update it to DBpedia version 2015-10 in Chapter 8.

2.2.5 Result presentation

Result presentation in semantic search applications often goes beyond presenting “10-blue links” and their corresponding snippets. Many search engines nowadays present extra verticals in response to entity-bearing queries, such as direct answer panels and information cards. These cards are complex information objects that are displayed on both desktop and mobile devices [33, 178]. Providing content for these cards and studying users’ interactions with them is a relatively new topic that is being investigated both in academia [33, 116, 146, 190] and in industry [28, 161, 185].

Entity cards are made up of various components. An entity card typically contains image(s), entity name, short description, entity summary, and related entities (see Figure 1.1). Although these are the most central elements of entity cards in contemporary web search engines, additional entity-specific components such as maps, quotes, or tables may also be included. One of the central elements of entity cards is entity summaries: the truncated view of entity facts from a knowledge base. Below, we review different studies on entity cards and later focus on approaches for generating entity summaries.

Entity Cards

Most of the research related to entity cards has been geared towards understanding user behavior and interaction with entity cards. Navalpakkam et al. [146] performed eye and mouse tracking on SERPs and showed that relevant entity cards can affect users’ attention and, in overall, reduce the amount of time users spend to accomplish their task. In a similar study on mobile search, Lagun et al. [116] interleaved entity cards with organic search results and found that when entity cards are relevant, users can quickly find the answer and complete their task faster. With irrelevant cards, on the other hand, users spend more time on the page looking for the answer and pay attention to the results right below the card. In recent work on card content and structure, Bota et al. [33] showed that entity cards, regardless of their topics, can increase searcher engagement with organic web search results. The focus of attention in the aforementioned studies is on the search behavior of users and not on generating the actual content of the entity cards.

Entity Summarization

Summarizing entities over RDF data has been studied in various studies [48, 79, 80, 182, 183]. It has been also addressed by the more general problem

of *attribute ranking* (i.e., ranking entity predicates) [62, 185]. Notably, most of these studies have been performed by different communities in isolation, without knowing about each other.

Cheng et al. [48] introduced entity summarization over RDF data as the task of selecting top- k predicate-object pairs for an entity. Their system, called RELIN, leverages relatedness and informativeness of entity facts using the PageRank algorithm. In a similar vein, SUMMARUM [182] and LinkSUM [183] employ the PageRank algorithm to generate ranking scores for predicates involving two entities (i.e., no literal values are considered). The FACES [79] and FACES-E [80] systems approach entity summarization as a classification task; they partition entity facts into semantically similar groups (facets), and pick the best fact from each facet to form the summaries. In another line of work, entity summarization is approached as a classification task. FACES [79] partitions entity facts into semantically similar groups (facets), ranks the facts within each group, and finally generates the entity summary by picking the top ranked facts from each of the facet groups. The extension of this system, (FACES-E) [80], computes types for object values and generates facets considering both object and datatype properties.

Entity summarization can be remotely connected to the attribute ranking approaches proposed in [62, 119, 185]. Lee et al. [119] proposed generative models for ranking attributes with respect to entity classes, i.e., how typical an attribute is to a given class. The framework presented in [62] ranks RDF attributes for the given entity using learning to rank algorithms. The recent patent by Vadrevu et al. [185] presents an attribute ranking approach for entity summarization. Their proposed approach hinges on a machine-learned ranker (classifier), with the features based on the global and type-specific importance of entity attributes.

It is important to point out that all discussed systems generate query-agnostic summaries. In Chapter 7 of this thesis, we present a query-aware entity summarization method and evaluate it in the context of entity cards.

Chapter 3

Query Understanding via Entity Linking

Query understanding has been a longstanding area of research in information retrieval [55, 172]. One way of capturing what queries are about is to annotate them with entities from a knowledge base. This general problem has been studied in many different forms and using a variety of techniques over the recent years [29, 43, 135]. Approaches have been inspired by methods that recognize and disambiguate entities appearing in full-text documents by mapping them to the corresponding entries in a knowledge base, a process known as *entity linking* [137] (or *wikification* [141]). Successful approaches to entity linking incorporate context-based features in a machine learning framework to disambiguate between entities that share the same surface form [69, 138, 141, 143]. While the same techniques can be applied directly to short, noisy texts, such as microblogs or search queries, there is experimental evidence showing that the same methods perform substantially worse on short texts (tweets) than on longer documents (news) [51, 166]. One problem is the lack of proper spelling and grammar, even of the most basic sort, like capitalization and punctuation. Therefore, approaches that incorporate richer linguistic analysis of text cannot be applied.

There is, however, an even more fundamental difference concerning entity annotations in documents vs. queries that has not received due attention in the literature. When evaluating entity linking techniques for documents, it is implicitly assumed that the text provides enough context for each entity mention



Figure 3.1: Example of an ambiguous query, where the mention “france” can be linked to entities FRANCE and FRANCE NATIONAL FOOTBALL TEAM. The interpretations of this query are: {FRANCE, FIFA WORLD CUP} and {FRANCE NATIONAL FOOTBALL TEAM, FIFA WORLD CUP}.

to be resolved unambiguously. Search queries, on the other hand, typically consist of only a few terms, providing limited context. Specifically, we focus on a setting where there is no context, such as previous queries or clicked results within a search session, available for queries. In this setting, it may be impossible to select a single most appropriate entity for a given query segment. Consider, as an illustrative example, the query “new york pizza manhattan.” It could be annotated, among others, as “[NEW YORK CITY] pizza [MANHATTAN]” or as “[NEW YORK-STYLE PIZZA][MANHATTAN],” and both would be correct (linked entities are in brackets); see Figure 3.1 for an illustration.

The main research question we seek to answer in this chapter is the following:

RQ1. How can the inherent ambiguity of entity annotations in queries be handled and evaluated?

One line of prior work has dealt with this problem by adopting a retrieval-based approach: returning a ranked list of entities that are semantically related to the query [29, 135]. We refer to it as the task of *semantic linking*. The Entity Recognition and Disambiguation (ERD) Challenge [43] represents a different perspective by addressing the issue of ambiguity head-on: search queries can legitimately have more than a single interpretation. An interpretation is a set of entities, with non-overlapping mentions, that are semantically compatible with the query text [43]. We term this task as *entity linking in queries (ELQ)*. Both approaches have their place, but there is an important distinction to be made

as they are designed to accomplish different goals. Semantic linking is a tool for aiding users with suggestions that could be beneficial for enhancing navigation or for contextualization. Entity linking in queries, on the other hand, is a means to machine-understanding of queries. Table 3.1 illustrates the differences with a number of examples.

Once these differences are established and the tasks are defined, we focus on evaluation methodology. The current practice of rank-based evaluation is appropriate for the semantic linking task. As for entity linking in queries, interpretations are considered as atomic units, i.e., an interpretation is correct only if it contains the exact same entities as the ground truth; partial matches are not rewarded [43]. This is a rather crude method of evaluation. We present a relaxed alternative that considers both the correctness of interpretations, as atomic units, as well as the set of entities recognized in the query.

As with any problem in information retrieval, the availability of public datasets is of key importance. The recently released Yahoo! Webscope Search Query Log to Entities (YSQLE) dataset [1] is suitable for semantic linking, but not for entity linking in queries. The ERD challenge platform [43] is fitting for entity linking in queries, however, only the development set (91 queries) is publicly available, which is not large enough for training purposes. We, therefore, introduce and make publicly available a new dataset based on YSQLE, called Y-ERD. It contains interpretations for 2,398 queries and is accompanied by a clear set of annotation guidelines.

To further our understanding on the two tasks of semantic linking and entity linking in queries, we propose simple, easy-to-implement methods for each of them and illustrate the differences by experimental results. We introduce a pipeline architecture for both tasks, identify shared components, and evaluate them on the appropriate datasets.

In summary, the aim of this chapter is to make a methodological distinction between two tasks within the problem area of annotating queries with entities: *semantic linking* and *entity linking in queries* (Sections 3.1 and 3.2). We further explain this distinction by presenting methods and experimental results for the two tasks (Sections 3.4 and 3.5). We also develop a dataset and provide evaluation refinements for the task of entity linking in queries (Section 3.3). The resources corresponding to this chapter are presented in Appendix A.1.

3.1 Task Definitions

In this section, we first discuss the entity linking task for documents in Section 3.1.1. Next, in Section 3.1.3, we look at the semantic linking task and show that the task itself is easier than entity linking and resembles more of a *related entity finding* problem. Finally, in Section 3.1.2, we present the task of entity linking in queries, which deals with the inherent ambiguity of search queries.

3.1.1 Entity Linking (in Documents)

Entity linking is the task of recognizing entity mentions in text and linking (disambiguating) them each to the most appropriate entry in a reference knowledge base. This task implicitly assumes that the input text (document) provides enough context so that all entity occurrences can be resolved unambiguously.

Evaluation is performed against a gold-standard data set that consists of manual annotations. These annotations comprise the specific entity mentions (offsets in the text) and the corresponding links to the knowledge base. Effectiveness is measured in terms of precision and recall, where precision is defined as the number of correctly linked mentions divided by the total number of links established by the system, and recall is defined as the number of correctly linked mentions divided by the total number of links in the gold-standard annotations [141]. For overall system evaluation the F-measure is used. Both micro- and macro-averaging can be employed [51]. Since mention segmentation is often ambiguous, and the main focus is on the disambiguation of entities, the correctness of entity mention boundaries is often relaxed [43]. On the other hand, evaluation is rather strict in that credit is only given for a given mention if the linked entity (unique entity identifier) perfectly matches the gold standard. Overlapping entity mentions in the annotations are not allowed, i.e., any given segment of the document may be linked to at most a single entity.

3.1.2 Entity Linking in Queries

Existing entity linking approaches can be used out-of-the-box to annotate queries with entities, analogously to how it is done for documents; after all, the input is text, which is the same as before, just shorter and less grammatical (the quality of the resulting annotations is another matter). The fundamental difference between documents and queries is that queries offer very limited context. A search query, therefore, “can legitimately have more than one interpretation,”

Table 3.1: Comparison of different tasks in the problem are of annotating queries with entities.

	Entity Linking [†]	Semantic Linking	Entity Linking in Queries
Result	set	ranked list	sets of sets
Entities explicitly mentioned	Yes	No	Yes
Mentions can overlap	No	Yes	No [‡]
Evaluation criteria	mentioned entities found	relevant entities found	interpretations found
Evaluation measures	set-based	rank-based	set-based
Examples			
“obama mother”	{BARACK OBAMA}	ANN DUNHAM BARACK OBAMA	{{BARACK OBAMA}}
“new york pizza manhattan”	{NEW YORK CITY, MANHATTAN} [*]	NEW YORK CITY NEW YORK-STYLE PIZZA MANHATTAN MANHATTAN PIZZA ...	{{NEW YORK CITY, MANHATTAN}, {NEW YORK-STYLE PIZZA, MANHATTAN}}
“the music man”	{THE MUSIC MAN} [*]	THE MUSIC MAN THE MUSIC MAN (1962 FILM) THE MUSIC MAN (2003 FILM) ...	{{THE MUSIC MAN} {THE MUSIC MAN (1962 FILM)}, {THE MUSIC MAN (2003 FILM)}}}

[†] This refers to traditional entity linking (for documents) applied to queries. We argue in this chapter that entity linking in this form should be avoided.

[‡] Not within the same interpretation.

^{*} A single interpretation is selected arbitrarily; there are multiple options.

where each interpretation consists of a set of “non-overlapping linked entity mentions that are semantically compatible with the query text” [43]. The inherent presence of multiple query interpretations is addressed head-on by the setup introduced at the Entity Recognition and Disambiguation (ERD) Challenge [43], where “interpretations of non-overlapping linked entity mentions” [43] are to be returned. Formally, let q be a query and \hat{I} be the set of interpretations for this query (according to the ground truth), $\hat{I} = \{\hat{E}_1, \dots, \hat{E}_n\}$, where n is the number of interpretations, \hat{E}_i is a query interpretation, $\hat{E}_i = \{(m_1, e_1), \dots, (m_k, e_k)\}$, and (m, e) is a mention-entity pair. For simplicity, the specific offsets of entity mentions are not considered, however, the corresponding entity mentions in \hat{E}_i must not overlap. It is important to point out that the query might not have any entity linking interpretations ($\hat{I} = \emptyset$).

If the traditional evaluation methodology were to be adopted (as in Section 3.1.1, with the simplification of ignoring the offsets of mentions), the ground truth would need to consist of a single set of entities; we denote this set as \hat{E} . As long as the query has a single interpretation, it is straightforward; entities in that interpretation will amount to the ground truth set. Having no valid interpretation is also painless, we set $\hat{E} = \emptyset$. For entities with multiple interpretations, there are two natural ways of setting \hat{E} .

Collapsing interpretations The first option is to collapse all interpretations into a single set: $\hat{E} = \bigcup_{i \in [1..n]} \hat{E}_i$. (This is similar in spirit to the approach that is followed in the semantic linking task, cf. Section 3.1.3.) With this solution, however, the requirements that the linked entities within an interpretation must be semantically related and their mentions must not overlap are violated. It also ignores the element of multiple interpretations altogether.

Selecting a single interpretation The second option is to pick a single interpretation $\hat{E} = \hat{E}_j$, where $j \in [1..n]$. Given that all interpretations are of equal importance, selecting j in an arbitrary way would be unfair, as it would randomly favor certain systems over others. A better alternative would be to choose j individually for each system such that it maximizes the system’s performance on a given evaluation measure, e.g., F1-score. Essentially, the system’s output would be scored based on the closest matching interpretation. While the latter variant appears to be a viable solution, it still disregards the fundamental aspects of finding multiple interpretations for queries.

In summary, the entity linking task cannot be performed the same way for queries as it is done for documents, because of the element of multiple interpretations. We, however, define entity linking in queries as the task of *finding a set of interpretations* $\hat{I} = \{\hat{E}_1, \dots, \hat{E}_n\}$, where each interpretation consists of *non-overlapping mentions*. In the remainder of this thesis, when we talk about the ELQ task, we follow this definition.

3.1.3 Semantic Linking

Semantic linking is primarily intended to support users in their search and browsing activities by returning entities that can help them to acquire contextual information or valuable navigational suggestions [135]. For semantic linking, all entities that can be linked to the query (i.e., entities from all interpretations) are relevant. Beyond those, entities that are not explicitly mentioned, but referred to, may also be considered relevant; see Table 3.1 for illustrative examples. The goal, therefore, is quite different from that of finding interpretation(s) of the query for machine understanding. The requirements on the linked entities are relaxed: (i) the mentions can be overlapping, (ii) they do not need to form semantically compatible sets, (iii) they do not even need to be explicitly mentioned, as long as they are semantically related to the query.

Formally, let \hat{E} denote the set of relevant entities for the semantic linking task. This set is formed from entities across all interpretations ($\bigcup_{i \in [1..n]} \hat{E}_i^e$), plus, optionally, additional entities (E^*) that are indirectly referenced from the query: $\hat{E} = \bigcup_{i \in [1..n]} \hat{E}_i^e \cup E^*$. Semantic linking returns a ranked list of entities $\vec{E} = \langle e_1, \dots, e_m \rangle$, which can be compared against \hat{E} using standard rank-based measures, such as mean average precision (MAP) or mean reciprocal rank (MRR). Importantly, if $\hat{E} = \emptyset$ then the given query is ignored in the evaluation, meaning, that there is no difference made between system A that does not return anything and system B that returns meaningless or nonsense suggestions. This is undesired behavior; it also stands in contrast to standard entity linking, where false positives decrease system performance.

The above relaxations make semantic linking a substantially easier and different task from what entity linking for queries entails in its entirety. Therefore, the terminological distinction becomes useful and important.

3.2 Evaluation Methodology

We now discuss evaluation methodology for the ELQ task based on the literature and make suggestions for further refinements. We write $\hat{I} = \{\hat{E}_1, \dots, \hat{E}_m\}$ to denote the query interpretation according to the ground truth, and $I = \{E_1, \dots, E_n\}$ is the interpretation returned by the system. Precision and recall, for a given query, are defined at the ERD Challenge [43] as follows:

$$P = \frac{|I \cap \hat{I}|}{|\hat{I}|}, \quad R = \frac{|I \cap \hat{I}|}{|I|}. \quad (3.1)$$

Here, a hypothesized interpretation set E_i matches the reference set \hat{E}_j if their entities match ($E_i^e = \hat{E}_j^e$) and mentions of E_i do not overlap with each other. Note that according to this definition, if the query does not have any interpretations in the ground truth ($\hat{I} = \emptyset$) then recall is undefined; similarly, if the system does not return any interpretations ($I = \emptyset$), then precision is undefined. To cover such situations, we define precision and recall for interpretation-based evaluation as:

$$P_{\text{int}} = \begin{cases} |I \cap \hat{I}|/|I|, & I \neq \emptyset \\ 1, & I = \emptyset, \hat{I} = \emptyset \\ 0, & I = \emptyset, \hat{I} \neq \emptyset. \end{cases} \quad (3.2)$$

$$R_{\text{int}} = \begin{cases} |I \cap \hat{I}|/|\hat{I}|, & \hat{I} \neq \emptyset \\ 1, & \hat{I} = \emptyset, I = \emptyset \\ 0, & \hat{I} = \emptyset, I \neq \emptyset. \end{cases} \quad (3.3)$$

This evaluation is methodologically correct, it captures the extent to which the interpretations of the query are identified. It does so, however, in a rather strict manner: partial matches are not given any credit. This strictness is also pointed out in [43]. Their alternative solution, albeit purely for analysis purposes, was to measure micro-averaged precision, recall, and F1-score on the entity level. That measure, on its own, is imperfect as “entities belonging to different interpretations were mixed together” [43]. Further, by micro-averaging, the query borders are also collapsed. We propose an alternative “lenient” evaluation for interpretation finding that rewards partial matches while respecting query boundaries.

Lenient evaluation Our proposal is to combine interpretation-based evaluations (cf. Equations 3.2 and 3.3) with the conventional entity linking evaluation,

referred to as entity-based evaluation, from now on. Formally, entity-based evaluation is defined as follows:

$$P_{\text{ent}} = \begin{cases} |E \cap \hat{E}|/|E|, & E \neq \emptyset \\ 1, & E = \emptyset, \hat{E} = \emptyset \\ 0, & E = \emptyset, \hat{E} \neq \emptyset. \end{cases} \quad (3.4)$$

$$R_{\text{ent}} = \begin{cases} |E \cap \hat{E}|/|\hat{E}|, & \hat{E} \neq \emptyset \\ 1, & \hat{E} = \emptyset, E = \emptyset \\ 0, & \hat{E} = \emptyset, E \neq \emptyset. \end{cases} \quad (3.5)$$

We write \hat{E} to denote the set of all entities from all interpretations in the ground truth, $\hat{E} = \bigcup_{i \in [1..m]} \hat{E}_i^e$, and E is a set of all entities from all interpretations returned by the entity linking system, $E = \bigcup_{j \in [1..n]} E_j^e$. Finally, we define precision and recall as a linear combination of interpretation-based and entity-based precision and recall:

$$P = \frac{P_{\text{int}} + P_{\text{ent}}}{2}, \quad R = \frac{R_{\text{int}} + R_{\text{ent}}}{2}. \quad (3.6)$$

For simplicity, we consider them with equal weight, but it could easily be controlled by adding a weight parameter. In all cases, the F-measure is computed according to:

$$F = \frac{2 \cdot P \cdot R}{P + R}. \quad (3.7)$$

For computing precision, recall, and the F-measure on the whole evaluation set, an unweighed average over all queries are taken (i.e., macro-averaging is used). This provides an intuitive, easy-to-implement, and methodologically correct solution. A reference implementation is made publicly available.

3.3 Test Collections

We present two publicly available test collections for the semantic linking and ELQ tasks, and introduce a new dataset for ELQ.

3.3.1 YSQLE

The Yahoo Search Query Log to Entities (YSQLE) dataset [1] comprises a selection of queries that are manually annotated with Wikipedia entities. Annotations are performed within the context of search sessions. Each annotation

is aligned with the specific mention (“span”) within the query. In addition, the linked entities may be labelled as *main*, to specify the intent or target of the user’s query, regardless of whether the entity is mentioned explicitly in the query. For example, the query “france 1998 final” is annotated with three entities, FRANCE NATIONAL FOOTBALL TEAM, FRANCE, and 1998 FIFA WORLD CUP FINAL, of which only the last one is considered as the main annotation. Out of 2,635 queries in the YSQL dataset, 2,583 are annotated with Wikipedia entities.

YSQLE is not suitable for the task of entity linking in queries, due to a number of issues. First and foremost, the dataset does not provide query interpretations, which is an essential part of entity linking in queries as we discussed in Section 3.1. Moreover, it is not possible to automatically form interpretation sets from the annotations. An example is the query “france world cup 1998,” linked to the entities 1998 FIFA WORLD CUP, FRANCE NATIONAL FOOTBALL TEAM, and FRANCE. This query has two valid interpretations {1998 FIFA WORLD CUP, FRANCE NATIONAL FOOTBALL TEAM} and {1998 FIFA WORLD CUP, FRANCE}. One could assume that the main annotations would serve as interpretations, but it does not hold, as there exist queries with multiple or overlapping main annotations. For example, the query “yahoo! finance,” has two main annotations, linking the mention “yahoo!” to YAHOO! and the mention “yahoo! finance” to YAHOO! FINANCE. Second, the linked entities are not necessarily mentioned explicitly in the query, but sometimes are only being referred to. For example, the query “obama’s mother” is linked to BARACK OBAMA and ANN DUNHAM, where the latter is specified as the main annotation. Another example is “charlie sheen lohan,” which is linked to ANGER MANAGEMENT (TV SERIES) and to the two actors CHARLIE SHEEN and LINDSAY LOHAN. While this, in a way, is just a matter of how the annotation guidelines are defined, it is not in accordance with the requirements for entity linking in queries; i.e., entity linking is performed on explicit mentions and reference resolution is not part of the task. Carmel et al. [43] brings the query “Kobe Bryant’s wife” as an example, which should be annotated as “[KOBE BRYANT]’s wife.” Accordingly, the “obama’s mother” query should have a single interpretation, BARACK OBAMA. Further, annotations are created by considering other queries from the session; this represents a different setting from what is discussed in Section 3.1.2. Lastly, the annotations in YSQL are not always complete, meaning that some query spans that should be linked to entities are ignored. For instance the query “louisville courier journal” is annotated with THE COURIER JOURNAL, whereas the link for the mention [louisville] (to LOUISVILLE, KENTUCKY) is missing.

In summary, even though the YSQLE dataset is intended for the purpose of entity linking in queries, in practice it is mostly suitable for the semantic linking task. Nevertheless, it offers a great starting point; we show in Section 3.3.3 that with some manual effort, YSQLE can be adjusted to suit entity linking in queries.

3.3.2 ERD

The Entity Recognition and Disambiguation (ERD) Challenge [43] introduced the first query entity linking evaluation platform that properly considers query interpretations. For each query, it contains all possible interpretations (from the pool of all participating systems). Human annotations are created in accordance with the following three rules [43]: (i) the longest mention is used for entities; (ii) only proper noun entities should be linked; (iii) overlapping mentions are not allowed within a single interpretation. A training set, consisting of 91 queries, is publicly available.¹ The ERD Challenge runs evaluation as a service; entity linking systems are evaluated upon sending a request to the evaluation server (hosted by the challenge organizers). Therefore, the test set, comprising 500 queries, is unavailable for traditional offline evaluation. In order to make a distinction between the two query sets provided by the ERD Challenge, we refer to the former one (91 queries) as ERD-dev and to the latter one (500 queries) as ERD-test.

The ERD-dev dataset includes a small number of queries, of which only half (45 queries) are linked to entities; see Table 3.2. Therefore, the dataset cannot be used for training purposes and the need for a large entity linking test collection for queries still remains. In the following, we describe our new test collection, which aims to provide just that.

3.3.3 Y-ERD

To overcome the limitations of the YSQLE and ERD datasets, we set out to develop a test collection for interpretation finding based on YSQLE. Taking YSQLE as our starting point, we manually (re)annotated all queries following a set of guidelines, which are based on the ERD Challenge. The application context is general web search. The resulting dataset, referred to as *Y-ERD*, contains 2398 queries in total; see the statistics in Table 3.2.

We further note that there is a small overlap between ERD-dev /test and Y-ERD (18 queries, to be precise). We removed those queries from Y-ERD

¹<http://web-ngram.research.microsoft.com/erd2014/Datasets.aspx>.

Table 3.2: Statistics of test collections for entity linking in queries.

Query types	Y-ERD	ERD-dev
No entity	1142	46
Single entity	1133	34
Single set; >1 entity	114	7
Multiple sets	9	4
Total	2398	91

for our experiments, so that it is possible to train systems using Y-ERD and evaluate them using ERD-dev/test.

From YSQL to Y-ERD

Taking the YSQL dataset as our input, we proceeded as follows. First, we filtered out duplicate queries. Recall that YSQL queries are annotated within the context of search sessions and there are queries that appear in multiple sessions. We annotate queries on their own, regardless of search sessions, just like it was done at the ERD Challenge. Next, we created candidate interpretations using the following rules: (i) if the mentions are not overlapping, the linked entities form a single interpretation; (ii) if the entity mentions are identical, then each entity is considered as a separate interpretation (a set with a single element); (iii) queries that have been linked to a single entity, a single-element interpretation is created. Then, we asked three human annotators to judge these candidate query interpretations (including both finding interpretations and aligning the linked entities with the specific mention) following a set of annotation guidelines.

Annotation Guidelines

These guidelines are based on those of the ERD Challenge [43], complemented by some additional rules:

- R1 The annotated entities should be proper noun entities rather than general concepts [43]. E.g., the query “SUNY albany hospital location” is only linked to UNIVERSITY AT ALBANY, SUNY and the entity LOCATION (GEOGRAPHY) is ignored.

- R2 The query should be linked to an entity via its longest mention [43]. E.g., in the query “penticton bc weather,” the longest mention for the entity PENTICTON is “penticton bc.” This implies that the term “bc” is not to be linked to BRITISH COLUMBIA.
- R3 Terms that are meant to restrict the search to a certain site (such as Facebook or IMDB) should not be linked. This is a case for navigational queries, where the site is not the focus of the query. E.g., the entity FACEBOOK is not linked in the query “facebook obama slur,” while is it a valid annotation for the query “how to reactivate facebook.”
- R4 Linked entities must be explicitly mentioned in the query. One example is the query “charlie sheen lohan” that we already discussed for YSQLE in Section 3.3.1. For us, only CHARLIE SHEEN and LINDSAY LOHAN are valid annotations. Another example is “Kurosawa’s wife,” which should be linked solely to the entity AKIRA KUROSAWA, and not to YŌKO YAGUCHI.
- R5 It could be argued either way whether misspelled mentions should be linked to entities or not. In our definition, misspellings that are recorded as name variants in DBpedia are not considered as spelling errors. We believe that annotating misspelled mentions would introduce noise into the training data. Therefore, we do not perform spell correction and do not consider misspelled mentions in our ground truth in the experiments reported in this chapter. Nevertheless, we also made a spell-corrected version of Y-ERD publicly available.

Based on the above rules, the assessors were instructed to: (i) identify mentions, (ii) drop invalid linked entities, (iii) change linked entities to different ones if they are a better match, (iv) complement existing interpretations with more entities. Note that by the last rule, we restrict annotators to not adding new entities to those originally identified in YSQLE, except for the case of erroneous interpretation sets. Recall that our application domain is general web search; we trust that the annotations in YSQLE include all entities that are “meaningful” in this context. One might argue that annotations are dominated by popular entities; while this may be the case, it is no different from how annotations for the ERD Challenge were performed.

Resolving Disagreements

Regarding the interpretations (i.e., the sets of linked entities) all three annotators agreed on 84% of the queries, two agreed on 5%, and they all disagreed on

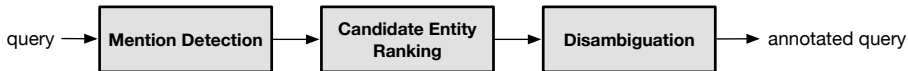


Figure 3.2: Pipeline for semantic linking (first two steps) and entity linking in queries (all steps).

the remaining 11%. For the entities linked by at least 2 assessors, the agreement on the mentions was 94%.

Disagreements were resolved through discussion, where the conflicting cases were categorized into *medium* and *hard* classes (unanimously agreed queries are regarded as *easy*). The former could be resolved through little discussion, while the latter was challenging to find agreements on. The difficulty levels are also recorded and released with the dataset.

3.4 Approaches

This section presents basic approaches for tackling the two tasks we have introduced in Section 3.1: semantic linking and entity linking in queries. Recall that semantic linking is the task of returning a ranked list of entities that are related to the query. Entity linking in queries is about finding (possibly multiple) interpretations, where an interpretation is a set of semantically compatible entities that are each mentioned in the query. We address both tasks in a pipeline architecture, shown in Figure 3.2. This pipeline is motivated by the canonical entity linking approach for documents; our components (mention detection, candidate entity ranking, and disambiguation) roughly correspond to the *extractor*, *searcher*, and *disambiguator* steps in traditional entity linking [83]. While this is a reasonable choice, it is certainly not the only one. We leave the exploration of alternative architectures to the future work. Notice that the first two components of the pipeline (discussed in Sections 3.4.1 and 3.4.2) are shared by the semantic linking and ELQ tasks. For ELQ, there is an additional disambiguation step to be performed (Section 3.4.3).

Before we continue, let us clarify the terminology. The term *span* refers to a query substring (n-gram). By *entity surface forms* (or *aliases*) we mean the names that are used to make reference to a particular entity. When we want to focus on a span that refers to (i.e., may be linked to) an entity, we use the term *mention*. A mention, therefore, is often paired with an entity, (m, e) , where m is a span matching one of the surface forms of entity e .

3.4.1 Mention Detection

The objective of the *mention detection* step is to identify query spans that can be linked to the entities. We view this as a recall-oriented task, as we do not want to miss any of the entities that are part of the query’s interpretation(s). To identify entities mentioned in the query, we perform lexical matching for all possible n-grams in the query against known entity surface forms. (Given that web queries are typically short, this is manageable.) Surface forms are gathered from two sources: from a manually curated *knowledge base* and from machine-annotated *web corpora*.

Knowledge base. We consider known surface forms from DBpedia that are recorded under the `rdfs:label` and `foaf:name` predicates. The names of redirected entities are also included. We write A_e to denote the set of aliases for entity e . Let M_{kb} be the set of entity mentions in the query:

$$M_{kb} = \{(m, e) | \exists a \in A_e : a = m, m \in q\}, \quad (3.8)$$

where $m \in q$ is a query span (can be the entire query) that matches one of the aliases (a) of entity e . Because DBpedia is a high-quality resource, we do not perform any additional filtering or cleansing step on this set.

Web corpora. We make use of web-scale document collections in which entity mentions have been automatically linked to the Freebase knowledge base. Google, Inc. has recently created and made available this resource, referred to as Freebase Annotations of the ClueWeb Corpora (FACC), for the ClueWeb09 and ClueWeb12 datasets [75]. We create a dictionary of surface forms that contains the linked Freebase IDs along with frequencies and link entities to DBpedia via `sameAs` relations. Note that we aggregate data from both ClueWeb collections (hence the usage of “corpora”).

Considering all entities that match a given surface form might leave us with a huge set of candidates; for example, “new york” matches over two thousand different entities. Therefore, we filter the set of matching candidate entities based on *commonness*. Commonness measures the overall popularity of entities as link targets [134]. Essentially, commonness is the maximum-likelihood probability that entity e is the link target of mention m :

$$\text{commonness}(e, m) = P(e|m) = \frac{\text{link}(e, m)}{\text{link}(m)}, \quad (3.9)$$

where $link(e, m)$ represents the number of times mention m is linked to entity e , and $link(m)$ is the total number of times mention m is linked to any entity. M_w refers to the set of mentions with a certain minimum commonness score:

$$M_w = \{(m, e) | m \in q, commonness(m, e) > c\}, \quad (3.10)$$

where the commonness threshold c is set empirically (or set to 0 if no pruning is to be performed). Using FACC as a source of entity surface forms and for a more reliable estimation of commonness scores is a novel approach; as we show later, it can warrant over 90% recall.

Combining sources The final set of mentions is created by combining the entities identified using the knowledge base and the web annotations: $M = M_{kb} \cup M_w$.

3.4.2 Candidate Entity Ranking

We now turn to the second component of our pipeline, which ranks the entities identified by the mention detection step. Formally, this step takes a list of mention-entity pairs (m, e) as input and associates each with a relevance score. For the semantic linking task, this ranking will constitute the final output. We note that (i) our methods are limited to returning entities explicitly mentioned in the query; this is not unreasonable (the same limitation is present, e.g., in [29]); (ii) for each entity we only consider its highest scoring mention. For ELQ, the resulting ranking provides input for the subsequent disambiguation step (cf. Figure 3.2). As the entity relevance scores will be utilized in a later component, it is essential that they are comparable across queries. (This requirement is not unique to our approach; it would also be the case if one were to use supervised learning, for example.)

Mixture of Language Models (MLM)

The predominant approach to ranking structured entity representations (typically described as a set of RDF triples) is to employ fielded extensions of standard document retrieval models, such as BM25F [27] or the Mixture of Language Models (MLM) [12, 147]. The MLM approach [152] combines language models estimated for different document fields. The model can readily be applied to ranking (document-based representations) of entities by considering different predicates as fields [111, 147]. The probability of a term t given the language

model of an entity e is estimated as follows:

$$P(t|\theta_e) = \sum_{f \in F} \mu_f P(t|\theta_{e_f}), \quad (3.11)$$

where F is the set of possible fields, f is a specific field, μ_f is the field weight (such that $\mu_f \in [0..1]$ and $\sum_{f \in F} \mu_f = 1$), and θ_{e_f} is the field language model, which is a maximum-likelihood estimate smoothed by a field-specific background model:

$$P(t|\theta_{e_f}) = (1 - \lambda_f) \frac{n(t, e_f)}{|e_f|} + \lambda_f P(t|C_f). \quad (3.12)$$

Here, $n(t, e_f)$ denotes the number of occurrences of term t in field f of entity e and $|e_f|$ is the length of the field. To keep things simple, we use a single smoothing parameter for all fields: $\lambda_f = 0.1$, based on the recommendations given in [208] for title queries.

The most common approach in language modeling is to rank items (here: entities) based on query likelihood:

$$P(e|q) = \frac{P(q|e)P(e)}{P(q)} \propto P(e)P(q|e) \quad (3.13)$$

$$\stackrel{rank}{=} P(e) \prod_{t \in q} P(t|\theta_e)^{n(t,q)}, \quad (3.14)$$

where θ_e is the entity language model (defined in Eq. 3.11) and $n(t, q)$ denotes the number of times term t is present in query q . Here $p(e)$ is the prior probability of an entity and is assumed to be uniform. When a single query is considered, dropping the query probability $P(q)$ in Eq. 3.13 can be done conveniently. For us, however, scores (probabilities) need to be comparable across different queries, as they are utilized in the subsequent disambiguation step (cf. Section 3.4.3). Therefore, the denominator, which depends on the query, should not be dropped. We perform normalization as suggested in [113] (length normalized query likelihood ratio), and use the following as our final ranking formula:

$$P(q|e) \stackrel{rank}{=} \frac{\prod_{t \in q} P(t|\theta_e)^{P(t|q)}}{\prod_{t \in q} P(t|C)^{P(t|q)}}, \quad (3.15)$$

where $P(t|q) = n(t, q)/|q|$ is the relative frequency of t in q . Therefore, the normalized MLM score is obtained by computing $P(t|\theta_e)$ based on Eq. 3.11 and $P(t|C)$ is taken to be a linear combination of collection language models $\sum_{f \in F} \mu_f P(t|C_f)$. The specific instantiation of the model (fields and weights) is discussed in Section 3.5.1.

Combining MLM and Commonness

Let us point out that MLM ranks entities, mentioned in the query, based on their relevance to the query. This is done irrespective of the specific surface form that is referenced in the query. There is useful prior information associated with surface forms, which is captured in commonness (Eq. 3.9). It therefore makes good sense to combine MLM and commonness. We propose two ways of doing this.

MLMc. The first method, MLMc, simply filters the set of entities that are considered for ranking based on commonness, by applying a threshold c in Eq. 3.10. The setting of c is discussed in Section 3.5.1.

MLMcg. The second method, MLMcg, also performs filtering, exactly as MLMc does. But, in addition to that, it also integrates the commonness scores in a generative model. It ranks entities based on the highest scoring mention; i.e., ranking is dependent not only on the query but also on the specific mention. The model considers the prior probability $P(e)$ in Eq. 3.13 as the probability of a mention m being linked to the entity e and takes the maximum score among the mentions of an entity:

$$P(e|q) \stackrel{\text{rank}}{=} \arg \max_{m \in e} P(e|m)P(q|e), \quad (3.16)$$

where $P(q|e)$ is estimated using MLM (Eqs. 3.11 and 3.15) and $P(m|e)$ is the same as commonness (cf. Eq. 3.9). We show later experimentally that this novel method provides solid results and is more effective than MLM and MLMc.

3.4.3 Disambiguation

The aim of this phase is to find the interpretations of a query, where an interpretation is a set of non-overlapping and semantically compatible entities that are mentioned in the query. Given a ranked list of mention-entity pairs from the previous step, our goal (and, as we argued, this should be the ultimate goal of entity linking in queries) is to identify all interpretations of the query.

We present an algorithm, named *Greedy Interpretation Finding* (GIF), that can detect multiple interpretations of a query; see Algorithm 1. It takes as input a list of mention-entity pairs (m, e) , each associated with a relevance score. Consider an example query “jacksonville fl,” for which the input for the algorithm would be $\{(\text{“jacksonville fl,” JACKSONVILLE FLORIDA}): 0.9, (\text{“jacksonville,”}$

Algorithm 1 Greedy Interpretation Finding (GIF)

Input: Ranked list of mention-entity pairs M ; score threshold s

Output: Interpretations $I = \{E_1, \dots, E_m\}$

```

1: begin
2:    $M' \leftarrow \text{Prune}(M, s)$ 
3:    $M' \leftarrow \text{PruneContainmentMentions}(M')$ 
4:    $I \leftarrow \text{CreateInterpretations}(M')$ 
5:   return  $I$ 
6: end

1: function CREATEINTERPRETATIONS( $M$ )
2:    $I \leftarrow \{\emptyset\}$ 
3:   for  $(m, e)$  in  $M$  do
4:      $h \leftarrow 0$ 
5:     for  $E$  in  $I$  do
6:       if  $\neg \text{hasOverlap}(E, (m, e))$  then
7:          $E.\text{add}((m, e))$ 
8:          $h \leftarrow 1$ 
9:       end if
10:    end for
11:    if  $h == 0$  then
12:       $I.\text{add}(\{(m, e)\})$ 
13:    end if
14:  end for
15:  return  $I$ 
16: end function

```

JACKSONVILLE, FLORIDA): 0.8, (“jacksonville fl,” NAVAL AIR STATION JACKSONVILLE): 0.2}. In the first step (line 2), GIF prunes entities based on absolute scores, controlled by the threshold parameter s . E.g., with a threshold of 0.3, (“jacksonville fl,” NAVAL AIR STATION JACKSONVILLE) would be filtered out here. We note that s is a global parameter, therefore ranking scores must be comparable across queries. (As mentioned in the previous section, our query length normalized ranking scores enable this.) In the next step (line 3), containment mentions are also filtered out, based on their retrieval scores. E.g., out of the two containment mentions “jacksonville fl” and “jacksonville,” only the pair

(“jacksonville fl,” JACKSONVILLE FLORIDA) with the score of 0.9 is kept. Then (in line 4), query interpretations are created in an iterative manner: adding an entity-mention pair to an existing interpretation E , such that it does not overlap with the mentions already present in E . In case it overlaps with all existing interpretations, the mention-entity pair constitutes a new interpretation; this will result in multiple interpretations for a query.

3.5 Experiments

In this section, we present results for the semantic linking and ELQ tasks, using the test collections introduced in Section 3.3 and the methods presented in Section 3.4. These results help to deepen our understanding on the differences between the two tasks of semantic linking and entity linking in queries.

3.5.1 Experimental Setup

Knowledge base. We consider entities present in both DBpedia and Freebase as our reference knowledge base. This choice is made for pragmatic reasons: (i) existing test collections provide annotations (or ground truth) either for one or the other, (ii) Freebase-annotated ClueWeb collections (FACC) [75] are leveraged for mention detection and (reliable) commonness estimation, (iii) entity descriptions in DBpedia provide a solid basis for entity ranking.

Semantic linking. The semantic linking task is evaluated on the YSQLLE test collection. We compare MLM, MLMc, and MLMcg from Section 3.4.2 and also include results for the TAGME system from their public API [69] (following [29]). The commonness threshold c for MLMc and MLMcg (Eq. 3.10) is set to 0.1 by performing a sweep using cross-validation. We use the FACC collection to compute the commonness scores. For ranking entities using MLM, we follow Neumayer et al. [148] and use an index with two fields, name and content, with a weight of 0.2 and 0.8, respectively. The *name* field holds the primary names of the entity (`rdfs:label`, `foaf:name`) and name variants extracted from redirected entities. The *content* field includes the content of the top 1000 most frequent predicates across the whole DBpedia collection. All URIs in the content fields are resolved, i.e., replaced with the name of the entity or title of the page they point to. The index is confined to the entities having a name and a short abstract (i.e., `rdfs:label` and `rdfs:comment`).

Table 3.3: Recall of different sources for mention detection.

	YSQLE	Y-ERD	ERD
KB	0.7489	0.7976	0.8556
Web	0.9127	0.9716	0.9956
KB+Web	0.9163	0.9724	1.0000

Entity linking in queries. For the ELQ task, we report our results on the Y-ERD and ERD-dev test collections. In this case, our reference knowledge base is confined to the entities present in the knowledge base snapshot used at the ERD Challenge [43]. This snapshot contains 2,351,157 entities; taking its intersection with DBpedia resulted in the removal of 39,517 entities. The GIF algorithm (see Section 3.4.3) is applied on top of the candidate entity ranking systems. We use cross-validation (5-fold for Y-ERD and leave-one-out for ERD-dev) for setting the score threshold of GIF by performing a sweep for the parameter s . In addition, we report on another method, called *TopRanked*. It uses the best performing entity ranking approach (MLMcg) and forms a single interpretation set from the top ranked entity.

3.5.2 Mention Detection

The mention detection component is shared by both the semantic linking and ELQ tasks, therefore we evaluate it on its own account. Specifically, we compare three options based on the source(s) of surface forms, as described in Section 3.4.1: (i) DBpedia (KB), (ii) web corpora (Web), and (iii) the combination of both (KB+Web). As this step is recall oriented, (i.e., all entity matches should be retrieved), we only report on recall.

Table 3.3 presents the results. We find that the machine-annotated web corpora provides a rich source of entity surface forms for this task and is a better source than DBpedia alone. Not surprisingly, the combination of the two sources yields the highest recall, albeit the improvement over Web is marginal. We also note that while recall is nearly perfect on the interpretation finding datasets (Y-ERD and ERD), it is a bit lower for YSQLE. Recall that YSQLE is created for evaluating the semantic linking task, where implicit entity mentions are also considered as relevant; these are not captured by our dictionary-based mention detection approach and would need to be identified by different means.

Table 3.4: Semantic linking results on the YSQL dataset.

	MAP	P@1	MRR
MLM	0.4582	0.3601	0.4638
MLMc	0.6228	0.5413	0.6312
MLMcg	0.7078	0.6403	0.7151
TAGME [†]	0.6230	0.6016	0.6385

[†]TAGME is an entity linking system and should not be evaluated on the semantic linking task using rank-based measures.

3.5.3 Semantic Linking

Table 3.4 presents the results for semantic linking. Given that this is a ranking task, we report on mean average precision (MAP), precision at position 1 (P@1), and mean reciprocal rank (MRR). Though we include results for TAGME, we note that this comparison, despite having been done in prior work (e.g., in [29]), is an unfair one. TAGME is an entity linking system that should not be evaluated using rank-based measures. The very reason for including TAGME is to illustrate that a simple semantic linking approach can achieve improvements over a state-of-the-art entity linking system, but this claim would be misleading. We find that MLMcg is the most effective method; it shows that incorporating commonness in a generative model (MLMcg) is better than using commonness as a filter before ranking entities (MLMc).

3.5.4 Entity Linking in Queries

Tables 3.5 and 3.6 present the results for interpretation finding on the Y-ERD and ERD-dev datasets, respectively. Two sets of evaluation measures are used: (i) strict (which is the same as in [43]) and (ii) lenient (Section 3.2). We notice at first glance that the TopRanked baseline is considerably worse than the other approaches. This shows that, even though there are many queries containing a single entity in our data sets (cf. Table 3.2), forming sets of entities is a crucial aspect of the ELQ task. The GIF algorithm in combination with MLMcg delivers solid performance and is the best performing of all approaches in all but one setting.

Comparing the two evaluation measures, lenient evaluation gives higher results for all systems. This is in line with our expectations based on the the-

Table 3.5: Entity linking in queries on the Y-ERD dataset.

Method	Strict eval.			Lenient eval.		
	P	R	F	P	R	F
TopRanked	0.4554	0.4542	0.4545	0.4771	0.465	0.4689
MLM-GIF	0.5259	0.5254	0.5255	0.5363	0.5387	0.5361
MLMc-GIF	0.6351	0.6354	0.6348	0.6422	0.642	0.6409
MLMcg-GIF	0.7191	0.7213	0.7195	0.7305	0.7308	0.7288

Table 3.6: Entity linking in queries on the ERD-dev dataset.

Method	Strict eval.			Lenient eval.		
	P	R	F	P	R	F
TopRanked	0.3846	0.3645	0.3700	0.4231	0.3837	0.3956
MLM-GIF	0.5824	0.5608	0.5659	0.5934	0.5718	0.5760
MLMc-GIF	0.7253	0.7037	0.7088	0.7445	0.7174	0.7234
MLMcg-GIF	0.7143	0.7125	0.7114	0.7335	0.7262	0.7260

oretical definitions of the measures (Section 3.4.3); when an interpretation is incomplete, yet contains relevant entities, the strict evaluation does not give any credit for returning correct entities, whereas the lenient one does.

3.6 Discussion

We now answer our research question based on the results presented in Section 3.5:

RQ1. How can the inherent ambiguity of entity annotations in queries be handled and evaluated?

Entity linking in queries should ultimately be addressed as finding sets of interpretations, where an interpretation is a set of non-overlapping entities that are semantically related to each other. If the query is ambiguous, with little or no context, there exist multiple interpretations, all of which should be found. Otherwise, a single interpretation should be detected, which is similar to the traditional entity linking task for documents. Determining when the query has no interpretations (in terms of entity annotations) is also a crucial part of the

problem that should be addressed (and considered in the evaluation). The *semantic linking* task (Section 3.1.3), which ranks entities based on their relevance to the query, serves a different purpose and should not be considered as an entity linking task, even for the simplified scenario of finding a single interpretation. This is because relevant entities can be overlapping and are not required to be semantically related to each other. Furthermore, entity disambiguation is an essential part of entity linking, an aspect that is completely ignored in semantic linking. A number of earlier studies refer to entity linking, while what they do in fact is semantic linking [29, 135, 136, 151]. Comparing semantic linking to results generated by traditional entity linking methods is inappropriate (cf. Section 3.5.3).

For entity linking in queries, similar to the traditional entity linking task [69, 141, 143], evaluation uses set-based measures (precision, recall and F-measure). However, since the output is a set of interpretations (and not entities) the evaluation methodology is different. The method presented in [43] considers the exact match between the retrieved sets and the ground truth, which is rather strict. The lenient evaluation method (Section 3.4.3), on the other hand, combines interpretation-based and entity-based evaluations. For semantic linking, standard rank-based measures (MAP, MRR, P@1) can be employed.

In the course of answering the above question, a couple of subsequent questions arose, which we answer below:

What are the similarities and differences between semantic linking and entity linking in queries in terms of approaches? The semantic linking and ELQ tasks can be addressed by using a similar pipeline architecture (see Section 3.4). Both tasks share the first component, mention detection, which can effectively be addressed by combining surface forms stored in knowledge bases and extracted from a large machine-annotated web corpora (cf. Section 3.5.2). The second component, which generates a ranked list of the mentioned entities based on their relevance to the query, can also be shared. One issue that requires special attention is the question of implicit mentions, that is, entities that are referred to but not explicitly mentioned in the query (e.g., “Obama’s mother”). These are not identified by the mention detection step and consequently not considered for ranking either. One interesting research challenge in semantic linking is finding these referred entities. For entity linking in queries, the third component is responsible for forming (possibly multiple) sets of interpretations. This is a highly nontrivial subtask that makes

entity linking in queries substantially more difficult than semantic linking.

As most queries have a single interpretation, how much effort should be expedited to find multiple interpretations? Although having multiple interpretations is an intrinsic feature of entity linking in queries, most of the queries in our test collections (both ERD and Y-ERD) have a single interpretation (see Table 3.2). This implies that a system can achieve high overall score by focusing on returning a single interpretation. This is also evidenced by the ERD Challenge results, where the top two performing systems [49, 52] return a single interpretation. We note that returning multiple interpretations, without hurting queries with single a interpretation, is an open research question.

3.7 Summary

In this chapter we have addressed fundamental questions in the problem area of query understanding through entity annotations of queries. We have differentiated between two tasks, semantic linking and entity linking in queries. The former ranks entities that are related to (but not necessarily explicitly mentioned in) the query, while the latter aims to identify sets of semantically related entities that are mentioned in the query, and is able to return more than one of such sets if the query has multiple interpretations. We have discussed evaluation methodology and carefully examined publicly available test collections for both tasks, and introduced a large, manually curated test collection for entity linking in queries.

In the next chapter, we make an effort at establishing a baseline for the ELQ task using a state-of-the-art entity linking approach. In Chapter 5 we investigate methods for effective and efficient entity linking in queries. We will also employ ELQ as an integral element of other entity-oriented tasks addressed in Chapters 6 and 7 of this thesis.

Chapter 4

Establishing a Baseline for Entity Linking in Queries

In Chapter 3 we have introduced the task of entity linking in queries (ELQ), the corresponding evaluation measures, and publicly available datasets for training and evaluation purposes. In this chapter, we aim to establish a strong baseline for this task. While ELQ has just recently received attention, the general problem of entity linking has been widely studied over the past years and various approaches have been proposed to annotate both long and short text with entities [56, 69, 87, 114, 141, 143]. The research question we address in this chapter is:

RQ2. How does a state-of-the-art entity linking approach perform on the ELQ task?

Among the top-performing entity linking systems, TAGME [69] is one of the most popular and influential ones, and is shown to be a competitive baseline in various studies [29, 123, 136, 159, 204]. It is specifically designed for efficient (“on-the-fly”) annotation of short texts (like tweets and search snippets), and has great potential to be considered as a baseline for the ELQ task. Although applying TAGME to the ELQ task restricts the output to a single interpretation, it can still deliver solid performance.

TAGME comes with a web-based interface and a RESTful API,¹ which offers a convenient way of using the system, without implementing the actual

¹<http://tagme.di.unipi.it/>

approach. For employing TAGME as a baseline for the ELQ task, however, we opt to re-implement the approach for two main reasons: (i) to avoid relying on an external service, and dealing with the network overhead that comes with an online service; (ii) to use a different knowledge base from what is used in the TAGME API. In the course of implementing the TAGME approach, we encountered some technical challenges that have not always been dealt with properly in the original paper and accordingly in its re-implementations [46, 132]. In addition, the effectiveness of our TAGME implementation was way different from what we could get from the public API. To address these issues, we decided to step back and first compare our implementation of TAGME with the numbers reported in the original paper, using the same the dataset and Wikipedia version. Thus, we set out the task of examining the repeatability, reproducibility, and generalizability of the TAGME system in a principled way. The SIGIR 2015 workshop on Reproducibility, Inexplicability, and Generalizability of Results (RIGOR) [5] defined these properties as follows:²

- *Repeatability*: “Repeating a previous result under the original conditions (e.g., same dataset and system configuration).”
- *Reproducibility*: “Reproducing a previous result under different, but comparable conditions (e.g., different, but comparable dataset).”
- *Generalizability*: “Applying an existing, empirically validated technique to a different IR task/domain than the original.”

We address each of these aspects in this chapter, as explained below.

Repeatability. Although TAGME facilitates comparison by providing a publicly available API, it is not sufficient for the purpose of repeatability. The main reason is that the API works much like a black-box; it is impossible to check whether it corresponds to the system described in [69]. Actually, it is acknowledged that the API deviates from the original publication,³ but the differences are not documented anywhere. Another limiting factor is that the API cannot be used for efficiency comparisons due to the network overhead. We report on the challenges around repeating the experiments in [69] and discuss why the results are not repeatable.

Reproducibility. TAGME has been re-implemented in several research papers, see, e.g., [45, 46], these, however, do not report on the reproducibility of results. In addition, there are some technical challenges involved in the TAGME

²<https://sites.google.com/site/sigirrigor/>

³http://tagme.di.unipi.it/tagme_help.html and is also mentioned in [51, 184]

approach that have not always been dealt with properly in the original paper and accordingly in these re-implementations (as confirmed by some of the respective authors).⁴ We examine the reproducibility of TAGME, as introduced in [69], and show that some of the results are not reproducible, while others are reproducible only through the TAGME API.

Generalizability. We test generalizability by applying TAGME on the task of entity linking in queries. The main difference between conventional entity linking and ELQ is that the latter accepts that a query might have multiple interpretations, i.e., the output is not a single annotation, but (possibly multiple) sets of entities that are semantically related to each other. Even though TAGME has been developed for a different problem (where only a single interpretation is returned), we show that it is generalizable to the ELQ task.

This study enables us to understand how TAGME should be used as a baseline for the ELQ task. It further provides us with invaluable insights about reproducibility, which is an important desiderata for reliable and extensible research. We wish to make a disclaimer that in the course of this study, we made a best effort to reproduce the results presented in [69] based on the information available to us: the TAGME papers [69, 70] and the source code kindly provided by the authors. Our main goal is to establish a baseline for the ELQ task and to learn about reproducibility, and by no means is intended to be a criticism of TAGME. The resources corresponding to this chapter as well as detailed responses from the TAGME authors (and any possible future updates) are made publicly available; see Appendix A.2.

In the remainder of this chapter, we continue with an overview of the TAGME approach in Section 4.1. We examine the repeatability, reproducibility, and generalizability aspects in Section 4.2–4.4, and end with discussion and summary sections.

4.1 Overview of TAGME

In this section, we provide an overview of the TAGME approach, as well as the test collections and evaluation measures used in the TAGME papers [69, 70].

4.1.1 Approach

TAGME performs entity linking in a pipeline of three steps: (i) parsing, (ii) disambiguation, and (iii) pruning (see Figure 4.1). We note that while Ferragina

⁴Personal communication with authors of [46, 69, 92]

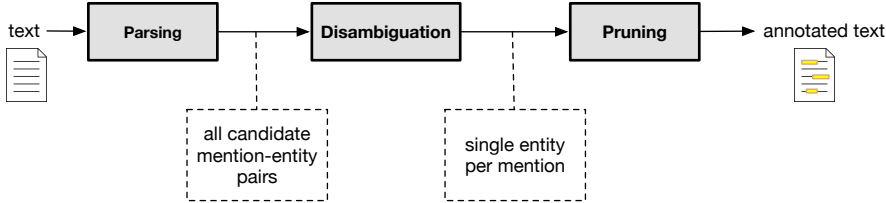


Figure 4.1: Annotation pipeline in the TAGME system.

and Scaiella [69] describe multiple approaches for the last two steps, we limit ourselves to their final suggestions; these are also the choices implemented in the TAGME API.

Before describing the TAGME pipeline, let us define the notation used throughout this chapter. Entity linking is the task of annotating an input text T with entities E from a reference knowledge base, which is Wikipedia here. T contains a set of entity mentions M , where each mention $m \in M$ can refer to a set of candidate entities $E(m)$. These need to be disambiguated such that each mention points to a single entity $e(m)$.

Parsing

In the first step, TAGME parses the input text and performs mention detection using a dictionary of entity surface forms. For each entry (surface form), the set of entities recognized by that name is recorded. This dictionary is built by extracting entity surface forms from four sources: anchor texts of Wikipedia articles, redirect pages, Wikipedia page titles, and variants of titles (removing parts after the comma or in parentheses). Surface forms consisting of numbers only or of a single character, or below a certain number of occurrences (2) are discarded. Further filtering is performed on the surface forms with low *link probability* (i.e., < 0.001). Link probability is defined as:

$$\text{lp}(m) = P(\text{link}|m) = \frac{\text{link}(m)}{\text{freq}(m)}, \quad (4.1)$$

where $\text{freq}(m)$ denotes the total number of times mention m occurs in Wikipedia (as a link or not), and $\text{link}(m)$ is the number of times mention m appears as a link.

To detect entity mentions, TAGME matches all n -grams of the input text, up to $n = 6$, against the surface form dictionary. For an n -gram contained by another one, TAGME drops the shorter n -gram, if it has lower link probability than the longer one. The output of this step is a set of mentions with their corresponding candidate entities.

Disambiguation

Entity disambiguation in TAGME is performed using a voting schema, that is, the score of each mention-entity pair is computed as the sum of votes given by candidate entities of all other mentions in the text. Formally, given the set of mentions M , the relevance score of the entity e to the mention m is defined as:

$$\text{rel}(m, e) = \sum_{m' \in M - \{m\}} \text{vote}(m', e), \quad (4.2)$$

where $\text{vote}(m', e)$ denotes the agreement between entities of mention m' and the entity e , computed as follows:

$$\text{vote}(m', e) = \frac{\sum_{e' \in E(m')} \text{relatedness}(e, e') \cdot \text{commonness}(e', m')}{|E(m')|}. \quad (4.3)$$

Commonness is the probability of an entity being the link target of a given mention [134]:

$$\text{commonness}(e', m') = P(e' | m') = \frac{\text{link}(e', m')}{\text{link}(m')}, \quad (4.4)$$

where $\text{link}(e', m')$ is the number of times entity e' is used as a link destination for m' and $\text{link}(m')$ is the total number of times m' appears as a link. *Relatedness* measures the semantic association between two entities [195]:

$$\text{relatedness}(e, e') = \frac{\log(\max(|\text{in}(e)|, |\text{in}(e')|)) - \log(|\text{in}(e) \cap \text{in}(e')|)}{\log(|E|) - \log(\min(|\text{in}(e)|, |\text{in}(e')|))}, \quad (4.5)$$

where $\text{in}(e)$ is the set of entities linking to entity e (i.e., in-links) and $|E|$ is the total number of entities.

Once all candidate entities are scored using Eq. 4.2, TAGME selects the best entity for each mention. Two approaches are suggested for this purpose: (i) disambiguation by classifier (DC) and (ii) disambiguation by threshold (DT), of which the latter is selected as the final choice. Due to efficiency concerns,

entities with commonness below a given threshold τ are discarded from the DT computations. The set of commonness-filtered candidate entities for mention m is $E_\tau(m) = \{e \in M(e) | \text{commonness}(m, e) \geq \tau\}$. Then, DT considers the top- ϵ entities for each mention and then selects the one with the highest commonness score:

$$m(e) = \arg \max_e \{\text{commonness}(m, e) : e \in E_\tau(m) \wedge e \in \text{top}_\epsilon[\text{rel}(m, e)]\}. \quad (4.6)$$

At the end of this stage, each mention in the input text is assigned a single entity, which is the most pertinent one to the input text.

Pruning

The aim of the pruning step is to filter out non-meaningful annotations, i.e., assign *NIL* to the mentions that should not be linked to any entity. TAGME hinges on two features to perform pruning: *link probability* (Eq. 4.1) and *coherence*. The coherence of an entity is computed with respect to the candidate annotations of all the other mentions in the text:

$$\text{coherence}(e, T) = \frac{\sum_{e' \in E(T) - \{e\}} \text{relatedness}(e, e')}{|E(T)| - 1}, \quad (4.7)$$

where $E(T)$ is the set of distinct entities assigned to the mentions in the input text. TAGME takes the average of the link probability and the coherence score to generate a ρ score for each entity, which is then compared to the pruning threshold ρ_{NA} . Entities with $\rho < \rho_{\text{NA}}$ are discarded, while the rest of them are served as the final result.

4.1.2 Test Collections

Two test collections are used in [69]: WIKI-DISAMB30 and WIKI-ANNOT30. Both comprise snippets of around 30 words, extracted from a Wikipedia snapshot of November 2009, and are made publicly available.⁵ In WIKI-DISAMB30, each snippet is linked to a single entity; in WIKI-ANNOT30 all entity mentions are annotated. We note that the sizes of these test collections (number of snippets) deviate from what is reported in the TAGME paper: WIKI-DISAMB30 and WIKI-ANNOT30 contain around 2M and 185K snippets, while the reported numbers are 1.4M and 180K, respectively. This suggests that the published test collections might be different from the ones used in [69].

⁵<http://acube.di.unipi.it/tagme-dataset/>

4.1.3 Evaluation Measures

TAGME is evaluated using three variations of precision and recall. The so-called *standard* precision and recall (P and R), are employed for evaluating the disambiguation phase, using the WIKI-DISAMB30 test collection. The two other measures, *annotation* and *topics* precision and recall are employed for measuring the end-to-end performance on the WIKI-ANNOT30 test collection. The annotation measures (P_{ann} and R_{ann}) compare both the mention and the entity against the ground truth, while the topics measures (P_{topics} and R_{topics}) only consider entity matches. The TAGME papers [69, 70] provide little information about the evaluation measures. In particular, the computation of the *standard* precision and recall is rather unclear; we discuss it later in Section 4.3.2. Details are missing regarding the two other measures too: (i) How are overall precision, recall and F-measure computed for the annotation measures? Are they micro- or macro-averaged? (ii) What are the matching criteria for the annotation measures? Are partially matching mentions accepted or only exact matches? In what follows, we formally define the annotation and topics measures, based on the most likely interpretation we established from the TAGME paper and from our experiments.

We write $\mathcal{G}(T) = \{(\hat{m}_1, \hat{e}_1), \dots, (\hat{m}_m, \hat{e}_m)\}$ for ground truth annotations of the input text T , and $\mathcal{S}(T) = \{(m_1, e_1), \dots, (m_n, e_n)\}$ for the annotations identified by the system. Neither $\mathcal{G}(T)$ nor $\mathcal{S}(T)$ contains NULL annotations. The TAGME paper follows [114], which uses macro-averaging in computing annotation precision and recall:⁶

$$P_{ann} = \frac{|\mathcal{G}(T) \cap \mathcal{S}(T)|}{|\mathcal{S}(T)|}, \quad R_{ann} = \frac{|\mathcal{G}(T) \cap \mathcal{S}(T)|}{|\mathcal{G}(T)|}. \quad (4.8)$$

The annotation (\hat{m}, \hat{e}) matches (m, e) if two conditions are fulfilled: (i) entities match ($\hat{e} = e$), and (ii) mentions match or contain each other ($\hat{m} = m$ or $\hat{m} \in m$ or $m \in \hat{m}$). We note that the TAGME paper refers to “perfect match” of the mentions, while we use a more relaxed version of matching (by considering containment matches). This relaxation results in the highest possible P_{ann} and R_{ann} , but even those are below the numbers reported in [69] (cf. Section 4.3.2).

⁶As explained later by the TAGME authors, they in fact used micro-averaging. This contradicts the referred paper [114], which explicitly defines P_{ann} and R_{ann} as being macro-averaged.

The topics precision and recall (P_{topics} and R_{topics}) [143] only consider entity matches ($\hat{e} = e$) and are micro-averaged over the set of all texts \mathcal{F} :

$$P_{topics} = \frac{\sum_{T \in \mathcal{F}} |\mathcal{G}(T) \cap \mathcal{S}(T)|}{\sum_{T \in \mathcal{F}} |\mathcal{S}(T)|}, \quad R_{topics} = \frac{\sum_{T \in \mathcal{F}} |\mathcal{G}(T) \cap \mathcal{S}(T)|}{\sum_{T \in \mathcal{F}} |\mathcal{G}(T)|}. \quad (4.9)$$

For all measures the overall F-measure is computed from the overall precision and recall.

4.2 Repeatability

By definition, *repeatability* means that a system should be implemented under the same conditions as the reference system. In our case, the repeatability of the TAGME experiments in [69] is dependent on the availability of (i) the knowledge base and (ii) the test collections (text snippets and gold standard annotations).

The reference knowledge base is Wikipedia, specifically, the TAGME paper uses a dump from November 2009, while the API employs a dump from July 2012. Unfortunately, neither of these dumps is available on the Web nor could be provided by the TAGME authors upon request. We encountered problems with the test collections too. As we already explained in Section 4.1.2, there are discrepancies between the number of snippets the test collections (WIKI-DISAMB30 and WIKI-ANNOT30) actually contain and what is reported in the paper. The latter number is higher, suggesting that the results in [69] are based only on subsets of the collections.⁷ Further, WIKI-DISAMB30 is split into training and test sets in the TAGME paper, but those splits are not available.

Due to these reasons, which could all be classified under the general heading of *unavailability of data*, we conclude that the TAGME experiments in [69] are not repeatable. In the next section, we make a best effort at establishing the most similar conditions, that is, we attempt to reproduce their results.

4.3 Reproducibility

This section reports on our attempts to reproduce the results presented in the TAGME paper [69]. The closest publicly available Wikipedia dump is from April

⁷It was later explained by the TAGME authors that they actually used only 1.4M out of 2M snippets from WIKI-DISAMB30, as Weka could not load more than that into memory. From WIKI-ANNOT30 they used all snippets, the difference is merely a matter of approximation.

2010,⁸ which is about five months newer than the one used in [69]. On a side note we should mention that we were (negatively) surprised by how difficult it proved to find Wikipedia snapshots from the past, esp. from this period. We have (re)implemented TAGME based on the description in the TAGME papers [69, 70] and, when in doubt, we checked the source code. For a reference comparison, we also include the results from (i) the TAGME API and (ii) the Dexter entity linking framework [46]. Even though the implementation in Dexter (specifically, the parser) slightly deviates from the original TAGME system, it is still useful for validation, as that implementation is done by a third (independent) group of researchers. We do not include results from running the source code provided to us because it requires the Wikipedia dump in a format that is no longer available for the 2010 dump we have access to; running it on a newer Wikipedia version would give results identical to the API. In what follows, we present the challenges we encountered during the implementation in Section 4.3.1 and then report on the results in Section 4.3.2.

4.3.1 Implementation

During the (re)implementation of TAGME, we encountered several technical challenges, which we describe here. These could be traced back to differences between the approach described in the paper and the source code provided by the authors. Without addressing these differences, the results generated by our implementation are far from what is expected and are significantly worse than those by the original system.

Link probability computation. Link probability is one of the main statistical features used in TAGME. We noticed that the computation of link probability in TAGME deviates from what is defined in Eq. 4.1: instead of computing the denominator $freq(m)$ as the number of occurrences of mention m in Wikipedia, TAGME computes the number of documents that mention m appears in. Essentially, document frequency is used instead of term (phrase) frequency. This is most likely due to efficiency considerations, as the former is much cheaper to compute. However, a lower denominator in Eq. 4.1 means that the resulting link probability is a higher value than it is supposed to be. In fact, this change in the implementation means that it is actually not link probability, but more

⁸<https://archive.org/details/enwiki.20100408>

like *keyphraseness* that is being computed. Keyphraseness [141] is defined as:

$$\text{kp}(m) = P(\text{keyword}|m) = \frac{\text{key}(m)}{\text{df}(m)}, \quad (4.10)$$

where $\text{key}(m)$ denotes number of Wikipedia articles where mention m is selected as a keyword, i.e., linked to an entity (any entity), and $\text{df}(m)$ is the number of articles containing the mention m . Since in Wikipedia a link is typically created only for the first occurrence of an entity ($\text{link}(m) \approx \text{key}(m)$), we can assume that the numerator of link probability and keyphraseness are identical. This would mean that TAGME as a matter of fact uses keyphraseness. Nevertheless, as our goal in this chapter is to reproduce the TAGME results, we followed their implementation of this feature, i.e., $\text{link}(m)/\text{df}(m)$.⁹

Relatedness computation. We observed that the *relatedness* score, defined in Eq. 4.5, is computed as $1 - \text{relatedness}(e, e')$, furthermore, for the entities with zero inlinks or no common inlinks, the score is set to zero. These details are not explicitly mentioned in the paper, while they have significant impact on the overall effectiveness of TAGME.

Pruning based on commonness. In addition to the filtering methods mentioned in the parsing step (cf. Section 4.1.1), TAGME filters entities with commonness score below 0.001, but it is not documented in the TAGME papers. We followed this filtering approach, as it makes the system considerably faster.

4.3.2 Results

We report results for the intermediate disambiguation phase and for the end-to-end entity linking task. For all reproducibility experiments, we set the ρ_{NA} threshold to 0.2, as it delivers the best results and is also the recommended value in the TAGME paper.

Disambiguation phase. For evaluating the disambiguation phase, we submitted the snippets from the WIKI-DISAMB30 test collection to the TAGME API, with the pruning threshold set to 0. This setting ensures that no pruning is performed and the output we get back is what is supposed to be the outcome of

⁹The proper implementation of link probability would result in lower values (as the denominator would be higher) and would likely require a different threshold value than what is suggested in [69]. This goes beyond the scope of this chapter.

Table 4.1: Results of TAGME reproducibility on the WIKI-DISAMB30 test collection.

Method	P	R	F
Original paper [69]	0.915	0.909	0.912
TAGME API	0.775	0.775	0.775

Table 4.2: Results of TAGME reproducibility on the WIKI-ANNOT30 test collection.

Method	P _{ann}	R _{ann}	F _{ann}	P _{topics}	R _{topics}	F _{topics}
Original paper [69]	0.7627	0.7608	0.7617	0.7841	0.7748	0.7794
TAGME API	0.6945	0.7136	0.7039	0.7017	0.7406	0.7206
TAGME-wp10 (our)	0.6143	0.4987	0.5505	0.6499	0.5248	0.5807
Dexter	0.5722	0.5959	0.5838	0.6141	0.6494	0.6313

the disambiguation phase. We tried different methods for computing precision and recall, but we were unable to get the results that are reported in the original TAGME paper (see Table 4.1). We therefore relaxed the evaluation conditions in the following way: if any of the entities returned by the disambiguation phase matches the ground truth entity for the given snippet, then we set both precision and recall to 1; otherwise they are set to 0. This gives us an upper bound for the performance that can be achieved on the WIKI-DISAMB30 test collection; any other interpretation of precision or recall would result in a lower number. What we found is that even with these relaxed conditions the F-score is far below the reported value (0.775 vs. 0.912). One reason for the differences could be the discrepancy between the number of snippets in the test collection and the ones used in [69]. Given the magnitude of the differences, even against their own API, we decided not to go further to get the results for our implementation of TAGME. We conclude that this set of results is not reproducible, due to *insufficient experimental details* (test collection and measures).

End-to-end performance. Table 4.2 shows end-to-end system performance according to the following implementations: the TAGME API, our implementation using a Wikipedia snapshot from April 2010, and the Dexter implementation using a Wikipedia snapshot from March 2013. For all experiments, we

compute the evaluation measures described in Section 4.1.3. We observe that the API results are lower than in the original paper, but the difference is below 10%. We attribute this to the fact that the ground truth is generated from a 2009 version of Wikipedia, while the API is based on the version from 2012.

Concerning our implementation and Dexter (bottom two rows in Table 4.2) we find that they are relatively close to each other, but both of them are lower than the TAGME API results; the relative difference to the API results is -19% for our implementation and -12% for Dexter in F_{topics} score. Ceccarelli et al. [46] also report on deviations, but they attribute these to the processing of Wikipedia: “we observed that our implementation [...] behaves only slightly worse than TAGME after the top 5 results, probably due to a different processing of Wikipedia.” The difference between Dexter and our implementation stems from the parsing step. Dexter relies on its own parsing method and removes overlapping mentions at the end of the annotation process. We, on the other hand, follow TAGME and delete overlapping mentions in the parsing step (cf. Section 4.1.1). By analyzing our results, we observed that this parsing policy resulted in early pruning of some correct entities and led accordingly to lower results.

Our experiments show that the end-to-end results reported in [69] are reproducible through the TAGME API, but not by (re)implementation of the approach by a third partner. This is due to *undocumented deviations from the published description*.

4.4 Generalizability

We now test the generalizability of TAGME on the ELQ task. As discussed in Chapter 3, the aim of ELQ is to detect all possible entity linking *interpretations* of the query. This is different from conventional entity linking, where a single annotation is created. In other words, the output of conventional entity linking systems is a set of mention-entity pairs, while entity linking in queries returns set(s) of entity sets. Applying a conventional entity linker to the ELQ task restricts the output to a single interpretation, but can deliver solid performance nonetheless [43]. We detail our experimental setup in Section 4.4.1 and report on the results in Section 4.4.2.

Table 4.3: TAGME results for the entity linking in queries task.

Method	ERD-dev			Y-ERD		
	P_{strict}	R_{strict}	F_{strict}	P_{strict}	R_{strict}	F_{strict}
TAGME API	0.8352	0.8062	0.8204	0.7173	0.7163	0.7168
TAGME-wp10 (our)	0.7143	0.7088	0.7115	0.6518	0.6515	0.6517
TAGME-wp12 (our)	0.7363	0.7234	0.7298	0.6535	0.6532	0.6533
Dexter	0.7363	0.7073	0.7215	0.6989	0.6979	0.6984

4.4.1 Experimental Setup

Implementations. We compare four different implementations to assess the generalizability of TAGME to the ELQ task: the TAGME API, our implementation of TAGME with two different Wikipedia versions, one from April 2010 and another from May 2012 (which is part of the ClueWeb12 collection), and Dexter’s implementation of TAGME. Including results using the 2012 version of Wikipedia facilitates a better comparison between the TAGME API and our implementation, as they both use similar Wikipedia dumps. It also demonstrates how the version of Wikipedia might affect the results.

Datasets and evaluation measures. We use two publicly available test collections developed for the ELQ task: ERD-dev [43] and Y-ERD [92]. ERD-dev includes 99 queries, while Y-ERD offers a larger selection, containing 2398 queries (cf. Section 3.3). The annotations in these test collections are confined to proper noun entities from the Freebase snapshot provided by the ERD challenge [43]. We therefore remove entities that are not present in this snapshot in a post-filtering step. In all the experiments, ρ_{NA} is set to 0.1, as it delivers the highest results both for the API and for our implementations, and is also the recommendation of the TAGME API. Evaluation is performed in terms of precision, recall, and F-measure, using the strict evaluation measures described in Section 3.2.

4.4.2 Results

Table 4.3 presents the TAGME generalizability results. Similar to the reproducibility experiments, we find that the TAGME API provides substantially better results than any of the other implementations. The most fair comparison between Dexter and our implementations is the one against TAGME-wp12, as

that has the Wikipedia dump closest in date. For ERD-dev they deliver similar results, while for Y-ERD Dexter has a higher F-score (but the relative difference is below 10%). Concerning different Wikipedia versions, the more recent one performs better on the ERD-dev test collection, while the difference is negligible for Y-ERD. If we take the larger test collection, Y-ERD, to be the more representative one, then we find that TAGME API > Dexter > TAGME-wp10, which is consistent with the reproducibility results in Table 4.2. However, the relative differences between the approaches are smaller here. We thus conclude that the TAGME approach can be generalized to the ELQ task.

4.5 Discussion

Based on our findings in Sections 4.2–4.4, we answer our main research question:

RQ2. How does a state-of-the-art entity linking approach perform on the ELQ task?

TAGME can be used as a baseline approach for the ELQ task. For a reliable and meaningful comparison between TAGME and a newly proposed entity linking method, the TAGME approach should be (re)implemented, like it has been done in some prior work [45]. We recommend to use the TAGME API, much like a black-box, when entity linking is performed as part of a larger task.

4.5.1 Further Investigations

TAGME is an outstanding entity linking system. The authors offer invaluable resources for the reproducibility of their approach: the test collections, source code, and a RESTful API. However, one important question that is raised by our experimental results is: *Where do the result differences (between the TAGME API and third party implementation) stem from?* Upon the acceptance of our TAGME reproducibility paper [94], the TAGME authors clarified some of the issues that surfaced in this study. This information came only after the paper was accepted, even though we have raised our questions during the writing of the paper (at that time, however, the reply we got only included the source code and the fact that they no longer have the Wikipedia dumps used in the paper). We integrated their responses throughout our study as much as it was possible; we include the rest of them here.

First, it turns out that the public API as well as the provided source code correspond to a newer, updated version (“version 2”) of TAGME. The source code for the original version (“version 1”) described in the TAGME papers [69, 70] is no longer available. This means that even if we managed to find the Wikipedia dump used in the TAGME papers and ran their source code, we would have not been able to reproduce their results. Furthermore, TAGME performs additional non-documented optimizations when parsing the spots, filtering inappropriate spots, and computing relatedness, as explained by the authors.

Another reason for the differences in performance might have to do with how links are extracted from Wikipedia. TAGME uses wiki page-to-page link records, while our implementation (as well as Dexter’s) extracts links from the body of the pages. This affects the computation of relatedness, as the former source contains 20% more links than the latter. (It should be noted that this file was not available for the 2010 and 2012 Wikipedia dumps.) The authors also clarified that all the evaluation measures are micro-averaged and explained how the disambiguation phase was evaluated. We refer the interested reader to the online appendix¹⁰ of this chapter for further details.

4.5.2 Lessons Learned

This study has led to invaluable insights about reproducibility requirements, which we list here:

1. All technical details that affect effectiveness or efficiency should be explained (or at least mentioned) in the paper; sharing the source code helps, but finding answers in a large codebase can be highly non-trivial.
2. If there are differences between the published approach and publicly made available source code or API (typically, the latter being an updated version), those should be made explicit.
3. It is encouraged that authors keep all data sources used in a published paper (in particular, historical Wikipedia dumps, esp. in some specific format, are more difficult to find than one might think), so that these can be shared upon requests from other researchers.
4. Evaluation measures should be explained in detail.

¹⁰<http://bit.ly/tagme-rep>

Maintaining an “online appendix” to a publication is a practical way of providing some of these extra details that would not fit in the paper due to space limits, and would have the additional advantage of being easily editable and extensible.

4.6 Summary

In this chapter we have aimed at establishing a baseline for the task of entity linking in queries using a state-of-the-art entity linking approach. We have introduced TAGME as a tentative baseline; it is an outstanding entity linking system, has the ability of coping with very short texts like queries, and comes with invaluable resources for the reproducibility of their approach. We have attempted to (re)implement the system described in [69], studying the repeatability, reproducibility and generalizability of TAGME to the ELQ task. Our experiments have shown that some of the results are not reproducible, even with the API provided by the authors. For the rest of the results, we have found that (i) the results reported in the chapter are higher than what can be reproduced using their API, and (ii) the TAGME API gives higher numbers than what is achievable by a third-party implementation (both our and that of Dexter [45]) (iii) the TAGME approach can be generalized to the ELQ task. Based on our findings, we will use TAGME as a baseline for the ELQ task in the next chapter.

Chapter 5

Methods for Entity Linking in Queries

In Chapter 3, we introduced a pipeline architecture for addressing the task of entity linking in queries (ELQ), and presented basic methods for each element of the pipeline. In this chapter, we focus on the most challenging elements of the pipeline: candidate entity ranking and disambiguation. We propose a number of methods for each of them, with the overall goal of finding an ELQ approach that can strike a balance between effectiveness and efficiency.

Entity linking has been extensively studied for long texts [56, 83, 87, 114, 141, 143, 153]. Despite the large variety of approaches, there are two main components that are present in all entity linking systems: (i) *candidate entity ranking*, i.e., identifying entities that can be possibly linked to a mention, and (ii) *disambiguation*, i.e., selecting the best entity (or none) for each detected mention. There is also a general consensus on the two main categories of features that are needed for effective entity linking: (i) contextual similarity between a candidate entity and the surrounding text of the entity mention, and (ii) interdependence between all entity linking decisions in the text (extracted from the underlying KB). Previous studies [46, 83] have investigated these aspects in a unified framework and derived general recommendations for entity linking in documents. Entity linking in queries, however, has only recently started to draw attention [43, 53, 92] and such systematic evaluation of the different components has not been conducted until now.

What is special about entity linking in queries? First, queries are short, noisy text fragments where the ambiguity of a mention may not be resolved because of the limited context. That is, a mention can possibly be linked to more than one entity (see Table 5.1 for examples). This is unlike entity linking in documents, where it is assumed that there is enough context for disambiguation. Second, ELQ is an online process that happens during query-time, meaning that it should be performed under serious time constraints (in contrast with traditional entity linking, which is offline). The ideal solution is not necessarily the most effective one, but the one that represents the best trade-off between effectiveness and efficiency. Therefore, the same techniques that have been used for entity linking in documents may not be suitable for queries. The overall research question driving our investigations is:

RQ3. How should entity linking in queries be performed to achieve high efficiency and effectiveness?

This general research question gives rise to two subquestions:

RQ3a. Given the response time requirements of an online setting, what is the relative importance of candidate entity ranking vs. disambiguation? In other words, if we are to allocate the available processing time between the two, which one would yield the highest gain?

RQ3b. Given the limited context provided by queries, which group of features is needed the most for effective entity disambiguation: contextual similarity, interdependence between entities, or both?

To answer the above research questions, we set up a framework where different candidate entity ranking and disambiguation methods can be plugged in. For each of these components, we experiment with both unsupervised and supervised alternatives, resulting in a total of four different ELQ systems. Supervised methods are expected to yield high effectiveness coupled with lower efficiency, while for unsupervised approaches it is the other way around. Our results reveal that it is more beneficial to use supervised learning for the candidate entity ranking step. If this step provides high-quality results, then disambiguation can be successfully tackled with a simple and elegant greedy algorithm. Moreover, our analysis shows that entity interdependencies provide little help for disambiguation. This is an interesting finding as it stands in contrast to the established postulation for entity linking in documents. Consequently, we identify a clearly preferred approach that uses supervised learning for candidate entity ranking

Table 5.1: Example queries with their linked entities. Each set represents an interpretation of the query; ambiguous queries have multiple interpretations (i.e., multiple table rows).

Query	Entity linking interpretation(s)
nashville thrift stores	{NASHVILLE TENNESSEE, CHARITY SHOP}
obama’s wife	{BARACK OBAMA}
cambridge population	{CAMBRIDGE}
	{CAMBRIDGE (MASSACHUSETTS)}
new york pizza manhattan	{NEW YORK-STYLE PIZZA, MANHATTAN}
	{NEW YORK, MANHATTAN}

and an unsupervised algorithm for disambiguation. Using the evaluation platform of the Entity Recognition and Disambiguation (ERD) challenge [43], we show that our preferred approach performs on a par with the current state of the art.

The main contribution of this chapter is to present the first systematic investigation of the ELQ task by bringing together the latest entity linking techniques and practices in a unified framework (Section 5.1). In addition, we develop a novel supervised approach for entity disambiguation in ELQ, which encompasses various textual and KB-based relatedness features. Finally, we make a best practice recommendation for ELQ and demonstrate that our recommended approach achieves state-of-the-art performance (Section 5.3). The resources developed with this paper are presented in Appendix A.3.

5.1 Entity Linking in Queries

As discussed in Chapter 3, the task of entity linking in queries (ELQ) is to identify, given an input query q , a set of *entity linking interpretations* $I = \{E_1, \dots, E_m\}$, where each interpretation $E_i = \{(m_1, e_1), \dots, (m_k, e_k)\}$ consists of a set of mention-entity pairs. Mentions within E_i are non-overlapping and each mention m_j is linked to an entity e_j in a reference knowledge base. By way of illustration, the output of ELQ for the query “new york pizza manhattan” would be $I = \{E_1, E_2\}$, where $E_1 = \{(\text{new york pizza}, \text{NEW YORK-STYLE PIZZA}), (\text{manhattan}, \text{MANHATTAN})\}$ and $E_2 = \{(\text{new york}, \text{NEW YORK}), (\text{manhattan}, \text{MANHATTAN})\}$. Following [43, 92], we restrict ourselves to detecting proper noun entities and do not link general concepts (e.g., “PIZZA”).

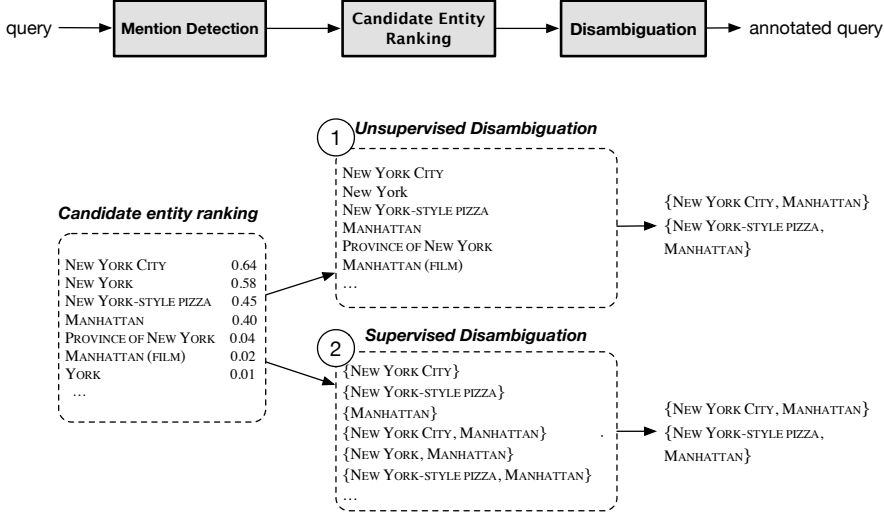


Figure 5.1: Pipeline for entity linking in queries with two alternatives for the disambiguation phase.

We frame the ELQ problem as a sequence of the following three subtasks (cf. Section 3.4): *mention detection*, *candidate entity ranking* (CER) and *disambiguation*. The first subtask takes the query q and identifies all mentions and their corresponding entities. In Chapter 3, we showed that using lexical matching of query n-grams against a rich dictionary of entity name variants allows for the identification of candidate entities with close to perfect recall (cf. Section 3.5). We, therefore, follow this approach in this chapter and focus only on the two other subtasks. The CER subtask takes all mention-entity pairs (from the mention detection step) and outputs a ranked list of mention-entity pairs, along with the corresponding scores. The last subtask takes this list as input and forms the set of entity linking interpretations I . For each of the latter two subtasks, we present two alternatives: unsupervised and supervised (see Figure 5.1). The resulting four possible combinations are compared experimentally in Section 5.3.1.

Table 5.2: Feature set used for ranking entities, categorized to mention (M), entity (E), mention-entity (ME), and query (Q) features.

Feature	Description	Type
$Len(m)$	Number of terms in the entity mention	M
$NTEM(m)^\ddagger$	Number of entities whose title equals the mention	M
$SMIL(m)^\ddagger$	Number of entities whose title equals part of the mention	M
$Matches(m)$	Number of entities whose surface form matches the mention	M
$Redirects(e)$	Number of redirect pages linking to the entity	E
$Links(e)$	Number of entity out-links in DBpedia	E
$Commonness(e, m)$	Likelihood of entity e being the target link of mention m	ME
$MCT(e, m)^\ddagger$	True if the mention contains the title of the entity	ME
$TCM(e, m)^\ddagger$	True if title of the entity contains the mention	ME
$TEM(e, m)^\ddagger$	True if title of the entity equals the mention	ME
$Pos_1(e, m)$	Position of the 1 st occurrence of the mention in entity abstract	ME
$SimM_f(e, m)^\ddagger$	Similarity between mention and field f of entity; Eq. 3.15	ME
$LenRatio(m, q)$	Mention to query length ratio: $\frac{ m }{ q }$	Q
$QCT(e, q)$	True if the query contains the title of the entity	Q
$TCQ(e, q)$	True if the title of entity contains the query	Q
$TEQ(e, q)$	True if the title of entity is equal query	Q
$Sim(e, q)$	Similarity between query and entity; Eq. 3.15	Q
$SimQ_f(e, q)^\ddagger$	LMS similarity between query and field f of entity; Eq. 3.15	Q

[‡] Entity title refers to the `rdfs:label` predicate of the entity in DBpedia

[†] Computed for all individual DBpedia fields $f \in \mathcal{F}$ and also for field *content* (cf. Section 5.2.1)

5.1.1 Candidate Entity Ranking

This subtask is responsible for ranking mention-entity pairs based on how likely the entities are link targets (in any interpretation of the query). The objective is to achieve both high recall and high precision at early ranks, as the top-ranked mention-entity pairs obtained here will be used directly in the subsequent disambiguation step. Formally, our focus of attention below is on ranking candidate (m, e) pairs with respect to the query, i.e., estimating $score(m, e, q)$.

Unsupervised

For the unsupervised ranking approach, we take the top performing generative model from Chapter 3: MLMcg. This model considers both the likelihood of the given mention and the similarity between the query and the entity: $score(m, e, q) = P(e|m)P(q|e)$, where $P(e|m)$ is the probability of a mention being linked to an entity (a.k.a. *commonness* [134]), computed from the FACC

collection [75]. The query likelihood $P(q|e)$ is estimated using the query length normalized language model similarity [113] (Eq. 3.15), where the entity and collection language models, $P(t|\theta_e)$ and $P(t|C)$, are computed using the Mixture of Language Models (MLM) approach [152] (Eq. 3.11).

Supervised

Our supervised approach employs learning to rank (LTR), where each (query, mention, entity) triple is described using a set of features. The ranking function is trained on a set of mention-entity pairs with binary labels, with positive labels denoting the correctly annotated entities for the given query. We use a total of 28 features (developed by ourselves or taken from the literature [136]), which are summarized in Table 5.2.

5.1.2 Disambiguation

The disambiguation step is concerned with the formation of entity linking interpretations $\{E_1, \dots, E_m\}$. Similar to the previous step, we examine both unsupervised and supervised alternatives.

Unsupervised

We employ the greedy algorithm introduced in Chapter 3, which forms interpretations in three consecutive steps (cf. Section 3.4.3): (i) pruning, (ii) containment mention filtering, and (iii) set generation. In the first step, the algorithm takes the ranked list of mention-entity pairs and discards the ones with ranking score below the threshold τ_s . This threshold is a free parameter that controls the balance between precision and recall. The second step removes containment mentions (e.g., “kansas city mo” vs. “kansas city”) by keeping only the highest scoring one. Finally, interpretations are built iteratively by processing mention-entity pairs in decreasing order of score and adding them to an existing interpretation E_i , where the mention does not overlap with other mentions already in E_i and i is minimal; if no such interpretation exists then a new interpretation $E_{|E|+1}$ is created.

Supervised

The overall idea is to generate all possible interpretations from a ranked list of mention-entity pairs, then employ a binary classifier to collectively select the most pertinent interpretations. Our approach takes the top- K mention-entity

Table 5.3: Feature set used in the supervised disambiguation approach. Type is either query dependent (QD) or query independent (QI).

Set-based Features		Type
$CommonLinks(E)$	Number of common links in DBpedia: $\bigcap_{e \in E} out(e)$.	QI
$TotalLinks(E)$	Number of distinct links in DBpedia: $\bigcup_{e \in E} out(e)$	QI
$J_{KB}(E)$	Jaccard similarity based on DBpedia: $\frac{CommonLinks(E)}{TotalLinks(E)}$	QI
$J_{corpora}(E)^\ddagger$	Jaccard similarity based on FACC: $\frac{ \bigcap_{e \in E} doc(e) }{ \bigcup_{e \in E} doc(e) }$	QI
$Rel_{MW}(E)^\ddagger$	Relatedness similarity [143] according to FACC	QI
$P(E)$	Co-occurrence probability based on FACC: $\frac{ \bigcap_{e \in E} doc(e) }{TotalDocs}$	QI
$H(E)$	Entropy of E : $-P(E)\log(P(E)) - (1-P(E))\log(1-P(E))$	QI
$Completeness(E)^\dagger$	Completeness of set E as a graph: $\frac{ edges(G_E) }{ edges(K_{ E }) }$	QI
$LenRatioSet(E, q)^\S$	Ratio of mentions length to the query length: $\frac{\sum_{e \in E} m_e }{ q }$	QD
$SetSim(E, q)$	Similarity between query and the entities in the set; Eq 5.1	QD
Entity-based Features		
$Links(e)$	Number of entity out-links in DBpedia	QI
$Commonness(e, m)$	Likelihood of entity e being the target link of mention m	QD
$Score(e, q)$	Entity ranking score, obtained from the CER step	QD
$iRank(e, q)$	Inverse of rank, obtained from the CER step: $\frac{1}{rank(e, q)}$	QD
$Sim(e, q)$	Similarity between query and the entity; Eq. 3.15	QD
$ContextSim(e, q)$	Contextual similarity between query and entity; Eq 5.3	QD

$^\ddagger doc(e)$ represents all documents that have a link to entity e

$^\dagger G_E$ is a DBpedia subgraph containing only entities from E ; and $K_{|E|}$ is a complete graph of $|E|$ vertices

$^\S m_e$ denotes the mention that corresponds to entity e

pairs (obtained from the CER step) and generates all possible interpretations out of those. We further require that mentions within the same interpretation do not overlap with each other. The value of K is set empirically, and it largely depends on the effectiveness of the CER step. If CER has high precision then K can be low, while less effective approaches can be compensated for with higher K values.

Once the candidate sets are generated, each is represented by a feature vector. We devise two main families of features: (i) set-based features are computed for the entire interpretation set, and (ii) entity-based features are calculated for individual entities. Features in the first group are computed collectively on all entities of the set and measured as a single value, while the members of the second group need to be aggregated (we use *min*, *max*, *avg* as aggregators). It is worth noting that each interpretation typically consists of few entities. Therefore, considering all entities for computing set-based features

is feasible; it also captures more information than one could get from aggregated pair-wise similarity features. Table 5.3 summarizes our feature set.

We highlight two novel and important features. $SetSim(E, q)$ measures the similarity between all entities in the interpretation E and the query q :

$$SetSim(E, q) = \frac{\prod_{t \in q} P(t|\theta_E)^{P(t|q)}}{\prod_{t \in q} P(t|C)^{P(t|q)}}. \quad (5.1)$$

It is calculated similar to Eq. 3.15, the main difference being that the probability of each term is estimated based on the interpretation’s language model:

$$P(t|\theta_E) = \sum_{e \in E} \sum_{f \in F} \mu_f P(t|\theta_{ef}). \quad (5.2)$$

In similar vein, $ContextSim(e, q)$ measures the similarity between the entity and the query context, where query context is the “rest” of the query, i.e., without the mention m_e that corresponds to entity e . Formally:

$$ContextSim(e, q) = P(q - m_e|e), \quad (5.3)$$

where $P(q - m_e|e)$ is computed using Eq. 3.15.

5.2 Experimental Setup

In this section, we describe our data sources, settings of methods, and evaluation measures.

5.2.1 Data

Knowledge base. We employ DBpedia 3.9 as our reference knowledge base and build an index of all entities that have both `rdfs:label` and `dbo:comment` predicates. The index includes the following set of fields: $\mathcal{F} = \{title, content, rdfs:label, dbo:wikiPageWikiLink, rdfs:comment, dbo:abstract\}$, where *title* is the concatenation of `rdfs:label`, `foaf:name` and `dbo:wikiPageRedirects` predicates, and *content* holds the content of all predicates of the entity; the remaining fields correspond to individual predicates.

Surface form dictionary. To recognize candidate entities in queries, we follow Chapter 3 and employ a rich surface form dictionary, which maps surface forms to entities. We utilize the FACC entity-annotated web corpora [75] and include surface forms above a commonness threshold of 0.1 [92]. Additionally, we add DBpedia name variants as surface forms; i.e., entity names from `rdfs:label`, `foaf:name`, and `dbo:wikiPageRedirects` predicates [56, 69, 92]. We confine our dictionary to entities present in the Freebase snapshot of proper named entities, provided by the ERD challenge [43].

Test collections. We evaluate our methods on two publicly available test collections: Y-ERD [92] and ERD-dev [43]. The former is based on the Yahoo Search Query Log to Entities (YSQLE) dataset¹ and consists of 2,398 queries. All results on this collection are obtained by performing 5-fold cross validation.² The ERD-dev collection contains 91 queries and is released as part of the ERD challenge [43]. We apply the trained models (on the whole Y-ERD collection) to ERD-dev queries and report on the results. In addition, ERD also provides an online evaluation platform which is based on a set of 500 queries (referred to as ERD-test); the corresponding annotations are not released. We evaluate the effectiveness³ of our recommended system using ERD-test to evaluate how it performs against the current state of the art.

5.2.2 Methods

Candidate entity ranking. For the unsupervised method (**MLMcg**), we follow [148] and use title and content fields, with weights 0.2 and 0.8, respectively. For the supervised method (**LTR**), we employ the Random Forest (RF) [36] ranking algorithm and set the number of trees to 1000 and the maximum features to 10% of size of the feature set [136]. We further include two baseline methods for reference comparison: (i) **MLM** is similar to **MLMcg**, but without considering the commonness score; i.e., computed based on Eq. 3.11; (ii) **CMNS** ranks entities based on the commonness score, while prioritizing longer mentions, and is shown to be a strong baseline [29, 92, 136].

¹<http://webscope.sandbox.yahoo.com/>

²It is important to note that Y-ERD contains queries that have been reformulated (often only slightly so) during the course of a search session; we ensure that queries from the same session are assigned to the same fold when using cross-validation.

³Carmel et al. [43] do not report on the efficiency of the approaches and the online leaderboard is no longer available, hence we present only effectiveness results from Cornolti et al. [53].

Disambiguation. The unsupervised disambiguation method (**GIF** algorithm) involves a score threshold parameter, which is set (using a parameter sweep) depending on the CER method used: 20 for MLMcg and 0.3 in case of LTR. For the supervised disambiguation method (**LTR**), we set the number of top ranked entities K to 5 (based on a parameter sweep) and use a RF classifier with similar setting to supervised CER. For baseline comparison, we consider the top-3 performing systems from the ERD challenge: SMAPH [53], NTUNLP [49], and Seznam [64]. We also compare our best performing approach with our implementation of TAGME [69], using the Wikipedia dump from June 16, 2015.

5.2.3 Evaluation

As both precision and recall matter for the candidate entity ranking step, we evaluate our methods using Mean Average Precision (MAP), recall at rank 5 ($R@5$), and precision at position 1 ($P@1$). When evaluating CER, we are only concerned about the ranking of entities; therefore, we consider each entity only once with its highest scoring mention: $score(e, q) = \arg \max_{m \in q} score(m, e, q)$. For the disambiguation step, we measure the end-to-end performance using set-based measures (precision, recall, and F-measure), according to the strict evaluation measures in Chapter 3, which is comparable to the official ERD results. We further report on the lenient evaluation measures from Section 3.2 for reference comparison. As for efficiency, we report on the average processing time for each query, measured in seconds. The experiments were conducted on a machine with an Intel Xeon E5 2.3GHz 12-core processor, running Ubuntu Linux v14.04. Statistical significance is tested using a two-tailed paired t-test. We mark improvements with $\triangle (p < 0.05)$ or $\blacktriangle (p < 0.01)$, deteriorations with $\nabla (p < 0.05)$ or $\blacktriangledown (p < 0.01)$, and no significance by \circ .

5.3 Results and Analysis

Next, we report on our experimental results and answer our research questions.

5.3.1 Results

We start by evaluating the *candidate entity ranking* and *disambiguation* steps and then answer our first research subquestion (RQ3a): “Given the response time requirements of an online setting, what is the relative importance of candidate entity ranking vs. disambiguation?”

Table 5.4: Candidate entity ranking results on the Y-ERD and ERD-dev datasets. Best scores for each measure are in boldface. Significance for line $i > 1$ is tested against lines $1..i - 1$.

Method	Y-ERD		
	MAP	R@5	P@1
MLM	0.7507	0.8556	0.6839
CMNS	0.7831 [▲]	0.8230 [▲]	0.7779 [▲]
MLMcg	0.8536 ^{▲▲}	0.8997 ^{▲▲}	0.8280 ^{▲▲}
LTR	0.8667^{▲▲▲}	0.9022^{▲▲°}	0.8479^{▲▲▲}

Method	ERD-dev		
	MAP	R@5	P@1
MLM	0.7675	0.8622	0.7333
CMNS	0.7037 [°]	0.7222 [▽]	0.7556 [°]
MLMcg	0.8543 ^{▲▲}	0.9015 ^{°▲}	0.8444^{°°}
LTR	0.8606^{▲▲°}	0.9289^{▲▲°}	0.8222 ^{°°°}

Candidate Entity Ranking

Table 5.4 presents the results for CER on the Y-ERD and ERD-dev datasets. We find that commonness is a strong performer (this is in line with the findings of Blanco et al. [29]). Combining commonness with MLM in a generative model (MLMcg) delivers excellent performance, with MAP above 0.85 and R@5 around 0.9. The LTR approach can bring in further slight, but for Y-ERD significant, improvements. This means that both of our CER methods (MLMcg and LTR) are able to find the vast majority of the relevant entities and return them at the top ranks.

Disambiguation

Table 5.5 reports on the disambiguation results. We use the naming convention X - Y , where X refers to the CER method (MLMcg or LTR) and Y refers to the disambiguation method (GIF or LTR) that is applied on top. Our observations are as follows. The MLM-GIF approach is clearly the most efficient but also the least effective one. Learning is more expensive for disambiguation than for CER, see LTR-GIF vs. MLMcg-LTR; yet, it is also clear from this com-

Table 5.5: End-to-end performance of ELQ systems on the Y-ERD and ERD-dev query sets. Significance for line $i > 1$ is tested against lines $1..i - 1$.

Method	Y-ERD			
	P_{strict}	R_{strict}	F_{strict}	Time (s)
MLMcg-GIF	0.709	0.709	0.709	0.058
MLMcg-LTR	0.725 [°]	0.724 [°]	0.724 [°]	0.893
LTR-LTR	0.731 ^{△°}	0.732 ^{△°}	0.731 ^{△°}	0.881
LTR-GIF	0.786^{▲▲▲}	0.787^{▲▲▲}	0.787^{▲▲▲}	0.382

Method	ERD-dev			
	P_{strict}	R_{strict}	F_{strict}	Time (s)
MLMcg-GIF	0.724	0.712	0.713	0.085
MLMcg-LTR	0.725 [°]	0.731 [°]	0.728 [°]	1.185
LTR-LTR	0.758 ^{°°}	0.748 ^{°°}	0.753 ^{°°}	1.185
LTR-GIF	0.852^{▲▲△}	0.828^{▲△°}	0.840^{▲▲△}	0.423

parison that more performance can be gained when learning is done for CER than when it is done for disambiguation. The most effective method is LTR-GIF, outperforming other approaches significantly on both test sets. It is also the second most efficient one. Interestingly, even though the MLMcg and LTR entity ranking methods perform equally well according to CER evaluation (cf. Table 5.4), we observe a large difference in their performance when the unsupervised disambiguation approach is applied on top of them. The reason is that the absolute scores produced by LTR are more meaningful than those of MLMcg (despite the query length normalization efforts for the latter; cf. Eq. 3.15). This plays a direct role in the GIF algorithm, where score thresholding is used. We note that the reported efficiency results are meant for comparison across different approaches. For practical applications, further optimizations to our basic implementation would be needed (cf. [29]).

Based on the results, LTR-GIF is our overall recommendation. We compare this method to the TAGME entity linking system in Table 5.6. The results show that this method significantly outperforms TAGME with respect to both strict and lenient evaluation measures. We also compare LTR-GIF to the top performers of the ERD challenge (using the official challenge platform); see Table 5.7. For this comparison, we additionally applied spell checking, as this

Table 5.6: ELQ results with respect to strict and lenient evaluation measures.

	Method	P _{strict}	R _{strict}	F _{strict}	P _{lenient}	R _{lenient}	F _{lenient}
<i>Y-ERD</i>	TAGME	0.665	0.664	0.664	0.682	0.685	0.681
	LTR-GIF	0.786	0.787	0.787	0.799	0.798	0.798
<i>ERD-dev</i>	TAGME	0.714	0.701	0.705	0.742	0.737	0.733
	LTR-GIF	0.852	0.828	0.840	0.852	0.828	0.840

Table 5.7: ELQ results on the official ERD test platform.

Method	F1
LTR-GIF	0.699
SMAPH-2 [53]	0.708
NTUNLP [49]	0.680
Seznam [64]	0.669

has also been handled in the top performing system (SMAPH-2) [53]. The results show that our LTR-GIF approach performs on a par with the state-of-the-art systems. This is remarkable taking into account the simplicity of the GIF algorithm vs. the considerably more complex solutions employed by others.

Here we turn into answering our first subquestion:

RQ3a. Given the response time requirements of an online setting, what is the relative importance of candidate entity ranking vs. disambiguation?

Our results reveal that candidate entity ranking is of higher importance than disambiguation for ELQ. Hence, it is more beneficial to perform the (expensive) supervised learning early on in the pipeline for the seemingly easier CER step; disambiguation can then be tackled successfully with a simple unsupervised algorithm (GIF). We note that selecting the top ranked entity does not yield an immediate solution; as shown in Chapter 3, disambiguation is an indispensable step in ELQ.

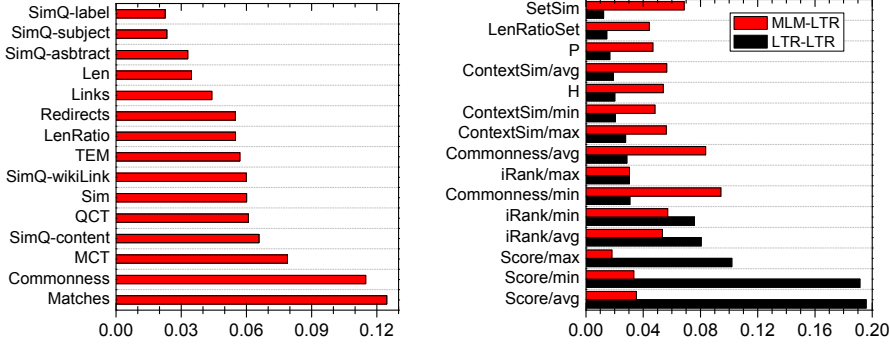


Figure 5.2: Most important features used in the supervised approaches, sorted by Gini score: (Left) Candidate entity ranking, (Right) Disambiguation.

5.3.2 Feature Analysis

To answer our second research subquestion (RQ3b), we analyze the features used in our supervised methods and report feature importance for both the CER and disambiguation steps. Figure 5.2(a) shows the top features used in the LTR entity ranking approach in terms of Gini score. We observe that *Matches*, *Commonness*, and the various query similarity features play the main role in the entity ranking function. As for the supervised disambiguation step, we selected the top 15 features independently for the MLMcg-LTR and LTR-LTR methods; interestingly, we ended up with the exact same set of features. Figure 5.2(b) demonstrates that nearly all influential features are query dependent; the only query independent features are *P* and *H*, capturing the co-occurrence of entities in a web corpus.

Based on the above results and analysis, we answer our second subquestion:

RQ3b. Given the limited context provided by queries, which group of features is needed the most for effective entity disambiguation?

Contextual similarity features are the most effective for entity disambiguation. This is based on two observations: (i) the unsupervised (GIF) method takes only the entity ranking scores as input, which are computed based on the contextual similarity between entity and query; (ii) the supervised (LTR) method relies the most on query-dependent features. This is an interesting finding, as it

stands in contrast to the common postulation in entity linking in documents that interdependence between entities help to better disambiguate entities. Entity interdependence features (and, in general, collective disambiguation methods) are more helpful when sufficiently many entities are mentioned in the text; this is not the case for queries.

5.4 Summary

In this chapter, we have performed the first systematic investigation of entity linking in queries (ELQ). We have presented a framework where different methods can be plugged in for two core components: *candidate entity ranking* and *disambiguation*. For each of these components, we have explored both unsupervised and supervised alternatives by employing and further extending state-of-the-art approaches. Our experiments have led to two important findings: (i) it is more rewarding to employ supervised learning for candidate entity ranking than for disambiguation, and (ii) entity interdependence features, which are the essence of collective disambiguation methods, have little benefit for ELQ. We further demonstrate that our recommended method achieves state-of-the-art performance. Overall, our findings have not only revealed important insights, but also provide guidance as to where future research and development in ELQ should be focused.

Chapter 6

Exploiting Entity Linking in Queries for Entity Retrieval

In the previous three chapters (Chapters 3–5) we have focused on understanding queries by annotating them with entities. In this chapter, we use these annotations to help answer the queries. The task of answering queries with entities, referred to as *entity retrieval*, has been extensively addressed over the past years [27, 111, 147, 148, 157, 210]. A common approach is to match queries against the term-based representation of entities. Here, we focus on improving entity retrieval performance by employing entity linking in queries.

Both *entity linking* and *entity retrieval* tasks represent key building blocks for semantic search [137] and are typically backed by a large-scale knowledge base. Despite this common ground, the two tasks have so far been studied mostly on their own, as standalone problems. We say mostly, as entity ranking, to a limited extent, has already been exploited in entity linking: most entity linking approaches involve an entity retrieval component for collecting candidate entities for a given text segment, see, e.g., [29, 69, 92]. The other direction, utilizing entity linking for entity retrieval, to the best of our knowledge, has not been explored yet. This chapter presents the first attempt at bridging this gap by performing entity linking on search queries and using the resulting annotations to improve entity retrieval.

It has been shown in prior work that entity retrieval can be improved by leveraging semantic annotations of the query, such as target entity types or related entities, see, e.g., [15, 37, 108, 162]. These studies, using the TREC

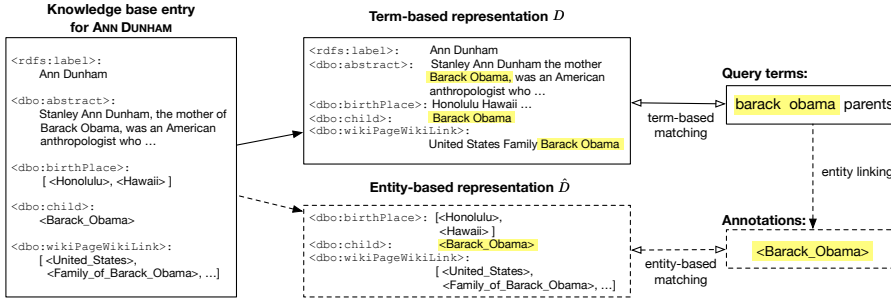


Figure 6.1: Demonstration of term- and entity-based representation of entities. The query terms match against the term-based representation of entity ANN DUNHAM, while the entity annotations of the query match against the entity-based representation. The dashed parts indicate the novel elements of our work.

Entity [14] and INEX-XER [59] benchmarking platforms, assume that semantic annotations (e.g., input entity, target type, or entity relations) are provided as part of the definition of the information need, i.e., complement the keyword query. In this chapter, we obtain entity annotations automatically, and consider both term- and entity-based representations of entities for general-purpose entity retrieval; see Figure 6.1.

It is worth relating our efforts to the large body of prior work that has shown that leveraging information about entity annotations of queries can improve document retrieval performance [34, 58, 121, 126, 197, 198]. Importantly, this task is very different from ours: we search for entities in a (manually curated) knowledge base where entities are first-class citizens. This stands in contrast with document retrieval, where the entity annotations are a result of some automated process, which always involves a degree of uncertainty. Not only the task, but the techniques used for utilizing entity annotations are also different; in document retrieval, entities are typically used for query expansion or as simple features in a learning to rank framework (see Section 2.2). We, on the other hand, represent and match entities directly as a separate component in the retrieval model. Against this background, the main research question driving this chapter is:

RQ4. How to exploit entity annotations of queries in entity retrieval?

To address this question, we introduce a general framework for leveraging entity annotations of queries into term-based models. Our framework is based on the Markov Random Field (MRF) model [139]. Within this framework, we introduce a new component for matching the linked entities from the query. This component, termed ELR (for “Entity Linking incorporated Retrieval”), may be seen as an extension that can be applied on top of any text-based retrieval model that can be instantiated as a MRF model. Entity matches are facilitated by an additional entity-based representation that preserves entity relationships as recorded in the knowledge base; see the block denoted with \tilde{D} in Figure 6.1. We address a number of technical and modeling issues that stem from the differences between terms and entities, including: (i) the sparseness of entity-based representations compared to term-based ones, (ii) the varying number of entity annotations per query, (iii) dealing with uncertainties involved with entity linking, and (iv) the mapping of entities to fields. Specifically, we seek to answer the following subquestions:

RQ4a: Can entity retrieval performance be improved by incorporating entity annotations of the query? (Section 6.4.1)

RQ4b: How are the different types of queries impacted by ELR? (Section 6.4.2)

RQ4c: How robust is our method with respect to parameter settings? (Section 6.4.3)

RQ4d: What is the impact of the entity linking component on end-to-end performance? (Section 6.4.4)

We conduct experiments on a test collection consisting of close to 500 heterogeneous queries, ranging from short keyword queries to natural language questions, and show that our model can consistently and significantly improve upon standard language models [206], semi-structured [111], and term-dependence models [139, 210], and outperforms the current state-of-the-art on ad hoc entity retrieval by over 6% in terms of MAP. In addition, we demonstrate its robustness against parameter setting and entity linking configuration. The resources developed within this paper are presented in Appendix A.4.

The remainder of this chapter is structured as follows. In Section 6.1, we present the MRF model as the basis of our retrieval framework. We then introduce our new entity linking integrated retrieval framework in Section 6.2. Sections 6.3 and 6.4 present our experimental setup and results obtained by applying our model on standard test collections. We end with a summary in Section 6.5.

6.1 Background

In this section, we describe the Markov Random Field (MRF) model [139], which is the basis of our proposed approach (following in Section 6.2). We further discuss two specific variations of the MRF model: the Sequential Dependence Model [139] in Section 6.1.2 and the Fielded Sequential Dependence Model [210] in Section 6.1.3.

6.1.1 The Markov Random Field Model

Markov Random Field models for information retrieval were first introduced by Metzler and Croft [139] to capture the dependencies between query terms. Given a document D and a query Q , the goal of these models is to compute the joint probability $P(Q, D)$:

$$P(D|Q) = \frac{P(Q, D)}{P(Q)} \stackrel{rank}{=} P(Q, D). \quad (6.1)$$

This probability is estimated based on a Markov Random Field, which is a common modeling choice for computing the joint probabilities of random variables. For document retrieval, a MRF is defined by a graph G with nodes consisting of query terms q_i and the document D , and edges representing the dependence between the nodes. The joint probability over variables of the graph G is computed as:

$$P_{\Lambda}(Q, D) = \frac{1}{Z_{\Lambda}} \prod_{c \in C(G)} \psi(c; \Lambda), \quad (6.2)$$

where $C(G)$ is the set of cliques in G and $\psi(c; \Lambda) = \exp[\lambda_c f(c)]$ is a non-negative *potential function*, parameterized by the weight λ_c and the *feature function* $f(c)$. The parameter Z_{Λ} serves as a normalization factor, which is generally ignored due to computational infeasibility. Substituting all these elements into Eq. 6.1, the final ranking function becomes:

$$P(D|Q) \stackrel{rank}{=} \sum_{c \in C(G)} \lambda_c f(c). \quad (6.3)$$

This ranking function provides a solid theoretical basis for a wide spectrum of retrieval models: from traditional unigram-based models to more sophisticated ones involving n-grams as well as additional task- or domain-specific features [21, 162]. To build a ranking function, all one needs to do is to define the graph structure and the potential functions over the graph cliques.

6.1.2 Sequential Dependence Model

The Sequential Dependence Model (SDM) is a popular MRF-based retrieval model, which provides a good balance between retrieval effectiveness and efficiency [139]. In the underlying graph of this model, only adjacent query terms are connected to each other, meaning that query terms are sequentially dependent on each other; i.e., the white nodes in Figure 6.2. Under this assumption, the potential functions are defined for two types of cliques: (i) 2-cliques involving a query term and the document, (ii) cliques containing two contiguous terms and the document. The potential function for the first type of cliques is:

$$\psi(q_i, D; \Lambda) = \exp[\lambda_T f_T(q_i, D)], \quad (6.4)$$

where $f_T(q_i, D)$ is the feature function for the query term q_i and the document D . There are two possibilities for the second type of cliques (two terms): either the terms occur contiguously in the query or they do not. These two cases make up the potential functions for *ordered* and *unordered* matches, and are denoted by the O and U subscripts, respectively:

$$\psi(q_i, q_{i+1}, D; \Lambda) = \exp[\lambda_O f_O(q_i, q_{i+1}, D) + \lambda_U f_U(q_i, q_{i+1}, D)]. \quad (6.5)$$

When substituting the two potential functions $\psi(q_i, D; \Lambda)$ in Eq. 6.4 and $\psi(q_i, q_{i+1}, D; \Lambda)$ in Eq. 6.5 into Eq. 6.3, and factoring out the λ parameters, the SDM ranking function becomes:

$$\begin{aligned} P(D|Q) \stackrel{rank}{=} & \lambda_T \sum_{q_i \in Q} f_T(q_i, D) + \\ & \lambda_O \sum_{q_i, q_{i+1} \in Q} f_O(q_i, q_{i+1}, D) + \\ & \lambda_U \sum_{q_i, q_{i+1} \in Q} f_U(q_i, q_{i+1}, D), \end{aligned} \quad (6.6)$$

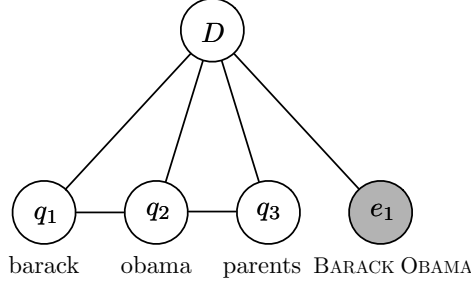


Figure 6.2: Graphical representation of the ELR model for the query “barack obama parents.” Here, all query terms are sequentially dependent and the phrase “barack obama” is linked to the entity BARACK OBAMA.

where the parameters should meet the constraint of $\lambda_T + \lambda_O + \lambda_U = 1$. The specific feature functions are set as follows:

$$f_T(q_i, D) = \log \left[\frac{tf_{q_i, D} + \mu \frac{cf_{q_i}}{|C|}}{|D| + \mu} \right] \quad (6.7)$$

$$f_O(q_i, q_{i+1}, D) = \log \left[\frac{tf_{\#1(q_i, q_{i+1}), D} + \mu \frac{cf_{\#1(q_i, q_{i+1})}}{|C|}}{|D| + \mu} \right] \quad (6.8)$$

$$f_U(q_i, q_{i+1}, D) = \log \left[\frac{tf_{\#uwN(q_i, q_{i+1}), D} + \mu \frac{cf_{\#uwN(q_i, q_{i+1})}}{|C|}}{|D| + \mu} \right], \quad (6.9)$$

where tf_D is the frequency of the term(s) in the document D and cf denotes the total number of occurrences of the term(s) in the entire collection. The function $\#1(q_i, q_{i+1})$ searches for the exact match of the phrase q_i, q_{i+1} , while $\#uwN(q_i, q_{i+1})$ counts the co-occurrence of terms within a window of N words (where N is set to 8 based on [139]). The parameter μ is the Dirichlet prior, which is taken to be the average document length in the collection.

Putting all these together, SDM is basically the weighted sum of language model scores obtained from three sources: (i) query terms, (ii) exact match of query bigrams, and (iii) unordered match of query bigrams. Our approach in Section 6.2 employs the same sequential dependence assumption that SDM does.

6.1.3 Fielded Sequential Dependence Model

The Fielded Sequential Dependence Model (FSDM) [210] extends the SDM model to support structured document retrieval. In essence, FSDM replaces the document language model of feature functions (Eqs. 6.7, 6.8, and 6.9) with those of the Mixture of Language Models (MLM) [152]. Given a fielded representation of a document (e.g., title, body, anchors, metadata, etc. in the context of web document retrieval), MLM computes a language model probability for each field and then takes a linear combination of these field-level models. Hence, FSDM assumes that a separate language model is built for each document field and then computes the feature functions based on fields $f \in \mathcal{F}$, with \mathcal{F} being the universe of fields. For individual terms, the feature function becomes:

$$f_T(q_i, D) = \log \sum_f w_f^T \frac{tf_{q_i, D_f} + \mu_f \frac{cf_{q_i, f}}{|C_f|}}{|D_f| + \mu_f}, \quad (6.10)$$

while ordered and unordered bigrams are estimated as:

$$f_O(q_i, q_{i+1}, D) = \log \sum_f w_f^O \frac{tf_{\#1(q_i, q_{i+1}), D_f} + \mu_f \frac{cf_{\#1(q_i, q_{i+1}), f}}{|C_f|}}{|D_f| + \mu_f}, \quad (6.11)$$

$$f_U(q_i, q_{i+1}, D) = \log \sum_f w_f^U \frac{tf_{\#uN(q_i, q_{i+1}), D_f} + \mu_f \frac{cf_{\#uN(q_i, q_{i+1}), f}}{|C_f|}}{|D_f| + \mu_f}. \quad (6.12)$$

The parameters w_f are the weights for each field, which are set to be non-negative with the constraint $\sum_f w_f = 1$. Zhiltsov et al. [210] trained both the field weights (w_f) and the feature function weights ($\lambda_T, \lambda_O, \lambda_U$ in Eq. 6.6) in two stages using the Coordinate Ascent algorithm [140].

6.2 The ELR Approach

This section presents our approach for incorporating entity linking into entity retrieval. We start by introducing our general MRF-based framework in Section 6.2.1, and continue with describing the feature functions in Section 6.2.2 and fielded representation of entities in Section 6.2.3.

6.2.1 Model

Our *Entity Linking incorporated Retrieval* (ELR) approach is an extension of the MRF framework for incorporating entity annotations into the retrieval model. We note, without detailed elaboration, that ELR is applicable to a wide range of retrieval problems where documents, or document-based representations of objects, are to be ranked, and entity annotations are available to be leveraged in the matching of documents and queries. Our main focus in this paper, however, is limited to entity retrieval; entity annotations are an integral part of the representation here, cf. Figure 6.1. (This is unlike traditional document retrieval, where documents would need to be annotated by an automated process that is prone to errors.) To show the generic nature of our approach, and also for the sake of notational consistency with the previous section, we shall refer to documents throughout this section. We detail how these documents are constructed for our particular task, entity retrieval, in Section 6.2.3.

Our interest in this work lies in incorporating entity annotations and not in creating them. Therefore, entity annotations of the query are assumed to have been generated by an external entity linking process, which we treat much like a black box. Formally, given an input query $Q = q_1 \dots q_n$, the set of linked entities is denoted by $E(Q) = \{e_1, \dots, e_m\}$. We do not impose any restrictions on these annotations, i.e., they may be overlapping and a given query span might be linked to multiple entities. It might also be that $E(Q)$ is an empty set. Further, we assume that annotations have confidence scores associated with them. For each entity $e \in E(Q)$, let $s(e)$ denote the confidence score of e , with the constraint of $\sum_{e \in E(Q)} s(e) = 1$.

The graph underlying our model consists of document, term, and entity nodes. As shown in Figure 6.2, we assume that the query terms are sequentially dependent on each other, while the annotated entities are independent of each other and of the query terms. Based on this assumption, the potential functions are computed for three types of cliques: (i) 2-cliques consisting of edges between the document and a term node, (ii) 3-cliques consisting of the document and two term nodes, and (iii) 2-cliques consisting of edges between the document and an entity node. The potential functions for the first two types are identical to the SDM model (Eqs. 6.4 and 6.5). We define the potential function for the third clique type as:

$$\psi_E(e, D; \Lambda) = \exp[\lambda_E f_E(e, D)], \quad (6.13)$$

where λ_E is a free parameter and $f_E(e, D)$ is the feature function for the entity e and document D (to be defined in Section 6.2.2). By substituting all feature functions into Eq. 6.3, the MRF ranking function becomes:

$$\begin{aligned}
 P(D|Q) \stackrel{rank}{=} & \sum_{q_i \in Q} \lambda_T f_T(q_i, D) + \\
 & \sum_{q_i, q_{i+1} \in Q} \lambda_O f_O(q_i, q_{i+1}, D) + \\
 & \sum_{q_i, q_{i+1} \in Q} \lambda_U f_U(q_i, q_{i+1}, D) + \\
 & \sum_{e \in E(Q)} \lambda_E f_E(e, D). \tag{6.14}
 \end{aligned}$$

This model introduces an additional parameter for weighing the importance of entity annotations, λ_E , on top of the three parameters ($\lambda_{\{T,O,U\}}$) from the SDM model (cf. Section 6.1.2). There is a crucial difference between entity-based and term-based matches with regards to the λ parameters. The number of cliques for term-based matches is proportional to the length of the query ($|Q|$ for unigrams and $|Q| - 1$ for ordered and unordered bigrams), which makes them compatible (directly comparable) with each other, irrespective of the length of the query. Therefore, in SDM, $\lambda_{\{T,O,U\}}$ are taken out of the summations (cf. Eq. 6.6) and can be trained without having to worry about query length normalization. The parameter λ_E , however, cannot be treated the same manner for two reasons. Firstly, the number of annotated entities for each query varies and it is independent of the length of the query. For example, a long natural language query might be annotated with a single entity, while shorter queries are often linked to several entities, due to their ambiguity. Secondly, we need to deal with varying levels of uncertainty that are involved with entity annotations of the query. The confidence scores associated with the annotations, which are generated by the entity linking process, should be integrated into the retrieval model.

To address the above issues, we rewrite the λ parameters as a parameterized function over each clique and define them as:

$$\lambda_T(q_i) = \lambda_T \frac{1}{|Q|}, \quad (6.15)$$

$$\lambda_O(q_i, q_{i+1}) = \lambda_O \frac{1}{|Q| - 1}, \quad (6.16)$$

$$\lambda_U(q_i, q_{i+1}) = \lambda_U \frac{1}{|Q| - 1}, \quad (6.17)$$

$$\lambda_E(e) = \lambda_E s(e), \quad (6.18)$$

where $|Q|$ is the query length and $s(e)$ is the confidence score of entity e obtained from the entity linking step. Considering this parametric form of the λ parameters, our final ranking function takes the following form:

$$\begin{aligned} P(D|Q) \stackrel{rank}{=} & \lambda_T \sum_{q_i \in Q} \frac{1}{|Q|} f_T(q_i, D) + \\ & \lambda_O \sum_{q_i, q_{i+1} \in Q} \frac{1}{|Q| - 1} f_O(q_i, q_{i+1}, D) + \\ & \lambda_U \sum_{q_i, q_{i+1} \in Q} \frac{1}{|Q| - 1} f_U(q_i, q_{i+1}, D) + \\ & \lambda_E \sum_{e \in E(Q)} s(e) f_E(e, D), \end{aligned} \quad (6.19)$$

where the free parameters λ are placed under the constraint of $\lambda_T + \lambda_O + \lambda_U + \lambda_E = 1$. This model ensures that the scores for the different type of matches (i.e., term, ordered window, unordered window, and entities) are normalized and the λ parameters, which are to be trained, are not influenced by the length of the query or by the number of linked entities. In addition, it provides us with a general ranking framework that can encompass various retrieval models. If the λ_O and λ_U parameters are set to zero, the model is an extension of unigram based models, such as LM and MLM. Otherwise, it extends SDM and FSDM. We also note that due to the normalizations applied to the different set of matches, the *full dependence* variant of MRF model [139] could also be instantiated in our framework; this, however, is outside the scope of this study.

6.2.2 Feature Functions

Feature functions form an essential part of MRF-based models. We now discuss the estimation of these for the ELR model. For all feature functions, we use a fielded document representation of entities, as it is a common and effective approach for entity retrieval, see, e.g., [12, 27, 78, 147, 210]. The first three feature functions in Eq. 6.19, f_T , f_O , and f_U , are computed as defined in Eqs. 6.10, 6.11, and 6.12, respectively.

Let us then turn to defining the function $f_E(e, D)$ in Eq. 6.19, which is a novel feature introduced by our ELR model. This function measures the goodness of the match between an entity e linked in the query and a document D . These matches are facilitated by an entity-based representation of documents. For each document D , an entity-based representation \hat{D} is obtained by ignoring document terms and considering only entities. In the context of our work, the entity represented by document D stands in typed relationships with a number of other entities, as specified in the knowledge base. The various relationships are modeled as fields in the document. Consider the example in Figure 6.1, where the document represents the entity ANN DUNHAM, who is being linked to the entity BARACK OBAMA (via the relationship `<dbo:child>`). This entity-based representation differs from the traditional term-based representation in at least two important ways. Firstly, each entity appears at most once in each document field. Secondly, if an entity appears in a field, then it should be considered a match, irrespective of what other entities may appear in that field. Consider again the example in Figure 6.1, where the field `<dbo:birthplace>` has multiple values, HONOLULU and HAWAII. Then, if either of these entities is linked in the query, that should account for a perfect match against this particular field, irrespective of how many other locations are present in that field. Motivated by these observations, we define the feature function f_E as:

$$f_E(e, D) = \log \sum_{f \in \mathcal{F}} w_f^E \left[(1 - \alpha) t f_{\{0,1\}(e, \hat{D}_f)} + \alpha \frac{df_{e,f}}{df_f} \right], \quad (6.20)$$

where the linear interpolation implements the Jelinek-Mercer smoothing method, with α set to 0.1, and $t f_{\{0,1\}(e, \hat{D}_f)}$ indicates whether the entity e is present in the document field \hat{D}_f or not. For the background model, we employ the notion of document frequency as follows: $df_{e,f} = |\{\hat{D} | e \in \hat{D}_f\}|$ is the total number of documents that contain the entity e in field f and $df(f) = |\{\hat{D} | \hat{D}_f \neq \emptyset\}|$ is the number of documents with a non-empty field f .

Table 6.1: Selected fields with the corresponding mapping probabilities for the term “finland” and the entity FINLAND.

“finland”		Finland	
Field name	Prob.	Field name	Prob.
<dct:subject>	0.210	<dbo:country>	0.223
<dbo:wikiPageWikiLink>	0.178	<dbo:wikiPageWikiLink>	0.201
types	0.168	contents	0.189
contents	0.113	<dbo:birthPlace>	0.170
<rdfs:comment>	0.089	<dbo:hometown>	0.053
<dbo:abstract>	0.070	<dbo:location>	0.047
<rdfs:label>	0.069	<dbo:nationality>	0.041
names	0.059	<dbo:deathPlace>	0.034
<foaf:isPrimaryTopicOf>	0.040	<dbo:locationCountry>	0.028
yago:<rdf:type>	0.001	<dbo:ground>	0.013

All feature functions (f_T , f_O , f_U , and f_E) involve free parameters w_f , which control the field weights. Zhiltsov et al. [210] set these type of parameters using a learning algorithm, which leads to a large number of parameters to be trained (the number of feature functions times the number of fields). Instead, we employ a parameter-free estimation of field weights, using field mapping probabilities introduced in the Probabilistic Retrieval Model for Semistructured Data (PRMS) [111]. This probability infers the importance of each field, with respect to a given query term, based on collection statistics of that term. Specifically, the probability of a field f , from the universe of fields \mathcal{F} , is computed with respect to a given term t as follows:

$$P(f|t) = \frac{P(t|f)P(f)}{\sum_{f' \in \mathcal{F}} P(t|f')P(f')}. \quad (6.21)$$

Here, $P(f)$ is the prior probability of field f , which is set proportional to the frequency of the field (across all documents in the collection), and $P(t|f)$ is estimated by dividing the number of occurrences of term t in field f by the sum of term counts in f across the whole collection. We compute the probability $P(f|t)$ for all query terms, ordered and unordered bigrams, and use the resulting values for the weights w_f^T , w_f^O , and w_f^U (used in Eqs. 6.10-6.12), respectively. The weights w_f^E (used in Eq. 6.20) are also estimated using Eq. 6.21, but this

time we compute this probability for entities instead of terms (i.e., t is replaced with e).

Employing the mapping probability $P(f|\cdot)$ instead of free parameters w_f [210] has three advantages. First, the field mapping probability specifies field importance for each query term (or bigram) individually, while the w_f parameters are the same for all query terms (or bigrams). Second, the number of free parameters in the feature functions f_T , f_O , f_U , and f_E reduces from $4 * |\mathcal{F}|$ to zero. Hence, the final model is more robust and can be employed in various settings, without risking overfitting. Lastly and most importantly, estimating the field weights this way allows us to have a query-specific selection of fields, depending on the linked entity, as opposed to having pre-trained (fixed) field weights.

6.2.3 Fielded Representation of Entities

We now detail how the term-based and entity-based representation are obtained for entities, from the knowledge base entry (i.e., subject-predicate-object triples) describing the entity. One of the challenges of working with a fielded document-based representation of entities is the appropriate selection of fields. While grouping SPO triples by predicates and mapping each predicate to a separate document field is straightforward, retrieval can become highly inefficient because of the large number of fields [147]. Previous work has suggested a number of solutions to alleviate this problem by reducing the number of fields, which can be summarized under two main categories: (i) selecting a subset of fields that are considered and (ii) grouping fields together into a handful of predefined categories. When using the first approach, predicates are commonly ordered by frequency and a rank-based cutoff is applied, e.g., top 1000 in [12]. There are two choices for assigning the field weights in this setting: to simply use uniform values for all fields or to employ some estimation technique (such as the field mapping probabilities in the PRMS model) as training is generally infeasible due to the large number of fields. Examples of the second technique, referred to as “predicate folding” in [147], include grouping fields into a handful of predetermined categories based on type [147, 210] or manually determined importance [27]. It has been shown in [148] that it is possible to achieve solid performance even with as few as two fields, “title” and “content.” One main advantage of predicate folding is that the estimation of field weights becomes tractable. The disadvantage is that a large part of the semantics associated with the individual predicates is discarded.

In this work, we combine these two strategies to get the best of both approaches. We employ predicate folding for three designated fields: names, types,

and content (see Section 6.3.1). In addition, we consider all fields, which are not included in names or types, on their own. From this combined set, we then select the top-N most frequent fields across the whole knowledge base and use them for the term-based entity representation.

The entity-based representation requires a different field selection procedure from the above, as entities (SPO triples with an URI value as object) occur less often and follow an entirely different pattern than terms. For instance, the entity FINLAND mostly occurs in the `<dbo:country>` and `<dbo:birthPlace>` fields, while the entity ANCIENT ROMAN ARCHITECTURE often appears in the `<dbo:architecturalStyle>` field. This illustrates that it is not desirable to have the same (and fixed) set of fields for all entities, but field selection should be performed on an entity-specific manner. Therefore, for each entity, we select the top-N fields, from the entity-based representations, that the entity occurs in. As this computation can be performed offline, it does not impact negatively the efficiency of retrieval. Table 6.1 shows an excerpt of the mapping probability distribution for a given term and entity.

6.3 Experimental Setup

This section presents our experimental setup, including the data set (Section 6.3.1), field selection (Section 6.3.2), parameter settings (Section 6.3.4), and how entity linking is performed (Section 6.3.5).

6.3.1 Data

We use DBpedia version 3.9 as our knowledge base along with the DBpedia-Entity test collection [12].

Indices. For our experiments, we created two fielded indices from subject-predicate-object triples: a *term-based index*, where all entities (URI objects) are resolved to terms, and an *entity-based index*, where only URI objects are kept. The former index is used to compute the unigram and bigram term probabilities (Eqs. 6.10-6.12), while the latter is employed for the entity probability computations (Eq. 6.20). We built the indices using Lucene and made use of SpanNearQuery to get the statistics for ordered and unordered phrases. Our indices are confined to entities having a name and a short abstract (i.e. fields `<rdfs:label>` and `<rdfs:comment>`), resulting in a total of 3,984,580 entities. They contain the top-1000 most frequent DBpedia predicates as fields, together

Table 6.2: Query subsets of the DBpedia-Entity test collection.

Query subset	#queries	avg(q)	#rel
SemSearch ES	130	2.7	1115
ListSearch	115	5.6	2390
INEX_LD	100	4.8	3680
QALD-2	140	7.9	5773
Total	485	5.3	12958

with three other fields: (i) the **names** field, which is the constitution of entity predicates `<rdfs:label>`, `<foaf:name>`, and redirected entities; (ii) the **types** field, which contains `<rdf:type>` and attribute names ending in “subject”; (iii) the **contents** field, which holds the contents of all entity fields except entity links in other languages (`<owl:sameAs>`). In the term-based index, terms are lowercased and stopped using the default Lucene stopwords list, and all URIs are replaced with the name of the corresponding entity. In the entity-based index, only URIs are indexed and all literal objects are ignored. In addition, the URI of each entity itself is also added to the **contents** field in the entity-based index.

Queries. We evaluate the effectiveness of our models using the DBpedia-Entity collection [12], which comprises 485 queries from a number of entity retrieval benchmarking campaigns. Following [21], queries are stopped using a handful of stop patterns (“which,” “who,” “what,” “where,” “give me,” “show me”) to improve entity linking and initial retrieval performance. We perform stopwords removal after the entity linking step. We break down retrieval results according to the four categories suggested by Zhiltsov et al. [210]:

- **SemSearch ES:** Keyword queries targeting specific entities, which are often ambiguous (e.g., “madrid,” “hugh downs”).
- **ListSearch:** Combination of INEX-XER, SemSearch LS, and TREC Entity queries, targeting a list of entities that match a certain criteria (e.g., “Airports in Germany,” “the first 13 american states”).
- **INEX_LD:** General keyword queries, involving a mixture of names, types, relations, and attributes (e.g., “Eiffel,” “vietnam war movie,” “gallo roman architecture in paris”).

- **QALD-2**: Natural language queries (e.g., “which country does the creator of miffy come from,” “give me all female russian astronauts”).

Table 6.2 provides descriptive statistics on these query subsets.

6.3.2 Field Selection

The number of fields used in the term-based representation is a parameter shared by all but two of the evaluated models (single-field LM and SDM). We examined retrieval performance against a varying number of fields ($n = 10^i, i = 0, 1, 2, 3$), see Figure 6.3. Note that fields are ordered by frequency. It is clear from the figure that the best results are obtained when the top 10 most frequent fields are used. We used this setting in all our experiments, unless stated otherwise. For consistency, we also used the same setting, i.e., top 10 fields, for the entity-based representation.

6.3.3 Baseline Models

We compare our model with a number of baseline models. The first four (LM, MLM-tc, MLM-all, and PRMS) are language modeling-based methods that were introduced as standard baselines for the DBpedia-Entity test collection [12]. The other two (SDM and FSDM) are taken from [210], the work that reported the best results on this collection so far. Specifically, the baseline models considered are:

- **LM**: The standard language modeling approach [206], against the **contents** field.
- **MLM-tc**: The Mixture of Language Models [152] with two fields, **names** and **contents**, with weights 0.2 and 0.8, respectively, as suggested in [148].
- **MLM-all**: The Mixture of Language Models using the top 10 fields with equal weights.
- **PRMS**: The Probabilistic Retrieval Model for Semistructured Data [111], using the top 10 fields.
- **SDM**: The Sequential Dependence Model [139], against the **contents** field.
- **FSDM***: The Fielded Sequential Dependence Model [210] on the top 10 fields, with field weights estimated using PRMS.

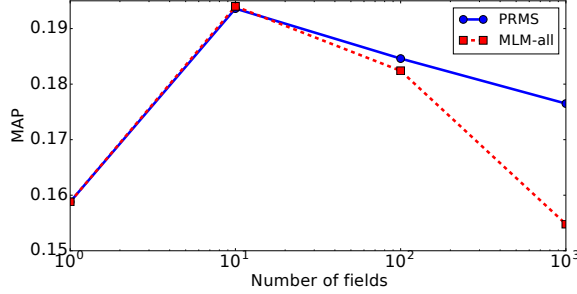


Figure 6.3: Effect of varying number of fields on MAP.

6.3.4 Parameter Setting

This section describes the parameter settings used in our experiments. The Dirichlet prior μ in language models is set to the average document/field length across the collection. The unordered window size N in Eqs. 6.9 and 6.12 is chosen to be 8, as suggested in [139, 210]. To estimate the λ parameters involved in SDM, FSDM, and our approach, we employ the Coordinate Ascent (CA) algorithm [140] and directly optimize Mean Average Precision (MAP). CA is a commonly used optimization technique, which iteratively optimizes a single parameter while holding all other parameters fixed. We make use of the CA implementation provided in the RankLib framework and set the number of random restarts to 3. Following [210], we estimate the λ parameters of SDM, FSDM, and ELR-based approaches using 5-fold cross validation for each of the 4 query subsets separately. We note that Zhiltsov et al. [210] train both the λ and w parameters (Eq. 6.6-6.12) for the FSDM model. As we use different entity representation from [210] (with 10 as opposed to 5 fields), training parameters in this manner would result in cross-validation of 33 parameters for each query subset, which would be prone to overfitting. We avoid this issue by employing the PRMS field mapping probability for field weights w (i.e., Eq. 6.21). Therefore, our implementation of FSDM slightly deviates from the original paper [210]; in acknowledgement of this distinction, we will refer to our implementation as FSDM*.

For all experiments, we employ a two stage retrieval method: first an initial set of top 1000 results is retrieved using Lucene’s default search settings, then this set is re-ranked with the specific retrieval model (using an in-house imple-

mentation). Evaluation scores are reported on the top 100 results. To perform cross-validation, we randomly create train and test folds from the initial result set, and use the same folds throughout all the experiments. To measure statistical significance we employ a two-tailed paired t-test and denote differences at the 0.01 and 0.05 levels using the \blacktriangle and \triangle symbols, respectively.

6.3.5 Entity Linking

Entity linking is a key component of the ELR approach. For the purpose of reproducibility, all the entity annotations in this work are obtained using an open source entity linker, TAGME [69], accessed through its RESTful API.¹ TAGME is one of the best performing entity linkers for short queries [43, 51]. As suggested in the API documentation, we use the default threshold 0.1 in our experiments; we analyze the effect of the threshold parameter in Section 6.4.4.

6.4 Results and Analysis

In this section, we present a series of experiments conducted to answer our research questions.

6.4.1 Overall Performance

To find out whether entity linking in queries can improve entity retrieval performance (**RQ4a**), we compare a number of entity retrieval approaches proposed in the literature. The top section of Table 6.3 displays the results for the baseline models, all of which were implemented from scratch.² The bottom part of Table 6.3 shows the results we get by applying ELR on top of these baselines. We observe consistent improvements over all baselines; the relative ranking of models remains the same ($\text{LM} < \text{SDM} < \text{MLM-tc} < \text{MLM-all} < \text{PRMS} < \text{FSDM}^*$), but their performance is improved by 4.4–7.3% in terms of MAP, and

¹<http://tagme.di.unipi.it/>

²We note the slight differences compared to the numbers reported in [12] and [210]. One major source of the differences is that we use a different DBpedia version, v3.9 with the updated DBpedia-Entity test collection, as opposed to v3.7 (used both in [12] and [210]). Our MLM-all and PRMS results are better than [12] because we use the top 10 fields, while top 1000 fields are used in [12] (cf. Figure 6.3). Compared to [210], we got higher scores for PRMS and lower ones for FSDM. The main reason behind this is the different choice of fields. Furthermore, as explained in Section 6.3.4, field weights are trained for each query subset in [210], while we employ a parameter-free estimation of field weights based on PRMS.

Table 6.3: Retrieval results for baseline models (Top) and with ELR applied on top of them (Bottom). Significance is tested against the corresponding baseline model. Best scores are in boldface.

Model	MAP	P@10
LM	0.1588	0.1664
MLM-tc	0.1821	0.1786
MLM-all	0.1940	0.1965
PRMS	0.1936	0.1977
SDM	0.1719	0.1707
FSDM*	0.2030	0.1973
LM + ELR	0.1693 [▲] (+6.6%)	0.1757 [▲] (+5.6%)
MLM-tc + ELR	0.1937 [▲] (+6.4%)	0.1895 [▲] (+6.1%)
MLM-all + ELR	0.2082 [▲] (+7.3%)	0.2054 ^Δ (+4.5%)
PRMS + ELR	0.2078 [▲] (+7.3%)	0.2085 [▲] (+5.5%)
SDM + ELR	0.1794 [▲] (+4.4%)	0.1812 [▲] (+6.1%)
FSDM* + ELR	0.2159[▲] (+6.3%)	0.2078[▲] (+5.3%)

by 4.5–6.1% in terms of P@10. All improvements are statistically significant. Based on these results, we answer our first research question positively: entity annotations of the query can indeed improve entity retrieval performance.

For the analysis that follows later in this section we retain four of these models: LM, PRMS, SDM, and FSDM*. This selection enables us to make a meaningful and consistent comparison across two dimensions: (i) single vs. multiple fields (LM and SDM vs. PRMS and FSDM*), and (ii) term independence vs. dependence (LM and PRMS vs. SDM and FSDM*).

6.4.2 Breakdown by Query Subsets

How are the different query subsets impacted by ELR (**RQ4b**)? Intuitively, we expect ELR to improve the effectiveness of queries that mention entities plus contain some additional terms (e.g., specifying an attribute or relation). Queries that mention a single entity, without any modifiers, are less likely to benefit from our approach.

Table 6.4 provides a breakdown of results by query type. We find that the biggest improvements are obtained for the ListSearch and QALD-2 queries (+7.5–16% in terms of MAP and +6.5–19.1% in terms of P@10). The queries

Table 6.4: Results of our ELR approach on different query types. Significance is tested against the line above; the numbers in parentheses show the relative improvements, in terms of MAP.

Model	SemSearch ES		ListSearch	
	MAP	P@10	MAP	P@10
LM	0.2485	0.2008	0.1529	0.1939
LM+ELR	0.2531 ^Δ (+1.8%)	0.2008	0.1688 [▲] (+10.4%)	0.2096
PRMS	0.3517	0.2685	0.1722	0.2270
PRMS+ELR	0.3573 (+1.6%)	0.2700	0.1956 [▲] (+14.1%)	0.2417
SDM	0.2669	0.2108	0.1553	0.1948
SDM+ELR	0.2641 (-1%)	0.2115	0.1689 ^Δ (+8.8%)	0.2174 [▲]
FSDM*	0.3563	0.2692	0.1777	0.2165
FSDM*+ELR	0.3581 (+.5%)	0.2677	0.1973 ^Δ (+11%)	0.2391 [▲]

Model	INEX-LD		QALD-2	
	MAP	P@10	MAP	P@10
LM	0.1129	0.2210	0.1132	0.0729
LM+ELR	0.1244 [▲] (+10.2%)	0.2330	0.1241 [▲] (+9.6%)	0.0836 [▲]
PRMS	0.1184	0.2240	0.1180	0.0893
PRMS+ELR	0.1303 [▲] (+10%)	0.2330	0.1343 [▲] (+13.8%)	0.1064 [▲]
SDM	0.1167	0.2250	0.1369	0.0750
SDM+ELR	0.1264 [▲] (+8.3%)	0.2340	0.1472 ^Δ (+7.5%)	0.0857
FSDM*	0.1261	0.2290	0.1364	0.0921
FSDM*+ELR	0.1332 (+5.6%)	0.2330	0.1583 ^Δ (+16%)	0.1086 [▲]

in these subsets often seek entities related to other entities; a lot can be gained here from entity linking. The INEX-LD queries are also significantly improved by ELR (with the exception for FSDM*), but the relative improvements are smaller than for ListSearch and QALD-2 (here, it is +5.6–10.2% for MAP and +1.7–5.4% for P@10), but still significant in all but one case. This set is more diverse than the other two and comprises a mixture of short entity queries, type queries, and long natural language style queries. Finally, on the SemSearch ES subset, ELR could make a significant difference only for the weakest baseline, LM. These are short keyword queries, which are already handled effectively by a fielded representation (cf. PRMS and FSDM; also note that incorporating term dependence does not improve performance).

To understand how much importance is attributed to entity-based matches, we plot the values of the $\lambda_{\{T,O,U,E\}}$ parameters for each query subset in Figure 6.4. The values are obtained by averaging the trained parameter values across all folds of the cross-validation process. We can observe a similar trend across all retrieval methods: ListSearch and QALD-2 queries are assigned the highest λ_E values, INEX-LD gets a somewhat lower but still sizable portion of the distribution, while for SemSearch ES it bears little importance.

Based on Table 6.4 and Figure 6.4, we conclude that different query types are impacted differently by the ELR method. The results confirm our hypothesis that ELR can improve complex (ListSearch and QALD-2) as well as heterogeneous (INEX-LD) queries, which involve entity relationships. On the other hand, short keyword queries, referring to a single, albeit often ambiguous, entity (SemSearch ES) are mostly unaffected.

6.4.3 Parameter Settings

How sensitive is ELR to the choice of λ parameters (**RQ4c**)? To answer this question, we compare two configurations: (i) default parameter settings, and (ii) parameters trained using the CA algorithm. The default parameters are set as follows. For SDM and FSDM, we follow [139, 140] and set $\lambda_T = 0.8$, $\lambda_O = 0.1$, and $\lambda_U = 0.1$. For the other models with ELR applied, we set λ_E to the single best performing value across the entire query set; that is, we do not train it separately for the different query subsets, like before. The resulting configurations are: (i) $\lambda_T = 0.9$ and $\lambda_E = 0.1$ for LM + ELR and PRMS + ELR, and (ii) $\lambda_T = 0.8$, $\lambda_O = 0.05$, $\lambda_U = 0.05$, and $\lambda_E = 0.1$ for SDM + ELR and FSDM* + ELR.

Table 6.5 compares retrieval results using default and trained parameters. Note that LM and PRMS do not involve any parameters, hence the empty cells.

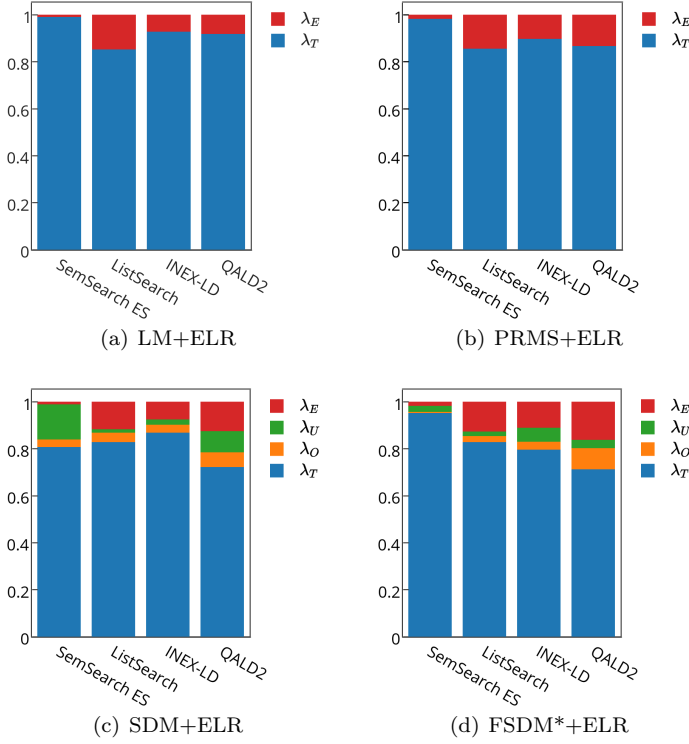


Figure 6.4: Values of the λ parameters (λ_T : unigrams, λ_O : ordered bigrams, λ_U : unordered bigrams, λ_E : entities) in our experiments, by the different query subsets (trained using Coordinate Ascent).

We find that the results are robust, i.e., ELR can improve the performance of term-based models, even with default parameter values. MAP differences are significant for all methods, except SDM + ELR. (For that model, the default λ_E value is higher than it would be optimal for SemSearch ES queries, thereby reducing overall retrieval effectiveness.) This experiment also confirms that our improvements are not a result of overfitting.

Table 6.5: Comparison of default vs. trained λ parameters over all queries. Significance is tested against the line above.

Model	Default params.		Trained params.	
	MAP	P@10	MAP	P@10
LM	0.1588	0.1664		
LM + ELR	0.1668 ^Δ	0.1724	0.1693 [▲]	0.1757 [▲]
PRMS	0.1936	0.1977		
PRMS + ELR	0.2028 [▲]	0.2035	0.2078 [▲]	0.2085 [▲]
SDM	0.1672	0.1685	0.1719	0.1707
SDM + ELR	0.1721	0.1722	0.1794 [▲]	0.1812 [▲]
FSDM	0.1969	0.1973	0.2030	0.1973
FSDM + ELR	0.2043 [▲]	0.1996	0.2159 [▲]	0.2078 [▲]

6.4.4 Impact of Entity Linking

What is the impact of the entity linking component on end-to-end entity retrieval performance (**RQ4d**)? Entity linking systems typically involve a threshold parameter that defines the required degree of certainty for linking entities. This threshold for TAGME ranges between 0 and 1, where 0 returns the maximum number of entities and 1 returns no entity. To answer the above research question, we measure retrieval performance while varying the entity linking threshold value. Figure 6.5 reports the results for the best performing model, FSDM* + ELR, for both trained and default λ parameters. Apart from the small fluctuations in the 0.4–0.6 range, retrieval performance is shown to improve as the entity linking threshold is lowered. This observation implies that ELR is robust with respect to entity linking; considering more entity annotations, even those with low confidence, improves retrieval performance. The entity linker currently used allows for the annotation of overlapping entity mentions, but it returns a single entity for each mention. In future work it might be worth experimenting with multiple entities per mention, especially in highly ambiguous situations, as our framework seems to be able to benefit from having more annotations.

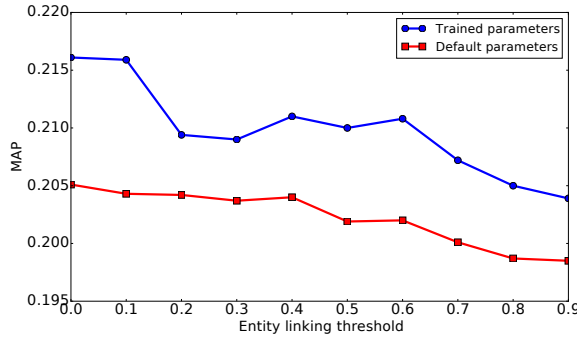


Figure 6.5: Effect of changing entity linking threshold (TAGME) on the performance of FSDM* + ELR model.

6.5 Summary

In this chapter, we have presented a novel retrieval approach (ELR) that complements term-based retrieval models with entity-based matches, using automatic means to annotate queries with entities. Our model is based on Random Markov Fields and is presented as a general framework, in which the entity-based matching component can be applied to a wide range of entity retrieval models, including standard language models, term dependence models, and their fielded variations. We have applied our approach as an extension to various state-of-the-art entity retrieval models and have shown significant and consistent improvements over all of them. The results have also shown that our model especially benefits complex and heterogeneous queries (natural language, type and relation queries), which are considered difficult queries in the context of entity retrieval. We have further demonstrated the robustness of our approach against parameter setting and entity linker configuration.

Chapter 7

Dynamic Factual Summaries for Entity Cards

In this chapter of the thesis, we turn to the presentation aspects of semantic search, and focus on the content of entity cards. Over the recent years, entity cards have become an integral element of search engine result pages (SERPs) on both desktop and mobile devices [33, 116]. Triggered by an entity-bearing search query, a card offers a summary of the entity directly on the results page, helping users to find the information they need without having to click on several documents [161]; see Figure 7.1 for examples. Studies have shown that entity cards can enhance the search experience by assisting users to accomplish their tasks [116, 146] and increase their engagement with organic search results [33].

Entity cards are complex information objects, consisting of several components such as images, entity name, short description, summary of facts, related entities, etc. The *factual summary* (or *summary*, for short), which is the focus of this chapter, is a truncated view of the top-ranked facts (i.e., predicate-value pairs) about the entity, coming from an underlying knowledge base. Summaries serve a dual purpose on the SERP: they offer a synopsis of the entity and, at the same time, can directly address users' information needs. Consider the examples in Figure 7.1, from a commercial search engine, and notice how the summary changes, depending on the query. Even though entity cards are now a commodity in contemporary search engines, to the best of our knowledge, there is no published work on how these (dynamic) summaries are generated and eval-

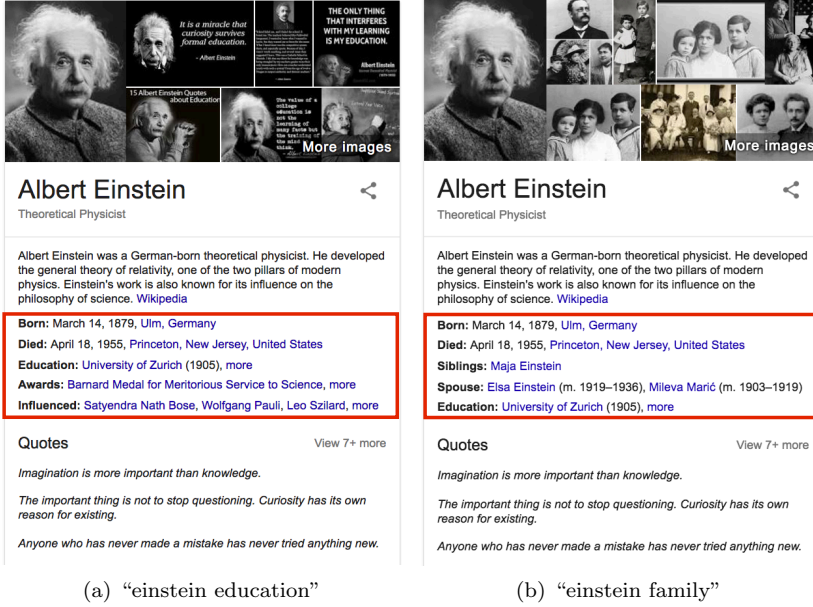


Figure 7.1: Examples of entity cards displayed on the Google SERP for different queries. The content of the factual entity summary (marked area) varies depending on the query.

uated. In this chapter, we make the first effort towards filling this important gap. In other words, the question that we address in this chapter is this:

RQ5 How to generate and evaluate factual summaries for entity cards?

Looking at the literature, the closest work related to this problem area is the task of ranking or selecting the most important facts about an entity, which has been addressed by different research communities over the recent years [48, 79, 80, 182, 183, 201]. All these works focus on the notion of *fact importance*, as the basis of ranking; a common approach is to compute PageRank-like graph centrality measures on the knowledge graph [48, 182, 183]. When considering factual summaries for entity cards, there are three important aspects that need to be addressed:

(i) Importance vs. Relevance. What is deemed important in general about an entity may be irrelevant in a given query context and vice versa. Take for example the predicate *nationality*, which is generally deemed important for a person; it, however, bears little relevance for the query “einstein awards.” This calls for *query-aware* entity summarization, where summaries are created by considering not only *fact importance*, but also *fact relevance* with respect to the query.

(ii) Summary generation. Generating an entity summary that will be shown on an entity card entails more than simply listing the top-k ranked facts. It needs to deal with, among others, issues such as semantically identical predicates (e.g., *homepage* and *website*), multivalued predicates (e.g., *children*), and presentation constraints (e.g., max height and width, which depend on the device).

(iii) Evaluation. Given the size of entity cards, it can reasonably be assumed that users consume all facts displayed in the summary. Therefore, in addition to evaluating the ranking of facts, the quality of the summary, as a whole, should also be assessed, with respect to the query. A fair comparison requires side-by-side evaluation of factual summaries by human judges.

In this chapter, we aim to address the above challenges head-on. It is important to note that this problem area is not limited to web search, where entity cards are typically displayed on the right hand side of the SERP; it also applies to any information access system that involves entities. Consider, for example, serving entity-annotated documents in response to a search query; when hovering over an entity, a context-dependent entity card is displayed to the user. Deciding when an entity card should be presented is a pivotal question, which should be addressed on its own account. This, however, is beyond the scope of this chapter. We shall assume that this decision has already been made by a separate component. Our sole focus is the generation of factual summaries for entity cards.

To address (i), we present a method for *fact ranking* that takes both importance and relevance into consideration. We design several novel features for capturing and distinguishing between importance and relevance, and combine these features in a supervised learning framework. For (ii), we introduce *summary generation* as a task on its own account, and develop an algorithm for producing a summary that meets the presentation requirements of an entity card. Concerning (iii), we build a benchmark for the fact ranking task, obtaining a large number of crowdsourced judgments with respect to both fact importance

and relevance. In addition, we evaluate the generated summaries, with regard to search queries, by performing user preference experiments via crowdsourcing. The results show that our proposed fact ranking approach significantly outperforms existing baseline systems. We also find that the summaries uniting both fact importance and relevance are preferred over those that are based on a single aspect. Overall, our results confirm the hypothesis that dynamic (query-aware) summaries are preferred over static (query-agnostic) ones; this is especially true for complex relational queries.

In summary, this chapter makes the following novel contributions:

- We present the first study on generating and evaluating dynamic factual summaries for entity cards. We formalize two specific subtasks: fact ranking and summary generation (Section 7.1).
- We introduce DynES, an approach for generating dynamic entity summaries, composed of *fact ranking* and *summary generation* steps. We introduce a set of novel features for the fact ranking task and present an algorithm for summary generation (Section 7.2).
- We design and make available a benchmark for the fact ranking task, with judgments for around 4K entity facts obtained via crowdsourcing. This test collection may be used in both query-aware and query-agnostic settings, which renders it useful not only in the context of web search, but also for entity summarization in general, which has been addressed in previous work [48, 79, 80, 182, 183] (Section 7.3).
- We perform an extensive evaluation of the proposed methods by (i) measuring fact ranking using the benchmark we developed (Section 7.4), and (ii) measuring the overall quality of summaries via a user preference study (Section 7.5).

The resources developed in the course of this chapter are presented in Appendix A.5.

7.1 Problem Statement

In this section, we describe and formally define the problem of dynamic entity summarization for entity cards.

We assume that entities are represented in a knowledge base (KB) as a set of subject-predicate-object ($\langle s, p, o \rangle$) triples.

DEFINITION 1 (Entity fact): *An entity fact (or fact, for short) f is a statement about the entity where the entity stands as subject, i.e., $f = \langle p, o \rangle$ is a predicate-object pair. We write \mathcal{F}_e to denote the set of facts about the entity e : $\mathcal{F}_e = \{ \langle p, o \rangle \mid \langle s, p, o \rangle, s = e \}$.*

This definition implies that multi-valued predicates (i.e., predicates with multiple objects) constitute multiple facts. For example, in Figure 7.1(b), there are two facts (predicate-object pairs) for the *Spouse* predicate: $\langle \text{Spouse}, \text{Elsa Einstein} \rangle$ and $\langle \text{Spouse}, \text{Mileva Marić} \rangle$. Formulating our problem based on the concept of *fact* (instead of predicate [62, 185]) allows us to handle multi-valued predicates properly. We note that the object of a fact can either be a literal or a URI. A literal object is often presented in the entity cards as it is stored in the KB (e.g., *March 14, 1879*), i.e., as a string. A URI object, on the other hand, links to another entity in the knowledge base and should be converted to a hyperlink with a human readable anchor, when shown in the card (see, e.g., *Elsa Einstein* in Figure 7.1(b)).

We now define the “goodness” of a fact for an entity summary from various aspects:

DEFINITION 2 (Importance): *The importance of fact f for an entity is denoted by i_f and reflects the general importance of that fact in describing the entity, irrespective of any particular information need.*

DEFINITION 3 (Relevance): *The relevance of fact f to query q , indicated by $r_{f,q}$, reflects how well the fact supports the information need underlying the query. E.g., a fact may hold the answer to the query or help explain why the entity is a good result for that query.*

DEFINITION 4 (Utility): *The utility of a fact, $u_{f,q}$, combines the general importance and the relevance of a fact into a single number, using a weighted combination of the two (where it is assumed that the two are on the same scale):*

$$u_{f,q} = \alpha i_f + \beta r_{f,q}. \quad (7.1)$$

For the sake of simplicity, we consider both importance and relevance with equal weights in our experiments, i.e., $\alpha = \beta = 1$. We note that this choice may be suboptimal, and different query types may require different parameter values. This exploration, however, is left for future work. The central point that we will demonstrate in our experiments is that incorporating fact relevance (as opposed to considering merely importance) leads to better entity summaries.

DEFINITION 5 (Fact ranking): *Fact ranking is the task of taking a set of entity facts (and a search query) as input, and returning facts ordered with respect to some criterion (importance, relevance, or utility). We write $\phi(\mathcal{F}_e, q)$ to denote the ranking function ($\phi : \mathcal{F} \times \mathcal{Q} \rightarrow \overline{\mathcal{F}}$) which returns a ranked list of entity facts, $\overline{\mathcal{F}}_e$.*

Once facts are ranked, they should be rendered in the form of an entity summary and presented on the entity card. Entity cards have a strong effect on users’ search experience [33, 116, 146], and the quality of entity summaries can directly impact users’ satisfaction. Therefore, simply presenting users with the top-k ranked facts is insufficient for generating an adequate summary. Additional processing steps are required, which may include, but not limited to: (i) resolving semantically identical predicates (e.g., *homepage* and *website*), (ii) grouping related predicates (e.g., *birth place* and *birth date* as single predicate *born*), (iii) dealing with multi-valued predicates (e.g., *children*), (iv) meeting the presentation constraints imposed by the SERP (e.g., max. height and width), and (v) following certain templates or editorial guidelines (e.g., always displaying birth information in the first summary line). Considering these challenges, we formulate *summary generation* as a separate task.

DEFINITION 6 (Summary generation): *Given a ranked list of entity facts $\overline{\mathcal{F}}_e$ as input, summary generation is the task of constructing an entity summary with a given maximum size (height and width), such that it maximizes user satisfaction.*

Thus, in this study, we formulate and address two tasks, as defined above: *fact ranking* and *summary generation*. Both of these tasks are novel and challenging on their own; combining the two, the overall goal of this chapter, is *dynamic entity summarization*, where “dynamic” refers to the query-dependent nature of the generated summaries (as opposed to static ones).

7.2 Approach

In this section we present our proposed approach, referred to as *DynES* (for **D**ynamic **E**ntity **S**ummarization). It consists of two steps that are performed sequentially. First, we take a set of entity facts and a query as input, and rank these facts based on utility (i.e., a combination of importance and relevance). Second, using a ranked list of facts as input, we generate an entity summary of a given size (ready to be included in the entity card).

Table 7.1: Glossary of the notations.

Name	Notation	Definition
Fact	f	$\langle p, o \rangle : f_p = p, f_o = o$
Entity facts	\mathcal{F}_e	$\{\langle p, o \rangle \mid \langle s, p, o \rangle \in KB, s = e\}$
Ranked entity facts	$\overline{\mathcal{F}}_e$	$(f_1, f_2, \dots, f_n); n = \mathcal{F}_e $
Fact frequency	$FF(f)$	$ \{\langle s, p, o \rangle \mid \langle s, p, o \rangle \in KB, p = f_p, o = f_o\} $
Fact frequency of predicate	$FF_p(p)$	$ \{\langle s', p', o' \rangle \mid \langle s', p', o' \rangle \in KB, p = f_p\} $
Fact frequency of object	$FF_o(o)$	$ \{\langle s', p', o' \rangle \mid \langle s', p', o' \rangle \in KB, o = f_o\} $
Entity Frequency of fact	$EF(f)$	$ \{e \mid e \in \mathcal{E}, f \in \mathcal{F}_e\} $
Entity frequency of predicate	$EF_p(p)$	$ \{e \mid e \in \mathcal{E}, \exists f \in \mathcal{F}_e : f_p = p\} $
Entity frequency of object	$EF_o(o)$	$ \{e \mid e \in \mathcal{E}, \exists f \in \mathcal{F}_e : f_o = o\} $
Entity frequency of predicate for a given type	$EF_p(p, t)$	$ \{e \mid e \in \mathcal{E}, t \in type(e), \exists f \in \mathcal{F}_e : f_p = p\} $
Type frequency of predicate	$TF_p(p)$	$ \{t \mid \langle s', p', o' \rangle \in KB : p' = p, t \in type(s')\} $

7.2.1 Fact Ranking

We approach the entity fact ranking task as a learning to rank problem, where we optimize the ranking of facts w.r.t. a target label. Formally, we define each fact-query pair (f, q) as a learning instance and represent it with a feature vector \mathbf{x}_i . Then, a pointwise ranking function $h(\mathbf{x}_i)$ generates a score \mathbf{y}_i . We choose *fact utility* to be our target label, where importance and relevance are taken into account with equal weights. We note that the learning framework allows us to optimize for any other target (e.g., more bias towards importance or relevance). The features we introduce here are designed to be able to handle different types of queries, ranging from named entity queries to verbose natural language queries. We acknowledge that fact ranking could benefit a lot from a query log; however, since we do not have access to that, our feature design is limited to publicly available data sources. Also note that for long tail (unpopular) entities the search log would not be of much help.

Before we proceed, a word on notation and terminology; see Table 7.1 for a summary. The underlying knowledge base (KB) consists of $\langle s, p, o \rangle$ triples, where the subject s is an entity identifier. To help explain the intuition behind the concepts we introduce, we draw an analogy to document retrieval. The concepts fact frequency ($FF(f)$) and entity frequency ($EF(f)$) are loosely analogous to collection frequency and document frequency. The former counts the total number of triples matching a fact, while the latter considers the number of entities that have that fact. Entities have types assigned to them in the KB (typically several, but at least one per entity). Each entity type may be viewed as a document, with predicates of the entities with that type being terms of the

document. Using this analogy, the two type-related concepts, entity frequency of predicate for a type ($EF_p(p, t)$) and type frequency of predicate ($TF_p(p)$), are similar to term frequency and document frequency. The former counts the number of times a given predicate appears in the virtual document of the type (i.e., number of entities with that predicate and type), the latter counts the number of documents (types) which contain that predicate.

Next, we describe the features we designed for capturing fact importance and fact relevance. Unless indicated by a reference, the feature is introduced in this chapter, and, to the best of our knowledge, represents a novel contribution.

Importance Features

The first set of features reflects the general importance of a fact for a given entity and are computed based on various statistics from the knowledge base.

- *Normalized fact frequency.* The feature counts the overall frequency of the fact in the knowledge base, normalized by the total number of $\langle s, p, o \rangle$ predicates in the knowledge base ($|\mathcal{F}|$):

$$NFF(f) = \frac{FF(f)}{|\mathcal{F}|}. \quad (7.2)$$

We compute two other variants of this feature, $NFF_p(p)$ and $NFF_o(o)$, where the numerator is replaced with fact frequency of predicate $FF_o(o)$ and entity frequency of object $FF_o(o)$, respectively.

- *Normalized entity frequency.* This feature captures the entity-wise frequency of a fact, normalized by the cardinality of entities in the knowledge base ($|\mathcal{E}|$):

$$NEF(f) = \frac{EF(f)}{|\mathcal{E}|}. \quad (7.3)$$

Similarly to the previous feature, we compute predicate and object variations of the feature ($NEF_p(f)$ and $NEF_o(f)$) by substituting $EF_p(p)$ and $EF_o(o)$ in the numerator.

- *Type-based importance.* The importance of a fact for an entity may not always be captured by the overall knowledge base statistics; the specific entity types should be taken into considerations. This is of particular importance for predicates, as their frequencies are biased towards the most frequent types: predicates of less frequent types have low frequency, although they

might be important for that specific type. As introduced in [185], the type-based importance is computed as:

$$TypeImp(p, e) = \sum_{t \in type(e)} EF_p(p, t) \cdot \log \frac{|\mathcal{T}|}{TF_p(f)}, \quad (7.4)$$

where $|\mathcal{T}|$ is the total number of types in the knowledge base.

- *Predicate specificity.* This feature identifies predicate-specific facts; i.e., facts with a common object, but rare predicate. Take for example the fact $\langle capital, Ottawa \rangle$ for the entity CANADA, where the predicate is relatively rare (only for capital cities) and the object is frequent. Predicate specificity, hence, combines the fact frequency of the object with the inverse entity frequency of the predicate:

$$PredSpec(f) = FF_o(o) \cdot \log \frac{|\mathcal{E}|}{EF_p(p)}. \quad (7.5)$$

- *Object specificity.* In contrast to the previous feature, object specificity captures facts with rare objects, but popular predicates; e.g., the object of value of $\langle Birth\ date, 1953-10-01 \rangle$ is relatively unique, while the predicate is frequent. Formally:

$$ObjSpec(f) = EF_p(p) \cdot \log \frac{|\mathcal{F}|}{FF_o(o)}. \quad (7.6)$$

It is worth noting that both *PredSpec* and *ObjSpec* represent specificity of a fact and highlight important features from the opposite ends of the spectrum; that is, a fact should have either high *PredSpec* or high *ObjSpec* to be considered important.

- *Other features.* Two other binary features are employed: *IsNum* identifies whether the object is a number or not, and *IsEntity* returns true if the object is an entity URI.

Relevance Features

The idea behind the second group of features is to determine the relevance of a fact with respect to the information need, specified by the search query (q). Various sources of information are used to extract these features: the query itself, linked entities in the query, retrieved entities in response to the query, and an external corpus to identify the semantic similarity between terms.

- *Context length.* This feature identifies the number of terms in the query that are not linked to any entity. Formally, it is defined as:

$$ConLen(q) = |\{t | t \in q, t \notin Link(q)\}|, \quad (7.7)$$

where t denotes a term and $Link(q)$ is the set of query terms that are linked to an entity. To obtain entity annotations for queries, we utilize the TAGME entity linking system [69] through its API. This feature helps to distinguish keyword queries from other complex queries, such as list or natural language queries. The underlying motivation is that in case of keyword queries targeting a specific entity (e.g., “*eiffel tower*”), users are more concerned about the most important facts of that entity, while for longer and more complex queries (e.g. “*points of interest in paris*”), entity facts that address the underlying information need (i.e., are relevant to the query) would be deemed more useful from the user’s perspective.

- *Semantic similarity.* In order to address the vocabulary mismatch between queries and facts, we compute their semantic similarity based on word embeddings, following recent common practice [35, 164]. Specifically, we use Word2Vec [142] with the 300 dimension vectors trained on the Google news dataset, and employ two methods to compute string similarity: aggregated and centroid similarity. The former aggregates the word-wise cosine similarity between each pair of words of in the two strings:

$$SemSimAgg(s, s') = \text{agg func}_{w \in s, w' \in s'} \cos(\vec{w}, \vec{w'}), \quad (7.8)$$

where the w represents a word of string s , and average and maximum are used as the aggregation functions. The second approach performs the aggregation at the vector level and computes the similarity of centroid vectors $\vec{C}, \vec{C'}$: $SemSimCent(s, s') = \cos(\vec{C}, \vec{C'})$.

In our settings, we substitute s with the query and s' with a predicate or object, thereby computing the semantic similarity for query-predicate and query-object pairs.

- *Lexical similarity.* In addition to semantic similarity, we also compute lexical similarity to capture spelling mismatches. Following [164], we employ the Jaro edit distance and apply it to query-predicate and query-object pairs (i.e., $LexSim_p, LexSim_o$).
- *Inverse rank.* This feature promotes facts with an object value that is considered highly relevant to the query [96]. We rank entities from the KB w.r.t.

the query and return the inverse rank of the entity that matches the object value (of the fact). Formally:

$$iRank(q, f) = \frac{1}{rank(f_o, Ret(q))}, \quad (7.9)$$

where $Ret(q)$ is the list of retrieved entities for the query q . In our experiments, we take a state-of-the-art entity retrieval approach [93] to compute this feature.

- *Other features.* Two additional features we use are (i) the *IsLinked* function that looks up the fact object among the linked entities of the query, and (ii) the Jaccard similarity (*JaccSim*) between the terms of the query and predicate and object of the fact (separately).

7.2.2 Summary Generation

We now turn to the task of generating a summary from the ranked list of entity facts. Our proposed approach, presented in Algorithm 2, has three main features that are believed to result in high quality summaries for entity cards: (i) it creates a summary of a given size, (ii) identifies identical facts and filters out unnecessary ones, and (iii) handles multi-valued predicates. We note that summary generation may involve additional processing steps (cf. Section 7.1). Our focus of attention here is to emphasize the essence of entity summary generation as a separate task and to address the minimum requirements for summaries that will be used on entity cards.

The algorithm takes as input a ranked list of entity facts $\overline{\mathcal{F}}_e$, and maximum height and width thresholds τ_h, τ_w . The output is maximum τ_h summary lines, each with a heading and one or multiple corresponding values with a maximum width of τ_w characters. Figure 7.2 illustrates these concepts.

The first step in generating the summary is to map knowledge base predicates to human readable labels. This is of particular importance as the same fact may be described with different predicates; e.g., both `dbo:BirthDate` and `dbp:DateOfBirth` have the same meaning. Recognizing these semantically identical predicates and mapping them to a canonical name is encapsulated in the function *Predicate-Name Mapping* (line 1 of Algorithm 2). Depending on the underlying knowledge base, this task can be highly non-trivial. In our experiments using DBpedia, we take two predicates as semantically identical if one of the followings holds: (i) one predicate name (irrespective the prefix) is a plural

Algorithm 2 Summary generation algorithm

Input: Ranked facts $\overline{\mathcal{F}}_e$, max height τ_h , max width τ_w
Output: Entity summary *lines*

```

1:  $\mathcal{M} \leftarrow \text{Predicate-Name Mapping}(\overline{\mathcal{F}}_e)$ 
2:  $\text{headings} \leftarrow []$  //Determine line headings
3: for  $f$  in  $\overline{\mathcal{F}}_e$  do
4:    $p_{\text{name}} \leftarrow \mathcal{M}[f_p]$ 
5:   if ( $p_{\text{name}} \notin \text{headings}$ ) AND ( $\text{size}(\text{headings}) \leq \tau_h$ ) then
6:      $\text{headings.add}((f_p, p_{\text{name}}))$ 
7:   end if
8: end for

9:  $\text{values} \leftarrow []$  //Determine line values
10: for  $f$  in  $\overline{\mathcal{F}}_e$  do
11:   if  $f_p \in \text{headings}$  then
12:      $\text{values}[f_p].\text{add}(f_o)$ 
13:   end if
14: end for
15:  $\text{lines} \leftarrow []$  //Construct lines
16: for ( $f_p, p_{\text{name}}$ ) in  $\text{headings}$  do
17:    $\text{line} \leftarrow p_{\text{name}} + \text{'.'}$ 
18:   for  $v$  in  $\text{values}[f_p]$  do
19:     if  $\text{len}(\text{line}) + \text{len}(v) \leq \tau_w$  then
20:        $\text{line} \leftarrow \text{line} + v$  //Add comma if needed
21:     end if
22:   end for
23:    $\text{lines.add}(\text{line})$ 
24: end for
```

form of the other, (ii) all object values of two predicates are identical, while the predicate names partially match each other.

The summary is built in three stages. First (from line 2 of Algorithm 2), the headings for each summary line are selected; the algorithm keeps the unique predicates corresponding to facts, such that the number of predicates does not exceed the threshold τ_h . Next (from line 9 of Algorithm 2), the values for each line are selected; this is the part where values for the multi-valued predicates are collected. Finally (from line 16 of Algorithm 2), the heading (human-readable

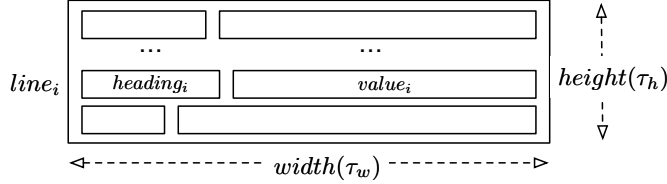


Figure 7.2: Structure of an entity summary in entity cards.

predicate) and object values are concatenated together such that they meet the width constraint τ_w .

7.3 Establishing a Benchmark

There is no existing test set for fact ranking that considers queries. Therefore, we develop and make publicly available a fact ranking benchmark via crowd-sourcing, as we shall explain in this section.

7.3.1 Data sources

To build the collection, we need a set of entity-bearing queries with their corresponding entities that should be summarized. Below, we describe the data sources used for this purpose.

Knowledge base We use DBpedia (version 2015-10) as our knowledge base, and restrict entities to those with a title and short abstract (`rdfs:label` and `rdfs:comment`); entities without these attributes may not be of sufficient importance to be presented as a card. Since we are concerned with generating entity summaries, we blacklisted predicates that are related to the other parts of the entity card, such as image, entity name and type, abstracts, and related entities. Furthermore, we filtered out noisy predicates that consist of numbers or a single character. To ensure that entity summarization is a meaningful exercise (i.e., entities have enough number of facts to select from), our collection is restricted to entities with at least 5 “valid” predicates after filtering.

Queries The queries are taken from the DBpedia-entity dataset [12], which is a standard test collection for entity retrieval [12, 47, 128, 149, 210]. It contains 485 queries from 4 different categories: **SemSearch ES** consisting of named

entity queries (e.g., “ashley wagner,” “carolina”), **List Search** made up of different entity list queries (e.g., “ratt albums”), **QALD-2** containing natural language queries (e.g., “Who founded Intel?”), and **INEX-LD** consisting of general keyword queries, including type, relation, and attribute queries (e.g. “vietnam war facts,” “England football player highest paid”).

7.3.2 Selecting Entity-Query Pairs

Given a set of queries, the next step is to form query-entity pairs that will constitute the input for query-dependent entity summarization. For each query in the DBpedia-entity collection, we select a single entity that is (i) known to be relevant and (ii) generally the most easily “retrievable.” We measure retrievability by considering several entity retrieval approaches from the literature and establishing a voting schema among them. Bear in mind that we do not decide whether the entity card should be displayed or not; we assume that our information access system generates a card for a retrievable and presumably relevant entity. We also note that our focus of attention in this chapter is on generating a summary for a given (assumed to be relevant) entity and not on the entity retrieval task itself. We therefore treat entity retrieval as a black box and combine several approaches to ensure that the findings are not specific to any particular entity retrieval method.

Formally, for a query q , we define \widehat{E}_q as the set of relevant entities according to the ground truth, and $E_{q,m}$ as the ranked list of entities retrieved by method $m \in M$, where M denotes the collection of retrieval methods. A single entity e is selected for q such that:

$$\begin{aligned} e &= \arg \max_{e_q \in E_q} \sigma(e_q), \\ \sigma(e_q) &= \frac{1}{|M|} \sum_{m \in M} \frac{1}{\text{rank}(e_q, E_{q,m})}, \\ E_q &= \{e | e \in \widehat{E}_q, \exists m \in M : e \in E_{q,m}\}. \end{aligned}$$

Basically, we select the entity that is retrieved at the highest rank by all methods, on average. (If the entity is not retrieved by method m , then the reciprocal rank is set to 0 by definition.) For our experiments, we consider 6 different entity retrieval approaches: BM25 and BM25F-all from [12], SDM and FSDM from [210], and SDM+ELR and FSDM+ELR from [93].

Using our voting mechanism, we were able to extract relevant entities for 421 queries. The average σ score for the selected entities is 0.32, meaning that

	Very important	Important	Unimportant	Relevance (total)
Very relevant	114	71	43	228
Relevant	241	218	115	574
Irrelevant	414	694	2159	3267
Importance (total)	769	983	2317	

Figure 7.3: Distribution of entity facts for different levels of importance and relevance.

these entities are retrieved among the top-3 rank positions, on average, by all retrieval methods. For the remaining 64 queries, none of the above methods could retrieve relevant results in the top-100 ($E_q = \emptyset$). These were mostly complex natural language queries. To avoid introducing any bias against these in our collection, we still included them and randomly selected one entity per query from the ground truth entities (\widehat{E}_q). Due to pragmatic reasons (i.e., keeping the evaluation costs sensible), we randomly chose 100 entity-query pairs, evenly spread across the 4 different query categories. For each entity in this selection, we extracted the facts from the underlying knowledge base, resulting in a total of approximately 4K facts.

7.3.3 Fact Ranking Test Set

We build our fact ranking test set by collecting human judgments using the CrowdFlower (CF) platform. We designed two independent tasks to assess the importance and relevance of entity facts. In one task, workers were presented with a single fact for an entity, and were asked to rate the importance of the fact w.r.t. the entity on a 3-point Likert scale: unimportant, important, or very important. In the other task, the search query was also presented in addition, and workers were asked to assess the relevance of the entity fact w.r.t. the query using a 3-point scale: irrelevant, relevant, or very relevant. For both tasks, workers were educated on the concepts of entity cards and entity summaries via examples. They were also supplied with a short description about the entity and a link to the entity’s Wikipedia page, to learn more about the entity, if needed.

Several policies were adopted to obtain high quality results from the crowd-sourcing experiments. Only the most trusted workers (level 3 on CF) were allowed to perform the tasks, and they had to retain 80% accuracy throughout the job. The workers who did not meet this threshold or spent very little time on each record, were banned from the rest of task and their judgments were considered untrusted. We also paid a reasonably high price (¢1 per record) to keep the high quality workers satisfied. Each record was judged by 5 different workers and the Fleiss' Kappa inter-annotator agreement was moderate: 0.52 and 0.41 for importance and relevance, respectively.

Figure 7.3 shows the distribution of the collected judgments. Considering importance judgments on their own, nearly half of the facts are rated as unimportant, while the rest are (almost evenly) distributed among the two other categories. As for relevance, around 81% of the facts are considered irrelevant, 14% are relevant, and only 5% are judged as very relevant. Taking the combination of these two aspects, the highest correlation is between unimportant and irrelevant facts (53%), while the lowest one is among unimportant facts that are highly relevant to the query (only 1% of all facts). Following our definition of utility (cf. Section 7.1), we combine importance and relevance with equal weights. In the end, we have three sets of ground truth for fact ranking, based on importance (3-point scale), relevance (3-point scale), and utility (5-point scale).

7.4 Fact Ranking Results

In this section we present our experimental results for the fact ranking task, and address the following research questions:

RQ5a: How does our fact ranking approach compare against the state-of-the-art?

RQ5b: How does fact ranking performance compare with respect to importance vs. relevance vs. utility?

7.4.1 Settings

We chose Gradient Boosted Regression Trees [74] as our learning model because it is shown as one of the best performing learning algorithms on a range of tasks [28, 136, 194]. We set the number of trees to 100 and the maximum depth of the trees to approximately 10% of our feature set; that is $d = 3$ when

all features are used and $d = 2$ when trained either on importance or relevance features. All the experiments are performed using 5-fold cross validation, ensuring that facts of the same entities are kept together. We report on NDCG at ranks 5 and 10.

We use a two-tailed paired t-test to measure statistical significance. Significant improvements are marked with Δ ($\alpha = 0.05$) or \blacktriangle ($\alpha = 0.01$), and we write ∇ and \blacktriangledown for a drop in performance (for $\alpha = 0.05$ and $\alpha = 0.01$, respectively); $^\circ$ stands for no significance.

7.4.2 Experiments

We report on various instantiations of our fact ranking approach in order to be able to tell apart the effect of considering fact relevance in addition/as opposed to fact importance: (i) **DynES** uses all features and is trained on utility judgments; (ii) **DynES/imp** considers importance features only and is trained on importance judgments; and (iii) **DynES/rel** employs relevance features only and is trained on relevance judgments.

We identified three approaches from the literature that can be considered as fact ranking baselines (cf. Section 2.2.5):

- **RELIN** [48] employs a variation of the PageRank algorithm to rank RDF triples for each entity. The scores indicate the importance of a fact for an entity and are computed based on the relatedness (or similarity) between two facts as well as their informativeness.
- **SUMMARUM** [182] computes the PageRank score for all entities in the knowledge graph and then takes the sum of the subject and the object scores as the final score for each entity fact.
- **LinkSUM** [183] combines PageRank scores with the Backlink algorithm [192] (a set-based heuristic for discovering related entities).

RELIN and our DynES-based approaches generate the scores for both URI and literal facts, while SUMMARUM and LinkSUM can only score URI entity facts and do not consider literal facts (cf. Section 7.1 for URI vs. literal facts). Therefore, when comparing SUMMARUM and LinkSUM with other approaches, we report on the results in a tailored setting, where literal facts are filtered out from the results of other approaches. We implemented RELIN based on the source code kindly provided by the authors and set the parameter $\lambda = 1$, as it delivers robust results across various rank positions [48]. We obtained results

Table 7.2: Comparison of fact ranking against state-of-the-art approaches with URI-only objects. Significance for lines $i > 3$ is tested against lines 1, 2, 3, and for lines 2, 3 is tested against lines 1, 2.

Model	Importance		Utility	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10
RELIN	0.6368	0.7130	0.6300	0.7066
LinkSum	0.7018 [△]	0.7031 [°]	0.6504 [°]	0.6648 [°]
SUMMARUM	0.7181^{▲°}	0.7412^{°△}	0.6719^{°°}	0.7111^{°°}
DynES/imp	0.8354 ^{▲▲▲}	0.8604 ^{▲▲▲}	0.7645 ^{▲▲▲}	0.8117 ^{▲▲▲}
DynES	0.8291^{▲▲▲}	0.8652^{▲▲▲}	0.8164^{▲▲▲}	0.8569^{▲▲▲}

for SUMMARUM and LinkSUM from their publicly available API,¹ offered for DBpedia ver. 2015-10.

7.4.3 Results

To answer our first research question, we compare the baseline systems with DynES and DynES/imp with respect to importance and utility. (As these baseline systems only address the importance aspect, we do not report on relevance.) As shown in Table 7.2, our fact ranking approaches perform significantly better than all baselines (16% relative improvement of DynES over SUMMARUM w.r.t. NDCG@10). Interestingly, none of the differences between the baseline systems are significant with respect to utility, even though many of the differences are significant for importance. We select RELIN as our baseline for the rest of the experiments, because it performs in par with other systems in terms of utility. More importantly, it is the only system that can rank both URI and literal facts; SUMMARUM and LinkSUM discard all literal facts (even important ones such as birth and death date), which is not desired for our entity card generation use-case.

For the second research question, we compare RELIN against the three variants of our approach. Table 7.3 presents the results with respect to importance, relevance, and utility. Our first observation is that all DynES variants significantly outperform RELIN in all aspects; the relative improvements of DynES for NDCG@10 are 48%, 50%, and 47% with regards to importance, relevance,

¹<http://km.aifb.kit.edu/services/link/>

Table 7.3: Fact ranking results w.r.t importance, relevance, and utility. Significance for line $i > 1$ is tested against lines $1 \dots (i - 1)$.

Model	Importance		Relevance	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10
RELIN	0.4733	0.5261	0.3514	0.4255
DynES/imp	0.7851[▲]	0.7959[▲]	0.4671 [▲]	0.5305 [▲]
DynES/rel	0.5756 ^{▲▼}	0.6151 ^{▲▼}	0.5269 ^{▲◦}	0.5775 ^{▲◦}
DynES	0.7672 ^{▲◦▲}	0.7792 ^{▲◦▲}	0.5771^{▲▲▲}	0.6423^{▲▲▲}

Model	Utility	
	NDCG@5	NDCG@10
RELIN	0.4680	0.5322
DynES/imp	0.7146 [▲]	0.7506 [▲]
DynES/rel	0.6138 ^{▲▼}	0.6536 ^{▲▼}
DynES	0.7547^{▲▲▲}	0.7873^{▲▲▲}

and utility, respectively. We also find that all systems perform better in absolute terms, when they are compared against importance or utility as opposed to relevance. Systems that are designed to capture only the importance of facts (i.e., RELIN and DynES/imp) achieve lower NDCG scores for relevance and utility than for importance. DynES/rel and DynES, on the other hand, deliver better results for utility than for importance. These results, while reflecting the expected behavior of the compared approaches, provide evidence that: (i) capturing the relevance of facts needs special treatment and different features from fact importance, and (ii) capturing the relevance aspect is considerably more challenging than importance.

7.4.4 Feature Analysis

In Table 7.4 we report on a feature ablation study, where we remove a single feature based on the relative difference it makes in terms of ranking performance (w.r.t. utility). The table shows the top-13 features individually. Interestingly, importance and relevance features are evenly distributed among the most influential features. The top-2 features (NEF_p , $TypeImp$) are computed based on fact predicates, while the rest of importance features involve fact objects. As for

Table 7.4: Fact ranking performance by removing features one-by-one from the full feature set (first line); features are sorted by the relative difference they make in terms of NDCG@10.

Group	Removed feature	NDCG@10	$\Delta\%$	p
	DynES - all features	0.7873	-	-
Imp.	- NEF_p	0.7757	-1.16	0.08
Imp.	- $TypeImp$	0.7760	-1.13	0.14
Rel.	- $LexSim_o/Max.$	0.7793	-0.8	0.20
Rel.	- $iRank$	0.7793	-0.8	0.22
Rel.	- $SemSimAgg_o/Avg.$	0.7801	-0.72	0.25
Imp.	- $IsURI$	0.7802	-0.71	0.22
Imp.	- $PredSep$	0.7812	-0.61	0.25
Rel.	- $ConLen$	0.7819	-0.54	0.35
Imp.	- $ObjSep$	0.7826	-0.47	0.38
Imp.	- NEF	0.7828	-0.45	0.41
Rel.	- $SemSimCent_p$	0.7834	-0.39	0.49
Imp.	- $IsNumber$	0.7851	-0.22	0.72
Rel.	- $JaccSim_o$	0.7851	-0.22	0.70

relevance features, we see four different versions of similarity features, three of them computed based on the object values: $LexSim_o$, $SemSimAgg_o$, $JaccSim_o$. The feature ablation study reveals that some variant of each of the proposed features (except NFF) are among the top features, indicating that each of the designed features captures utility from a specific angle. To further analyze the performance of each individual feature, we compared the features based on their performance as single feature rankers. The results show a large degree of overlap with the top-13 features identified in Table 7.4.

7.5 Summary Generation Results

In this section we present on our experimental results for the summary generation task and address the following research questions:

RQ5c: How satisfied are users with the different types of summaries?

RQ5d: How does our summary generation algorithm affect user preferences?

7.5.1 Settings

To evaluate the generated summaries, we performed side-by-side evaluation via crowdsourcing. Workers were presented with two summaries of the same entity along with the corresponding query, and were asked to select the preferred summary w.r.t this query, or the tie option when the two summaries are equally good. Providing users with a tie option enables us to clearly discern user preferences and to avoid randomness in the collected judgments. To avoid any bias, the summaries were randomly placed on the left or right side. We collected 10 judgments from level-3 workers for each pair of summaries. The width and height threshold of Algorithm 1 are set to $\tau_w = 70, \tau_h = 5$ in all experiments, inspired by entity cards used in present-day web search engines. The final results are presented as the total number of user agreements on win, loss, and tie options. We also compute the robustness index (RI) [50], defined as $\frac{N^+ - N^-}{|Q|}$ with N^+ and N^- being the number of wins and losses, and $|Q|$ denoting the total number of queries.

7.5.2 Experiments

We performed the following side-by-side evaluation of summaries to answer RQ5c. In all cases, we apply the same Algorithm 2, but feed it with a ranked list of facts from different sources. (i) **DynES vs. DynES/imp** uses DynES vs. DynES/Imp for fact ranking; (ii) **DynES vs. DynES/imp** uses DynES vs. DynES/rel for fact ranking; (iii) **DynES vs. RELIN** compares DynES vs. the top-5 ranked facts from RELIN; and (iv) **Utility vs. Importance** is an oracle comparison, by taking perfect fact ranking results from crowdsourcing.

For RQ5d, we compare our summary generation algorithm with three variations of the algorithm, all applied to the utility-based fact ranking. We compare DynES with: (i) **DynES(-GF)(-RF)**, which is Algorithm 2, without grouping of facts with the same predicate (GF), and removing identical facts (RF); (ii) **DynES(-GF)**, which is Algorithm 2, without the GF feature; and (iii) **DynES(-RF)**, which is Algorithm 2, without the RF feature.

7.5.3 Results

Table 7.5 shows the results of summary comparison for different fact ranking methods. According to the first row, query-dependent summaries (DynES) are preferred over query-agnostic ones (DynES/imp) for about half of the queries; the opposite is observed for 31% of queries. We performed the same comparison

Table 7.5: Side-by-Side evaluation of summaries for different fact ranking methods.

Model	Win	Loss	Tie	RI
DynES vs. DynES/imp	46	23	31	0.23
DynES vs. DynES/rel	75	12	13	0.63
DynES vs. RELIN	95	5	0	0.90
Utility vs. Importance	47	16	37	0.31

with the oracle setting (last row of Table 7.5) and witnessed a similar number of wins, but less losses, which is expected due to imperfect fact ranking. This verifies that the preference of dynamic over static summaries is true for both automatic and human generated summaries. When comparing DynES vs. DynES/rel, we observe that DynES wins in 75% of the cases, signifying that a combination of both importance and relevance is required for a profound entity summary. Finally, we measured the accumulated effect of the improved fact ranking and summary generation method by comparing DynES against RELIN. The results show the superiority of DynES, with a robustness index of 0.9. Based on these experiments, the answer to RQ5c is that dynamic utility-based summaries are indeed preferred over static importance- or relevance-based summaries.

Table 7.6 presents the comparison between different summary generation algorithms. The results clearly show that DynES summaries are preferred over the ones that do not address the individual presentation aspects. It also reveals that the grouping of multivalued predicates (GF features) is perceived as more important by the users than the resolution of identical facts (RF feature). Based on these results, our answer to RQ5d is that the summary generation algorithm has a major effect on user preferences and thus it should be paid attention within the entity summarization task.

7.5.4 Analysis

We analyze the differences in users preferences on the query level for the first two set of summarization experiments in Figure 7.4; i.e., we compare DynES with DynES/imp and DynES/rel. Each value in our query preference distribution indicates the number of users who preferred DynES summaries over DynES/imp (or DynES/rel) summaries; ties are ignored. Considering all queries (the black

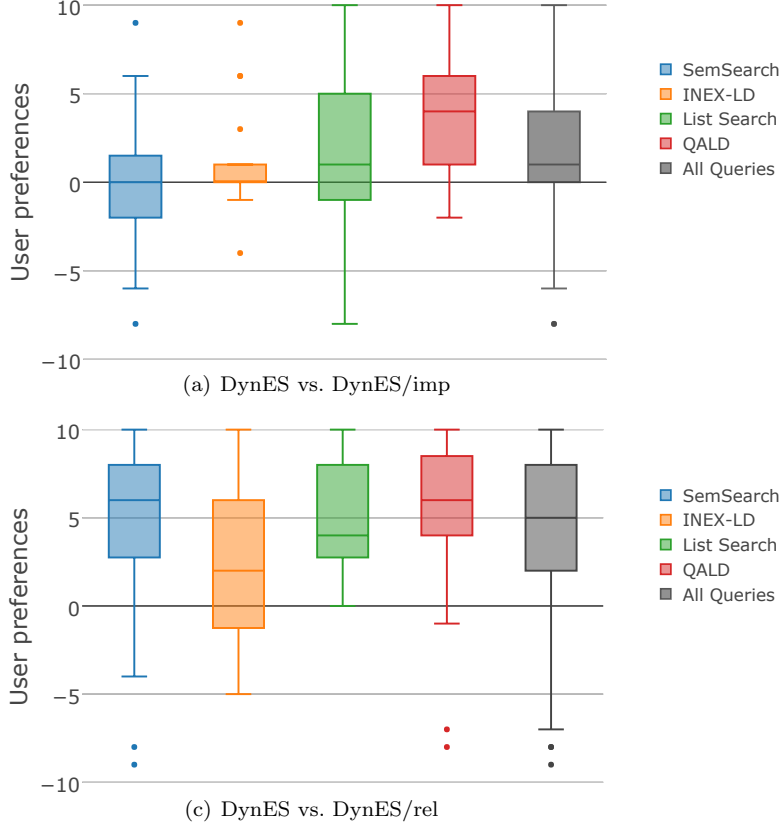


Figure 7.4: Boxplot for distribution of user preferences for each query subset. Positive values show that DynES is preferred over DynES/imp or DynES/rel.

boxes), we observe that the utility-based summaries (DynES) are generally preferred over the other two, and especially over the relevance-based summaries (DynES/rel). These summaries are highly biased towards the query and cannot offer a concise summary; the utility-based summaries, on the other hand, can strike a balance between diversity and bias. Considering the query type breakdowns in Figure 7.4(a), we observe that the ListSearch and QALD queries, which are identified as complex entity-oriented queries, benefit the most from

Table 7.6: Side-by-side evaluation of summaries for different summary generation algorithms.

Model	Win	Loss	Tie	RI
DynES vs. DynES(-GF)(-RF)	84	1	15	0.83
DynES vs. DynES(-GF)	74	0	26	0.74
DynES vs. DynES(-RF)	46	2	52	0.44

utility-based summaries. Interestingly, however, we do not observe any clear preferences for SemSearch and INEX-LD queries. This attests that our approach can generate dynamic summaries without hurting named entity and keyword queries.

7.6 Summary

In this chapter, we have introduced the novel problem of dynamic entity summarization: generating query-dependent entity summaries for entity cards. We have formulated two specific subtasks: fact ranking and summary generation. The first task entails the ranking of facts (predicate-object pairs) with respect to importance and/or relevance. We have addressed it in a learning to rank framework, and have demonstrated significant improvements over the most comparable state-of-the-art baselines using a purpose-build test collection. The second task concerns the rendering of ranked facts as a summary to be displayed on the entity card. We have presented a summary generation algorithm and have shown via a series of user preference comparisons that users favor dynamic (query-dependent) summaries over static (query-agnostic) ones.

Chapter 8

The DBpedia-Entity v2 Test Collection

In the previous chapters, we addressed several research questions related to semantic search. We now present a standard test collection that is designed to further research and development in this area, referred to as the *DBpedia-Entity v2* collection.

The DBpedia-Entity collection was first introduced by Balog and Neumayer [12], generated by assembling search queries from a number of entity-oriented benchmarking campaigns and mapping relevant results to DBpedia. Over the past years, this has become a standard test collection for the entity retrieval task, see [47, 93, 128, 149, 210]. The main objective of this chapter is to present a new, extended version of this test collection. We shall refer to the original collection in [12] as *DBpedia-Entity v1* and to our updated version as *DBpedia-Entity v2*. The DBpedia-Entity v2 is available for evaluating three semantic search tasks: (i) entity retrieval [98], (ii) target type identification [76], and (iii) entity summarization [95]. For each of these tasks, we take the queries from the DBpedia-Entity v1 collection and collect the crowdsourced judgments using DBpedia version 2015-10.¹ as the underlying knowledge base. All resources, including queries, relevance assessments (qrels), baseline runs, their evaluation results, and further details on indexing and preprocessing are made publicly available at <http://tiny.cc/dbpedia-entity>.

¹<http://wiki.dbpedia.org/Downloads2015-10>.

In this chapter, we first provide an overview of the DBpedia-Entity v2 collection in Section 8.1. We then describe the entity retrieval and target type identification subcollections of DBpedia-Entity v2 in Sections 8.2 and 8.3. For the entity summarization collection, we the reader to Section 7.3. We end with a summary in Section 8.4.

8.1 The Test Collection

This section describes the test queries and the reference knowledge base.

8.1.1 Test Queries

The queries in *DBpedia-Entity v2* are the same as in *v1*. We distinguish between four categories of queries:

- **SemSearch ES** queries are from the ad hoc entity search task of the Semantic Search Challenge series [26, 85]. These are short and ambiguous queries, searching for one particular entity, like “brooklyn bridge” or “08 toyota tundra.”
- **INEX-LD** queries are from the ad hoc search task at the INEX 2012 Linked Data track [193]. They are IR-style keyword queries, e.g., “electronic music genres.”
- **List Search** comprises queries from the list search task of the 2011 Semantic Search Challenge (SemSearch LS) [26], from the INEX 2009 Entity Ranking track (INEX-XER) [60], and from the Related Entity Finding task at the TREC 2009 Entity track [14]. These queries seek a particular list of entities, e.g., “Professional sports teams in Philadelphia.”
- **QALD-2** queries are from the Question Answering over Linked Data challenge [127]. These are natural language questions that can be answered by DBpedia entities, for example, “Who is the mayor of Berlin?”

Originally, the SemSearch queries were evaluated using crowdsourcing on a 3-point relevance scale. All other benchmarks employed expert evaluators (trained assessors or benchmark organizers/participants) and have binary judgments.

8.1.2 Knowledge Base

We use DBpedia as our knowledge base; it is often referred to as “the database version of Wikipedia.” DBpedia is a community effort, where a set of rules (“mappings”) are collaboratively created to extract structured information from Wikipedia. Since its inception in 2007, there have been regular data releases; it also has a live extraction component that processes Wikipedia updates real-time. DBpedia is a central hub in the Linking Open Data cloud, and has been widely used in various semantic search tasks [11, 95, 96, 135].²

We use the English part of the 2015-10 version of DBpedia. It contains 6.2 million entities, 1.1 billion facts, and an ontology of 739 types. In comparison, version 3.7, that has been used in *DBpedia-Entity v1*, contains 3.64 million entities, over 400 million facts, and an ontology of 358 types. DBpedia 2015-10 is also believed to be much cleaner due to better extraction techniques developed by the DBpedia community.

Preprocessing. We require entities to have both a title and abstract (i.e., `rdfs:label` and `rdfs:comment` predicates)—this effectively filters out category, redirect, and disambiguation pages. Note that list pages, on the other hand, are retained. In the end, we are left with a total of 4.6 million entities. Each entity is uniquely identified by its URI.

8.2 The Entity Retrieval Subcollection

The original DBpedia-Entity collection (v1) was developed for evaluating the entity retrieval task; it provided relevant entities in response to entity-bearing queries. In DBpedia-Entity v2, we updated the entity retrieval collection. The new version’s improvements are many-fold.

1. The original collection contains only relevant results and relevance is binary for most of the queries; we use graded relevance judgments for all queries and also include all judged items, relevant or not.
2. The DBpedia knowledge base has grown significantly over the past years. Many new relevant entities were not judged in the old version; we use a recent DBpedia version and judge the relevance of new entities.

²DBpedia is not the only general-purpose knowledge base available, but arguably the most suitable one. Alternatives include YAGO [179] (not updated regularly), Freebase (discontinued), and WikiData (still in its infancy).

3. Judgments in the original collection have been assembled from multiple campaigns, each with its own setup; we obtain relevance labels under the same conditions for all queries in the collection.

We also present details about how the DBpedia dump is processed and indexed, reducing the inconsistency in preprocessing. We provide rankings using both traditional and recently-developed entity search methods, making future comparison with prior work much easier.

8.2.1 Constructing the Collection

In *DBpedia-Entity v1*, the relevance judgments (“qrels”) are assembled from several different benchmarks. These assessments were created using different annotation guidelines, judges (trained assessors vs. crowdsourcing), pooling methods, and even different corpora (various versions of DBpedia or Wikipedia). For the updated collection, we generate new relevance judgments for all queries using the same setup. We pool candidate results from the same set of systems, and use the same annotation procedure and guidelines.

Pooling

Following standard practice of IR test collection building, we employ a *pooling* approach, and combine retrieval results from four main sources:

- **Original qrels.** All relevant entities from *DBpedia-Entity v1* are included, to ensure that results that have previously been identified as relevant get re-assessed.
- **Previous runs.** We consider 37 different retrieval methods (“runs”) that have been evaluated on *DBpedia-Entity v1* in prior work [93, 149, 200, 210]. All entity URIs returned by these runs are mapped to DBpedia version 2015-10; entities not present in DBpedia 2015-10 are discarded. The pool depth is 20, i.e., we take the top 20 ranked entities from each run.
- **New runs.** We obtained retrieval results for DBpedia 2015-10 from 13 different systems, by three independent research groups; see Section 8.2.2 for the description of these methods. Results are pooled from these runs up to depth 20.
- **SPARQL results.** For QALD-2 queries, the ground truth is obtained by executing a SPARQL query (manually constructed by the campaign

organizers [127]) over the knowledge base. We re-ran these queries against the DBpedia API endpoint to obtain up-to-date results, as the answers to some questions might have changed since (e.g., “Who is the mayor of Berlin?”).

The final assessment pool contains 50,516 query-entity pairs (104 entities per query on average).

Collecting Judgments

We collected relevance judgments using the CrowdFlower crowdsourcing platform. For each record (i.e., query-entity pair) in our pool, we provided the workers with the query, the name and short description (DBpedia abstract) of the entity, as well as the link to the entity’s Wikipedia page; see Figure 8.1. Since narratives are only available for a small number of queries in our query set (those from TREC and INEX), we decided to keep the setup uniform across all queries, and present assessors only with the query text. To avoid positional bias, records were presented in a random order. Workers were then asked to judge relevance on a 3-point Likert scale: highly relevant, relevant, or irrelevant. We educated workers about the notion of entities and provided them with the following working definitions for each scale (each further illustrated with examples):

- **Highly relevant (2):** The entity is a direct answer to the query (i.e., the entity should be among the top answers).
- **Relevant (1):** The entity helps one to find the answer to the query (i.e., the entity can be shown as an answer to the query, but not among the top results).
- **Irrelevant (0):** The entity has no relation to the intent of the query (i.e., the entity should not be shown as an answer).

We have taken quality control very seriously, which was a non-trivial task for a pool size of over 50K. During the course of the assessment, the accuracy of workers was regularly examined with hidden test questions. 400 query-entity pairs were randomly selected as test cases and judged by three authors of the paper; 373 of these were then used as test questions (where at least two of the experts agreed on the relevance label). Only workers with qualification level 2 (medium) or 3 (high) on CrowdFlower were allowed to participate. They were then required to maintain at least 70% accuracy throughout the job; those falling

Query:

Who founded Intel?

Entity:

Name: Gordon Moore

Wikipedia article: https://en.wikipedia.org/wiki/Gordon_Moore

Description: Gordon Earle Moore (born January 3, 1929) is an American businessman, co-founder and Chairman Emeritus of Intel Corporation, and the author of Moore's law. As of January 2015, his net worth is \$6.7 billion.

How relevant is this entity to the intent of the query?

☒ Highly relevant
☐ Relevant
☐ Irrelevant

Figure 8.1: Crowdsourcing task design.

below this threshold were not allowed to continue the job and their previous assessments were excluded. We collected 5 judgments for each record and paid workers a reasonable price of €1 per judgment. The final cost was over 3,500 USD, which makes this a very valuable test collection, also in the literal sense of the word. The Fleiss' Kappa inter-annotator agreement was 0.32, which is considered fair agreement. To determine the relevance level for a query-entity pair, we took the majority vote among the assessors. In case of a tie, the rounded average of relevance scores is taken as the final judgment.

Further inspection of the obtained results revealed that crowd workers are less likely to find answers to complex information needs. They are less patient and make judgments primarily based on the provided snippets and Wikipedia pages. When it would be required to read the Wikipedia article more carefully, or to consult additional sources, users are less likely to label them as attentively as expert annotators would. To further the quality of the test collections, we collected expert annotations for cases with “extreme disagreements,” i.e., cases without majority vote, or cases that are found irrelevant by crowd workers, but are highly relevant according to the original qrels.³ This resulted in the annotation of 8K query-entity pairs, each by two experts, with a Fleiss' Kappa

³This includes SPARQL query results for QALD queries, highly relevant judgments for SemSearch queries, and all TREC and INEX judgments.

Table 8.1: Query categories in the entity retrieval subcollection of DBpedia-Entity v2. R_1 and R_2 refer to the average number of relevant and highly relevant entities per query, respectively.

Category	#queries	Type	R_1	R_2
SemSearch ES	113	named entities	12.5	3.0
INEX-LD	99	keyword queries	23.5	9.2
ListSearch	115	list of entities	18.1	12.7
QALD-2	140	NL questions	28.4	29.8
Total	467		21.0	14.7

agreement of 0.48, which is considered moderate. The final label for the extreme disagreement cases was taken to be the expert-agreed label. If such a label did not exist, we took the rounded average between the two expert labels and the crowdsourcing decision (as a third label). Finally, queries that no longer have relevant results were removed (18 in total). Table 8.1 shows the statistics for the final *v2* collection.

8.2.2 Baseline Methods

In this section, we present a wide range of baseline entity retrieval methods. These methods were also used to obtain results for pooling.

Indexing and Query Processing

Retrieval results were obtained using two indices (Index A and Index B), built from the DBpedia 2015-10 dump, following the general approach outlined in [210]. In particular, we used the same entity representation scheme with five fields (*names*, *categories*, *similar entity names*, *attributes*, and *related entity names*) as in [210]. Index A was constructed using Galago, while Index B was created using Elasticsearch. They use slightly different methods for converting entity URIs to texts. Index B also contains an extra *catchall* field, concatenating the contents of all other fields. An extra URI-only index was built according to [93], which is used for the ELR-based methods; we write ‘+’ to denote when this index is used. All the runs were generated using preprocessed queries; i.e., removing the stop patterns provided in [93] and punctuation marks. Further details are provided in the collection’s GitHub repository.

Table 8.2: Comparison of methods for DBpedia-entity v1 vs. v2 qrels. The supervised methods (bottom block) are trained on v1.

Method	Index	v1		v2	
		MAP	P@10	MAP	nDCG@10
BM25	A	0.0884	0.0971	0.1893	0.3582
PRMS	B	0.1571	0.1682	0.2895	0.3905
MLM-all	B	0.1618	0.1705	0.3031	0.4021
LM	B	0.1709	0.1837	0.3144	0.4182
SDM	A	0.1860	0.1880	0.3259	0.4185
LTR	A	0.1723	0.1831	0.2446	0.3464
LM+ELR	B+	0.1772	0.1895	0.3103	0.4123
SDM+ELR	A+	0.1901	0.1986	0.3284	0.4200
MLM-CA	A	0.1905	0.2008	0.3061	0.4117
BM25-CA	A	0.2067	0.2056	0.3265	0.4231
FSDM	A	0.2069	0.2039	0.3279	0.4267
BM25F-CA	A	0.2088	0.2126	0.3361	0.4378
FSDM+ELR	A+	0.2210	0.2089	0.3295	0.4335

Retrieval Methods

We consider various entity retrieval methods that have been published over the recent years [12, 47, 93, 149, 210]. Unless stated otherwise, the parameters of methods are trained for each of the four query subsets, using cross-validation (with the same folds across all methods). Table 8.2 shows the particular index version that was used for each method.

Unstructured retrieval models. This group of methods uses a flattened entity representation. Specifically, we report on **LM** (Language Modeling) [156], **SDM** (Sequential Dependence Model) [139], and **BM25** [168]. All LM-based methods use Dirichlet prior smoothing with $\mu = 1500$ for index A, and $\mu = 2000$ for index B. The BM25 parameters are $k1 = 1.2$ and $b = 0.8$. We also report on **BM25-CA** with parameters trained using Coordinate Ascent.

Fielded retrieval models. This category of methods employs a fielded entity representation. We report on **MLM-CA** (Mixture of Language Models) [152], **FSDM** (Fielded Sequential Dependence Model) [210], and **BM25F-CA** [168]

Table 8.3: Results, broken down into query subtypes, on DBpedia-entity v2.

Model	SemSearch ES		INEX-LD		ListSearch		QALD-2		Total	
	@10	@100	@10	@100	@10	@100	@10	@100	@10	@100
BM25	0.2497	0.4110	0.1828	0.3612	0.0627	0.3302	0.2751	0.3366	0.2558	0.3582
PRMS	0.5340	0.6108	0.3590	0.4295	0.3684	0.4436	0.3151	0.4026	0.3905	0.4688
MLM-all	0.5528	0.6247	0.3752	0.4493	0.3712	0.4577	0.3249	0.4208	0.4021	0.4852
LM	0.5555	0.6475	0.3999	0.4745	0.3925	0.4723	0.3412	0.4338	0.4182	0.5036
SDM	0.5535	0.6672	0.4030	0.4911	0.3961	0.4900	0.3390	0.4274	0.4185	0.5143
LM+ELR	0.5554	0.6469	0.4040	0.4816	0.3992	0.4845	0.3491	0.4383	0.4230	0.5093
SDM+ELR	0.5548	0.6680	0.4104	0.4988	0.4123	0.4992	0.3446	0.4363	0.4261	0.5211
MLM-CA	0.6247	0.6854	0.4029	0.4796	0.4021	0.4786	0.3365	0.4301	0.4365	0.5143
BM25-CA	0.5858	0.6883	0.4120	0.5050	0.4220	0.5142	0.3566	0.4426	0.4399	0.5329
FSDM	0.6521	0.7220	0.4214	0.5043	0.4196	0.4952	0.3401	0.4358	0.4524	0.5342
BM25F-CA	0.6281	0.7200	0.4394	0.5296	0.4252	0.5106	0.3689	0.4614	0.4605	0.5505
FSDM+ELR	0.6563	0.7257	0.4354	0.5134	0.4220	0.4985	0.3468	0.4456	0.4590	0.5408

(the -CA suffixes refer to training using Coordinate Ascent). We also report on **MLM-all**, with equal field weights, and on **PRMS** (Probabilistic Model for Semistructured Data) [112], which has no free parameters.

Other models. The **LTR** (Learning to Rank) approach [47] employs 25 features from various retrieval models and is trained using the RankSVM algorithm. The **ELR** methods [93] employ TAGME [69] for annotating queries with entities, and use the URI-only index (with a single catchall field) for computing the ELR component.

8.2.3 Results and Analysis

In Table 8.2 we report on the performance of the different retrieval methods using both the original (*v1*) and new (*v2*) relevance judgments. (In case of the *v1* qrels, we removed entities that are not present in DBpedia 2015-10.) Methods in the top block of the table do not involve any training and use default parameter settings, while systems in the bottom block are trained for each query category using cross-validation. Training is done using the *v1* qrels. Since we have graded relevance judgments for *v2*, the “official” evaluation measure for the new collection is NDCG@10. However, to facilitate comparison with the *v1* results, we also report on MAP (at rank 100, accepting both levels 1 and 2 as relevant).

At first glance, we observe that the absolute MAP values for *v2* are higher than for *v1*; this is expected, as there are more relevant entities according to the new judgments. We also find that the relative ranking of methods in the top block remains the same when moving from *v1* to *v2*. On the other hand, methods that involve training (bottom block) show much smaller relative improvements over the models without training (top block) in *v2* as for *v1*. This is explained by the fact that training was done on *v1*. We note that we are not elaborating on the performance of individual methods as that is not the focus of this paper. One issue we wish to point out, nevertheless, is that default parameter settings may be a poor fit for entity retrieval; in particular, observe the large difference between BM25 with default parameters vs. BM25-CA with trained parameters (which are $b \approx 0.05$ and k_1 in the range 2..6, depending on the query subtype). In Table 8.3, we further report on the supervised models trained on the new (*v2*) qrels, and break evaluation results down into different query subsets. New retrieval systems, evaluated using DBpedia-Entity v2, are supposed to be compared against these results.

8.3 The Target Type Identification Subcollection

A characteristic property of entities is that they are typed, where types are typically organized in a hierarchical structure, i.e., a *type taxonomy*. The objective of *Target Type Identification (TTI)* task is to assign target types to queries from a type taxonomy [11, 76]. Formally:

Definition 8.1. *TTI is the task of finding the main target types of a query, from a type taxonomy, such that (i) these correspond to the most specific category of entities that are relevant to the query, and (ii) main types cannot be on the same path in the taxonomy. If no matching type can be found in the taxonomy then the query is assigned a special NIL-type.*

This definition allows for assigning multiple, one, or even no type to a query. Consider for example, the query “finland car industry manufacturer saab sisu,” where both *Company* and *Automobile* are valid types, and the query “Vietnam war facts,” which cannot be mapped to any type in the given taxonomy.

In DBpedia-Entity v2, we follow the above definition and develop a new subcollection for the TTI task. This extension to the DBpedia-Entity collection can be further used to improve the performance of entity retrieval methods, see, e.g., [15, 107, 154]. Below, we provide a detailed description of constructing this subcollection.

8.3.1 Constructing the Collection

We use the DBpedia Ontology (version 2015-10) as our type taxonomy, which is a manually curated and proper “is-a” hierarchy (unlike, e.g., Wikipedia categories).

Pooling

A *pool* of target entity types is constructed from four baseline methods, taking the top 10 types from each: entity-centric [11, 108, 186] and type-centric [11], using both BM25 and LM as retrieval methods. Additionally, we included all types returned by an *oracle* method, which has knowledge of the set of relevant entities for each query (from the ground truth). Specifically, the oracle score is computed as:

$$score_O(t, q) = \sum_{e \in Rel(q)} \mathbb{1}(e, t) \quad (8.1)$$

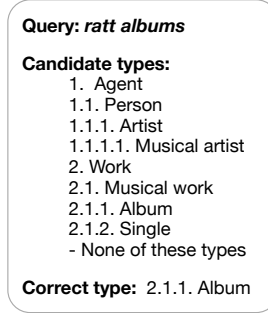


Figure 8.2: Example query from the crowdsourcing task description.

where $Rel(q)$ indicates the set of relevant entities for the query q . We employ this oracle to ensure that all reasonable types are considered when collecting human annotations.

Collecting Judgments

We obtained target type annotations via the CrowdFlower crowdsourcing platform. Specifically, crowd workers were presented with a search query (along with the narrative from the original topic definition, where available), and a list of candidate types, organized hierarchically according to the taxonomy. We asked them to “select the single most specific type, that can cover all results the query asks for” (in line with [11]). If none of the presented types are correct, they were instructed to select the “None of these types” (i.e., *NIL*-type) option. Figure 8.2 shows one of the example queries (alongside its candidate and correct types) that was given in the annotation instructions to crowd workers.

The annotation exercise was carried out in two phases. In the first phase, we sought to narrow down our pool to the most promising types for each query. Since the number of candidate types for certain queries was fairly large, they were broken down to multiple micro-tasks, such that for every top-level type, all its descendants were put in the same micro-task. Each query-type batch was annotated by 6 workers. In the second phase, all candidate types for a query were presented in a single micro-task; candidates include all types that were selected by at least one assessor in phase one, along with their ancestors up to the top level of the hierarchy. Each query was annotated by 7 workers.

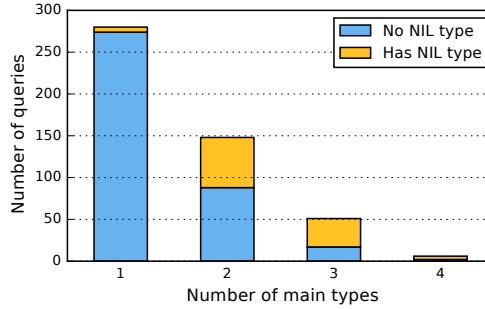


Figure 8.3: Distribution of the number of main target types.

The Fleiss’ Kappa inter-annotator agreement for this phase was 0.71, which is considered substantial.

Note that according to our task definition, the main target types of a query cannot lie on the same path in the taxonomy. To satisfy this condition, if two types were on the same path, we merged the more specific type into the more generic one (i.e., the more generic type received all the “votes” of the more specific one). This affected 120 queries. Figure 8.3 shows the distribution of queries according to the number of main types. 280 of all queries (57.73%) have a single target type, while the remainder of them have multiple target types. Notice that as the number of main types increases, so does the proportion of NIL-type annotations.

8.3.2 Baseline Methods

This section presents our baseline methods for the target type identification subcollection.

Entity-centric model. The entity-centric model [11, 76, 108, 186] can be regarded as the most common approach for determining the target types for a query. The idea is simple: first, rank entities based on their relevance to the query, then look at what types the top- K ranked entities have. The final score for a given type t is the aggregation of the relevance scores of entities with that type. We consider both Language Modeling (LM) and BM25 as the underlying entity retrieval model. For LM, we use Dirichlet prior smoothing

Table 8.4: Target type detection performance.

Method	NDCG@1	NDCG@5
EC, BM25 ($K = 20$)	0.1490	0.3223
EC, LM ($K = 20$)	0.1417	0.3161
TC, BM25	0.2015	0.3109
TC, LM	0.2341	0.3780
LTR	0.4842	0.6355

with the smoothing parameter set to 2000. For BM25, we use $k1 = 1.2$ and $b = 0.75$. The rank-cutoff threshold K is set empirically.

Type-centric model. The type-centric model [11, 76] considers for each type a direct term-based representation (pseudo type description document), by aggregating descriptions of entities of that type. Then, those type representations can be ranked much like documents. Specifically, the relevance score of a type for a given query is calculated as the sum of the individual query term scores, where the score is the underlying term-based retrieval model (e.g., LM or BM25).

LTR. The learning to rank approach, proposed in [76], combines entity- and type-centric scores. In addition, it leverages other signals that one could leverage, including taxonomy-driven features and type label similarities. Garigliotti et al. [76] employ the Random Forest algorithm, and set number of trees (iterations) to 1000, and the maximum number of features in each tree, m , to (the ceil of the) 10% of the size of the feature set.

8.3.3 Results and Analysis

Following [11], we report on NDCG at rank positions 1 and 5. The relevance level (“gain”) of a type is set to the number of assessors that selected that type. Detecting NIL-type queries is a separate problem on its own, which we are not addressing here. Therefore, the NIL-type labels are ignored in our experimental evaluation (affecting 104 queries). Queries that got only the NIL-type assigned to them are removed (6 queries in total). No re-normalization of the relevance levels for NIL-typed queries is performed (similar to the setting in [19]). For the LTR results, we used 5-fold cross-validation.

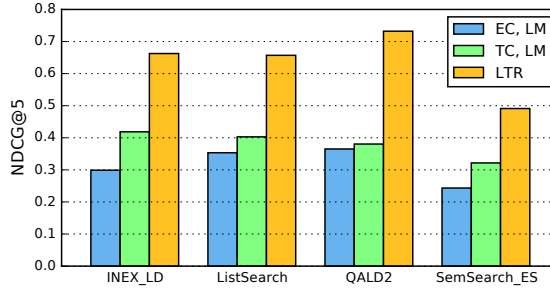


Figure 8.4: Performance across different query categories.

Table 8.4 presents the evaluation results. We find that the supervised learning approach significantly and substantially outperforms all baseline methods (relative improvement over 43% according to any measure, with $p < 0.001$ using a two-tailed paired T-test). In Figure 8.4, we break performance down into different query categories and observe that the LTR method outperforms other baselines, with the biggest improvements for QALD-2 queries.

8.4 Summary

In this chapter, we have presented two test collections based on the DBpedia-Entity test collection: (i) entity retrieval, (ii) target type identification collection. The entity retrieval subcollection uses a more recent DBpedia dump, a more consistent candidate document pool, and a unified relevance assessment procedure. The target type identification subcollection is a novel addition to the original DBpedia-Entity collection, and is based on DBpedia ontology as the type taxonomy. We have also provided details about processing and indexing, together with baseline results for both traditional and more recent entity retrieval and target type identification models. It is our hope that these new test collections will serve as the de facto testbed for these tasks, and will foster future research.

Chapter 9

Nordlys: A Semantic Search Toolkit

In the last technical chapter of this thesis, we introduce Nordlys, a toolkit for entity-oriented and semantic search. It provides functionality for entity cataloging, entity retrieval, entity linking in queries, and target type identification. There are two main reasons that motivated us to develop this toolkit. First, there exist a range of tools and demonstrators that address one specific task. Examples include GERBIL [184], TAGME [69], STICS [100], and Broccoli [18]. Second, repeatability and reproducibility of results is a fundamental requirement of scientific progress [5, 94, 124]. Having an open source and verifiable implementation of methods can foster research and development. In the area of entity-oriented search, despite recent advances, there is a lack of publicly available implementations of standard methods and techniques. With this work, we aim to fill that gap.

One of its distinguishing features of Nordlys is that it implements a number of traditional and state-of-the-art methods for a range of tasks: entity retrieval, entity linking in queries, identifying target types of queries, and entity cataloging. Another important characteristic is that it accommodates various usage needs on different levels:

- It is made available as an open source Python library that can be integrated into larger applications or can be used as a command line tool for research and experimentation. The code is organized in a three-tier architecture, cleanly separating the various layers of functionality.

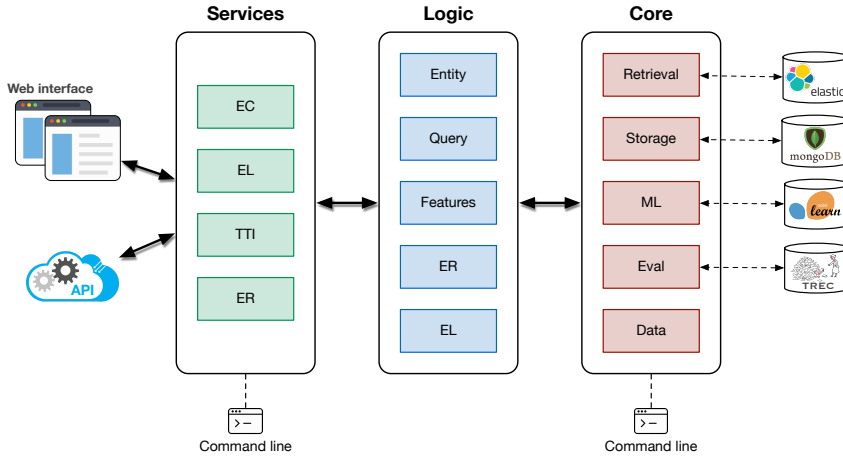


Figure 9.1: Nordlys architecture.

- It provides a RESTful API, through which Nordlys can be used as a service, much like a black box.
- The functionality is also available through a graphical web user interface. This interface can be used, e.g., to perform user studies on result presentation (similar to [33]).

In summary, Nordlys represents a major step towards reproducible and extensible semantic search research. It is meant to be a continuous effort, with additional methods included over time, as opposed to a one-time release of code. The toolkit is available at <http://nordlys.cc>.

9.1 Functionality

In this section we detail the functionality that is implemented in the Nordlys toolkit.

9.1.1 Entity Catalog

The main enabling component of semantic search, from a data perspective, is a *knowledge base*, which contains a collection of entities, their attributes, and

relations to other entities. It also contains information about entity types, which are often organized in a hierarchical structure (called *type taxonomy*). The *entity catalog* serves as a data access layer to the knowledge base. It is used to obtain information about entities, such as their IDs, names, attributes, and relationships. Additionally, it provides basic statistics about entities and properties; these statistics may be utilized, among others, for result presentation (e.g., identifying prominent properties when generating entity cards) [98].

The entity catalog may be seen as a data abstraction layer that makes the higher-level functionality largely independent of the particular data source. Nordlys is shipped with code and support for the DBpedia knowledge base, but data may be loaded into the entity catalog from any other knowledge base. Specifically, the entity catalog offers the following functionality:

- **Entity lookup.** Presents all entity properties from the knowledge base in a key-value format.
- **ID mapping.** Provides a one-to-one mapping of entity IDs across various knowledge bases via same-as links (DBpedia and Freebase).
- **Surface form lookup.** Presents all entities that match an entity surface form, obtained from the FACC collection for the ClueWeb09 and ClueWeb12 datasets [75].
- **Basic statistics.** A set of statistics are computed over contents of the knowledge base, including RDF types, predicates, and properties.

9.1.2 Entity Retrieval

Entity retrieval is a core building block of semantic search, where a ranked list of entities are presented in response to an entity-bearing query. Formally, it is defined as follows.

DEFINITION 7 (Entity retrieval): *Given a query q , entity retrieval is the task of returning a ranked list of entities (e_1, \dots, e_k) from an underlying knowledge base KB , ordered according to their relevance to the query q .*

The toolkit implements both standard baselines and more recent approaches:

- **BM25.** The BM25 model, as implemented in Elasticsearch, which is the most efficient retrieval model of our toolkit.

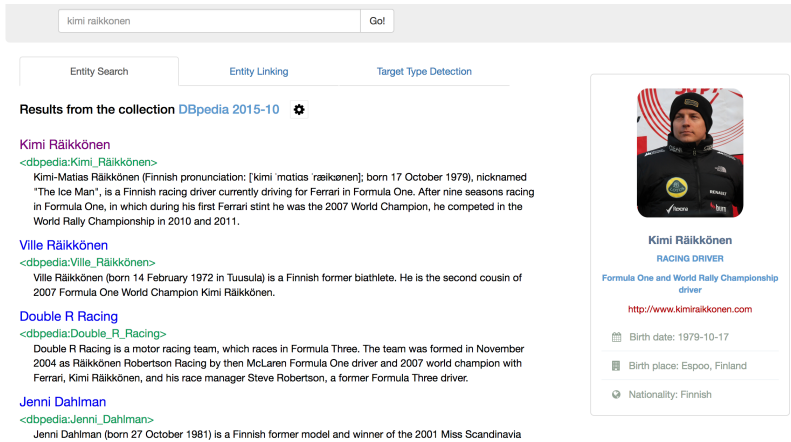


Figure 9.2: The Nordlys web user interface.

- **LM.** Language Modeling [156] approach (with Dirichlet prior and Jelinek-Mercer smoothing), which employs a single field representation of entities.
- **MLM.** The Mixture of Language Models [152], which represents entities as structured (fielded) documents, using a linear combination of language models built for each field. For DBpedia, we use a five-field representation (cf. Section 8.2.2).
- **PRMS.** The Probabilistic Model for Semistructured Data [111], which uses collection statistics to compute field weights in for MLM model (thereby making it parameter-free).
- **ELR.** The Entity Linking incorporated Retrieval model is a state-of-the-art approach, which extends text-based retrieval models by considering entities mentioned in the query (cf. Chapter 6).

The online repository includes retrieval performance measurements on the DBpedia-entity test collection [12, 98], which is a standard benchmark for entity retrieval.

9.1.3 Entity Linking in Queries

Identifying named entities in queries and linking them to the corresponding entry in the knowledge base is known as the task of *entity linking in queries*

(ELQ) [92]. Various demonstrations have been released for general purpose entity linking [45, 69, 138, 138, 184]. The most popular among them is TAGME [69], which has been designed for short texts such as tweets, search snippets. TAGME, however, does not handle the ambiguity of search queries and an entity mention is only linked to a single entity. The entity linking functionality in Nordlys aims to bridge this gap, by implementing both baselines and state-of-the-art approaches for entity linking in queries. The task is formally defined as follows.

DEFINITION 8 (Entity linking in queries): *Given a query q , entity linking in queries returns one or multiple interpretations of the query, A_1, \dots, A_n . Each interpretation consists of a set of entity linking decisions, i.e., mention-entity pairs: $A_i = \{(m_1, e_1), \dots, (m_k, e_k)\}$, where mention m_j is a query substring that refers to entity e_j in the knowledge base.*

The following methods are implemented in Nordlys:

- **CMNS.** The baseline method that performs entity linking based on the overall popularity of entities as link targets, i.e., the *commonness* feature [134]. This method has been introduced in [136], and has been widely used in later research, e.g., [29, 92, 199].
- **MLM-greedy.** An efficient generative retrieval model presented (cf. Chapter 3), that combines the commonness score with the textual similarity between the query and the entity.
- **LTR-greedy.** The recommended method (with respect to both efficiency and effectiveness) in Chapter 5, which employs a learning to rank model with various textual and semantic similarity features.

9.1.4 Target Type Identification

Target type detection is one specific form of query understanding, where the aim is to assign target types (or categories) to queries from some type taxonomy [11, 76]. Formally:

DEFINITION 9 (Target Type Identification): *Given a query q , the aim of target type identification task is to return a ranked list of entity types (t_1, \dots, t_k) , ordered with respect to their likelihood of being the target type of query q .*

We implement three different approaches from the literature [11, 76]:

- **Entity-centric.** This method first ranks entities based on their relevance to the query, then looks at what types the top- k ranked entities have. The final score for a given type t is the aggregation of the relevance scores of entities with that type [11].
- **Type-centric.** This approach builds, for each type, a direct term-based representation (pseudo type description document), by aggregating descriptions of entities of that type. Then, those type representations can be ranked much like documents [11].
- **Learning to rank.** A supervised learning approach, which considers a rich set of features such as distributional similarity, type label similarities, and taxonomy-driven features [76].

These methods has been evaluated using a publicly available test collection; the test collection and results are presented in [76].

9.2 The Nordlys Toolkit

In this section, we present the overall architecture and the various ways in which our toolkit may be used. We ship our code with a small sample taken from DBpedia, and include scripts for processing and indexing DBpedia and Freebase. We also make data-dumps available for download (entity catalog and search indices).

9.2.1 Architecture

Nordlys is based on a multitier architecture with three layers: *core* (data tier), *logic*, and *services* (presentation tier); see Figure 9.1.

Core. The *core* layer provides basic functionality, including retrieval (Elasticsearch), storage (MongoDB key-value store), machine learning (scikit-learn), and evaluation (trec_eval). In parentheses are the third-party tools we currently use. It is possible to connect additional external tools (or replace our default choices) by implementing standard interfaces of the respective core modules. Additionally, a separate data module is provided with functionality for loading and preprocessing standard data sets (DBpedia, Freebase, ClueWeb, etc.). The core layer represents a versatile general-purpose modern IR library, which may also be accessed using command line tools.

Logic. This layer contains the main business logic, which is organized around five main modules: (1) *entity* provides access to the entity catalog (including knowledge bases and entity surface form dictionaries); (2) *query* provides the representation of search queries along with various preprocessing methods; (3) *features* is a collection of entity-related features, which may be used across different search tasks; (4) *entity retrieval* (ER) contains various entity ranking methods, cf. Section 9.1.2; (5) *entity linking* (EL) implements entity linking functionality, cf. Section 9.1.3. The logic layer may not be accessed directly.

Services. The *services* layer provides end-user access to the toolkit’s functionality, through the command line, RESTful API, and web interface. Four main services are available: entity retrieval (ER), entity linking (EL), target type identification (TTI), and entity catalog (EC); these we have already explained in Section 9.1.

9.2.2 Usage Modes

Nordlys may be used as a Python library, RESTful API, command line tool, or a web interface.

Python Library. We base our toolkit on Python 3.5+, specifically, on the Anaconda distribution. The code is complemented with an automatically generated documentation as well as an extensive how-to. Upon downloading our toolkit, installing the prerequisites, and loading the necessary data, it should be possible to run Nordlys on any machine.

Command line. The main functionality may be accessed through a set of command line tools. This can be useful, e.g., for processing larger amounts of data and eliminating the network overhead.

RESTful API. We provide a public API endpoint (subject to registering for an API key). Figure 9.3 shows an excerpt from the response received from the API for an entity retrieval request. For convenience, the parameters of the methods are set to reasonable defaults; these values may be changed when calling the services.

Web Interface. Figure 9.2 shows an excerpt from our web user interface. The implementation is based on Flask and Bootstrap. The default settings of

```
http://api.nordlys.cc/er?key=xx&q=total+recall&model=lm
```

```
{ "query": "total recall",  
  "total_hits": 1000,  
  "results": {  
    "0": {  
      "entity": "<dbpedia:Total_Recall_(1990_film)>",  
      "score": -10.042525028471253  
    },  
    "1": {  
      "entity": "<dbpedia:Total_Recall_(2012_film)>",  
      "score": -10.295316626850521  
    },  
    ...  
  }  
}
```

Figure 9.3: Example call to the Nordlys API entity retrieval service.

the web interface are similar to that of the API endpoint, any may be changed on the interface.

9.3 Summary

We have introduced Nordlys, a toolkit that implements a range of methods for entity-oriented and semantic search. It is available as a Python library, as a command line tool, a RESTful API, and as a graphical web user interface. Nordlys is meant to be a continuous effort; we plan to include additional methods, e.g., entity summarization. We are also working on optimizations, to improve the efficiency of our methods.

Chapter 10

Conclusions

In this chapter, we will first summarize our main findings and draw conclusions by answering our research questions. We then give an outlook on future work related to query understanding, entity retrieval, entity summarization, and semantic search in general.

10.1 Main Findings

Here, we summarize the methods and our findings. We do so by answering the research questions raised in Chapter 1.

Query Understanding

In the first part of this thesis (Chapters 3-5), we focused on understanding queries by identifying entity mentions in queries and linking them to the corresponding entries in a reference knowledge base, referred to as the task of *entity linking in queries*. In Chapter 3 we asked our first research question:

RQ1 How can the inherent ambiguity of entity annotations in queries be handled and evaluated?

To answer this question, we discussed different ways to deal with query ambiguities and established entity linking in queries as the task of finding entity linking interpretation(s), where an interpretation is a set of non-overlapping entities that are semantically related to each other. If the query is ambiguous,

with little or no context, there exist multiple interpretations, all of which should be found. Otherwise, a single interpretation should be detected, which is similar to the traditional entity linking task for documents. Determining when the query has no interpretations (in terms of entity annotations) is also a crucial part of the problem that should be addressed (and considered in the evaluation). The semantic linking task, which ranks entities based on their relevance to the query, serves a different purpose and should not be considered as an entity linking task, even for the simplified scenario of finding a single interpretation. This is because relevant entities can be overlapping and are not required to be semantically related to each other. Furthermore, entity disambiguation is an essential part of entity linking, an aspect that is completely ignored in semantic linking. A number of earlier studies refer to entity linking, while what they do in fact is semantic linking. Comparing semantic linking to results generated by traditional entity linking methods is inappropriate.

For entity linking in queries, similar to the traditional entity linking task [69, 141, 143], evaluation uses set-based measures (precision, recall and F-measure). However, since the output is a set of interpretations (and not entities) the evaluation methodology is different. The method presented in [43] considers the exact match between the retrieved sets and the ground truth, which is rather strict. The lenient evaluation method (Section 3.4.3), on the other hand, combines interpretation-based and entity-based evaluations. For semantic linking, standard rank-based measures (MAP, MRR, P@1) can be employed.

We then moved on to identifying a baseline for entity linking in queries:

RQ2 How does a state-of-the-art entity linking approach perform on the ELQ task?

We selected TAGME [69] as a state-of-the-art entity linking system and systematically examined its repeatability, reproducibility, and generalizability. Our experiments showed that some of the TAGME results are not reproducible, even with the API provided by the authors. For the rest of the results, we found that (i) the results reported in their paper are higher than what can be reproduced using their API, and (ii) the TAGME API gives higher numbers than what is achievable by a third-party implementation (both ours and that of Dexter's [45]) (iii) the TAGME approach can be generalized to the ELQ task. Based on our findings, we conclude that the TAGME approach can be used as a baseline for the task of entity linking in queries.

After establishing the task, evaluation measures, test collection, and baseline, we focused on the solving of the task. We asked:

RQ3 How should entity linking in queries be performed to achieve high efficiency and effectiveness?

To answer the above research question, we set up a framework where different candidate entity ranking and disambiguation methods can be plugged in. For each of these components, we experimented with both unsupervised and supervised alternatives, resulting in a total of four different ELQ systems. Supervised methods are expected to yield high effectiveness coupled with lower efficiency, while for unsupervised approaches it is the other way around.

Our results revealed that candidate entity ranking is of higher importance than disambiguation for ELQ. Hence, it is more beneficial to perform the (expensive) supervised learning early on in the pipeline for the seemingly easier CER step; disambiguation can then be tackled successfully with an unsupervised (GIF) algorithm. We note that selecting the top ranked entity does not yield an immediate solution; as shown in Chapter 3, disambiguation is an indispensable step in ELQ.

We also found that contextual similarity features are the most effective for entity disambiguation. This is based on two observations: (i) the unsupervised (GIF) method takes only the entity ranking scores as input, which are computed based on the contextual similarity between entity and query; (ii) the supervised (LTR) method relies the most on query-dependent features. This is an interesting finding, as it stands in contrast to the common postulation in entity linking in documents that interdependence between entities help to better disambiguate entities. Entity interdependence features (and, in general, collective disambiguation methods) are more helpful when sufficiently many entities are mentioned in the text; this is not the case for queries.

Entity Retrieval

In the second part of the thesis, we focused on entity retrieval: answering search queries by returning a ranked list of entities. We asked:

RQ4 How to exploit entity annotations of queries in entity retrieval?

To address this question, we introduced a general framework for leveraging entity annotations of queries into the term-based models. Our framework is based on the Markov Random Field (MRF) model [139]. Within this framework, we introduced a new component for matching the linked entities from the query. This component, termed ELR (for Entity Linking incorporated Retrieval), may be seen as an extension that can be applied on top of any text-based retrieval

model that can be instantiated in the MRF framework. We applied our approach as an extension to various state-of-the-art entity retrieval models and showed significant and consistent improvements over all of them. We also showed that different query types are impacted differently by the ELR method. The results confirm our hypothesis that ELR can improve complex (ListSearch and QALD-2) as well as heterogeneous (INEX-LD) query sets, which involve entity relationships. On the other hand, short keyword queries, referring to a single, albeit often ambiguous, entity (SemSearch ES) are mostly unaffected.

We compared retrieval results using default and trained parameters. We found that the results are robust, i.e., ELR can improve the performance of term-based models, even with default parameter values. We also observed that ELR is robust with respect to entity linking; considering more entity annotations, even those with low confidence, improves retrieval performance.

Entity Summarization

In the third part of this thesis, we moved towards presentation aspects of the results. Specifically, we focused on generating content for entity cards, which are being used frequently in modern web search engines. We asked:

RQ5 How to generate and evaluate factual summaries for entity cards?

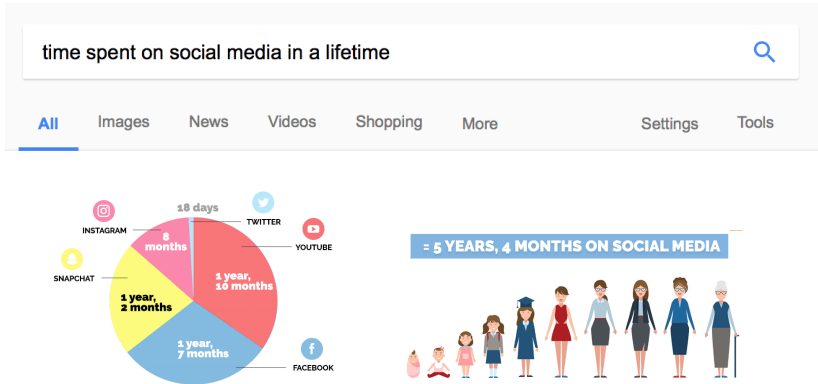
To address this question, we presented a method for fact ranking that takes both importance and relevance into consideration. We designed several novel features for capturing and distinguishing between importance and relevance, and combine these features in a supervised learning framework. We further introduced summary generation as a task on its own account, and developed an algorithm for producing a summary that meets the presentation requirements of an entity card. Concerning evaluation, we built a benchmark for the fact ranking task, obtaining a large number of crowdsourced judgments with respect to both fact importance and relevance. In addition, we evaluated the generated summaries, with regard to search queries, by performing user preference experiments via crowdsourcing. The results showed that our proposed fact ranking approach significantly outperforms existing baseline systems. We also found that the summaries uniting both fact importance and relevance are preferred over those that are based on a single aspect. Overall, our results confirmed the hypothesis that dynamic (query-aware) summaries are preferred over static (query-agnostic) ones; this is especially true for complex relational queries.

10.2 Future Directions

We believe that utilizing knowledge bases for semantic search systems will continue to remain an important research area. With the recent success of commercial conversational systems such as Apple Siri, Google Now, and Amazon Alexa, there is an increasing demand for understanding natural language requests and responding to those with direct answers [105, 160]. In addition, the presentation of the results in search engine result pages is evolving, with increasingly more interactive result panels taking the place of the ranked list of documents. In the followings, we present the most prominent future directions related to our research: query understanding, answer retrieval, and result presentation.

Query understanding. A large body of our research has been focused on annotating queries with entities, i.e., the task of entity linking in queries. One obvious follow-up direction is to improve ELQ systems, specifically developing more effective and efficient approaches for entity ranking. Another direction is to further the understanding of queries by annotating them not only with entities, but also with entity attributes; e.g., annotating “nationality” in the query “einstein nationality” with the `dbo:citizenship` predicate from the knowledge base. Understanding the whole query intent and identifying target entity types [11, 76] is another direction in this area. Finally, generating clarification questions for ambiguous queries is another aspect of query understanding in both textual and conversational search systems.

Answer retrieval. In this thesis, we have found that leveraging entity annotations of the queries into the retrieval model can improve entity retrieval performance, especially for complex natural language queries. Our ELR model falls in the family of explicit semantic representation approaches. Another possibility would be to employ neural networks and learn latent representations of words, entities, and their relations in the knowledge base [187, 200]. These latent representations can be learnt in an end-to-end ranking model or in an unsupervised manner to be subsequently leveraged in a ranking model. So far, we have focused on answering queries with a ranked list of entities. Answers, however, can be made even more focused, for instance, returning a single entity, a specific attribute of an entity, or a set of entities along with their attributes, organized in a table. Developing retrieval models that can return such answers is an open research direction, which involves both query understanding and result ranking.



source:

<https://www.socialmediatoday.com/marketing/how-much-time-do-people-spend-social-media-infographic>

Figure 10.1: A mock-up illustration on how a future search engine can make use of plots and interactive panels to provide direct answers.

Result presentation. In Chapter 7, we have discussed the importance of entity cards in current search engine interfaces and focused on generating entity summaries. Our study has assumed summaries of fixed size, while in practice the size of these summaries may vary depending on the device, entity, and query; generating summaries of different size is a potential extension in this area. Other elements of entity cards include images, entity type, and related entities. Providing content for each of these elements is a research question on its own: (i) How to create thumbnail images for entity cards (following Kennedy et al. [109] and Bota et al. [33])?, (ii) How to identify the most important type of the entity (i.e., ranking type-like relations [20, 90])?, and (iii) How to make related entity recommendation serendipitous (see, e.g., Bordino et al. [31]).

Apart from entity cards, there are other means of providing concise answers to entity-bearing queries. Consider for example the query “time spent on social media in a lifetime,” where the answers can be presented by a plot or a table rather than textual descriptions; see Figure 10.1 for an illustration. Determining how a query is best answered, and generating interactive information panels as such tables, plots or maps are challenging tasks that lie ahead of search engines.

Appendix A

Resources

In the course of this thesis, we considered the reproducibility of our research seriously, and made publicly available a collection of resources developed within our studies. This includes source code, test collections, and run files related to Chapters 3–7.

A.1 Query Understanding via Entity Linking

The resources related to Chapter 3 of this thesis are available on <http://bit.ly/ictir2015-elq>. These include test collections (YSQLE, ERD-dev, and Y-ERD), all run files, evaluation scripts, and the source of Greedy Interpretation Finding (GIF) algorithm presented in the chapter.

A.2 Establishing a Baseline for Entity Linking in Queries

We released the resources related to the TAGME reproducibility study (Chapter 4) at <http://bit.ly/tagme-rep>. It includes our implementation of the TAGME system, together with run files, and scripts for computing evaluation measures. We also provide a set of data files, including the TAGME test collections (complemented with numerical IDs) and files related to evaluating TAGME on the ELQ task. The Wikipedia indices and surface form dictionary are also available for download at <http://iai.group/downloads/tagme-rep/>.

A.3 Methods for Entity Linking in Queries

The resources developed for the experiments of Chapter 5 are provided at <http://bit.ly/ecir2017-elq>. The repository contains the implementation of the “candidate entity ranking” and “disambiguation” methods. It also contains the evaluation scripts, run files, and test collections.

A.4 Exploiting Entity Linking in Queries for Entity Retrieval

The repository <http://bit.ly/ictir2016-elr> contains the resources related to Chapter 6 of this thesis. We provide the source code required for running the entity retrieval methods, together with the query set and qrels files for the DBpedia-Entity test collection (version 3.9). The indices required for running the code are also available at <http://iai.group/downloads/elr/>.

A.5 Dynamic Factual Summaries for Entity Cards

The resources related to Chapter 7 of the thesis are made available at <http://tiny.cc/sigir2017-dynes>. The repository contains the crowdsourcing experiment designs, query-entity pairs selected from the DBpedia-Entity collection, the fact ranking collection, and the run files.

References

- [1] Yahoo! Webscope L24 dataset - Yahoo! search query log to entities, v1.0. URL <http://webscope.sandbox.yahoo.com/>.
- [2] Amit Singhal. Introducing the Knowledge Graph: Things, not Strings. <https://googleblog.blogspot.nl/2012/05/introducing-knowledge-graph-things-not.html>, 2012.
- [3] Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch, and Richard Cyganiak. Linking Open Data cloud diagram 2017. <http://lod-cloud.net/>, 2017.
- [4] Areej Alasiry, Mark Levene, and Alexandra Poulouvassilis. Detecting candidate named entities in search queries. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 1049–1050, 2012.
- [5] Jaime Arguello, Matt Crane, Fernando Diaz, Jimmy Lin, and Andrew Trotman. Report on the SIGIR 2015 workshop on reproducibility, inexplicability, and generalizability of results (RIGOR). *SIGIR Forum*, 49(2): 107–116, 2016.
- [6] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a Web of Open Data. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 722–735, 2007.
- [7] Steve Austin, Richard Schwartz, and Paul Placeway. The forward-backward search algorithm. In *Proceedings of International Conference*

- on Acoustics, Speech, and Signal Processing*, ICASSP '91, pages 697–700. IEEE, 1991.
- [8] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search*, Second edition. Pearson Education Ltd., Harlow, England, 2011.
- [9] Krisztian Balog. Semistructured data search. In *Bridging Between Information Retrieval and Databases*, volume 8173 of *Lecture Notes in Computer Science*, pages 74–96. 2014.
- [10] Krisztian Balog. *Encyclopedia of Database Systems*, chapter Entity Retrieval, pages 1–6. Springer New York, 2017.
- [11] Krisztian Balog and Robert Neumayer. Hierarchical target type identification for entity-oriented queries. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2391–2394, 2012.
- [12] Krisztian Balog and Robert Neumayer. A test collection for entity search in DBpedia. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 737–740, 2013.
- [13] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 43–50, 2006.
- [14] Krisztian Balog, Pavel Serdyukov, Arjen De Vries, Paul Thomas, and Thijs Westerveld. Overview of the TREC 2009 entity track. In *Proceedings of the Eighteenth Text REtrieval Conference (TREC 2009)*, 2010.
- [15] Krisztian Balog, Marc Bron, and Maarten De Rijke. Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.*, 29(4):22:1–22:31, 2011.
- [16] Krisztian Balog, Pavel Serdyukov, and Arjen De Vries. Overview of the TREC 2011 entity track. In *Proceedings of the Twentieth Text REtrieval Conference (TREC 2011)*, 2012.

-
- [17] Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, IJCAI'07, pages 2670–2676, 2007.
 - [18] Hannah Bast, Florian Baurle, Bjorn Buchhold, and Elmar Haumann. Semantic full-text search with Broccoli. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 1265–1266, 2014.
 - [19] Hannah Bast, Björn Buchhold, and Elmar Haussmann. Relevance scores for triples from type-like relations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 243–252, 2015.
 - [20] Hannah Bast, Buchhold Björn, and Elmar Haussmann. Semantic search on text and knowledge bases. *Found. Trends Inf. Retr.*, 10(2-3):119–271, 2016.
 - [21] Michael Bendersky, Donald Metzler, and W. Bruce Croft. Learning concept importance using a weighted dependence model. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 31–40, 2010.
 - [22] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, pages 1533–1544, 2013.
 - [23] Shane Bergsma and Qin Iris Wang. Learning noun phrase query segmentation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '07, pages 819–826, 2007.
 - [24] Tim Berners-Lee, James Hendler, Ora Lassila, and Others. The Semantic Web. *Scientific american*, 284(5):28–37, 2001.
 - [25] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
 - [26] Roi Blanco, Harry Halpin, Daniel M Herzig, Peter Mika, Jeffrey Pound, and Henry S Thompson. Entity search evaluation over structured Web

- data. In *Proceedings of the 1st International Workshop on Entity-Oriented Search*, pages 65–71, 2011.
- [27] Roi Blanco, Peter Mika, and Sebastiano Vigna. Effective and efficient entity search in RDF data. In *Proceedings of the 10th International Conference on The Semantic Web - Volume Part I*, ISWC’11, pages 83–97, 2011.
- [28] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. Entity recommendations in Web search. In *Proceedings of the 12th International Semantic Web Conference - Part II*, ISWC ’13, pages 33–48, 2013.
- [29] Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. Fast and space-efficient entity linking for queries. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM ’15, pages 179–188, 2015.
- [30] 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*, SIGMOD ’08, pages 1247–1250, 2008.
- [31] Ilaria Bordino, Yelena Mejova, and Mounia Lalmas. Penguins in sweaters, or serendipitous entity search on user-generated content. In *Proceedings of the 22Nd ACM International Conference on Information and Knowledge Management*, CIKM ’13, pages 109–118, 2013.
- [32] Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 152–160, 1998.
- [33] Horatiu Bota, Ke Zhou, and Joemon M Jose. Playing your cards right: The effect of entity cards on search behaviour and workload. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, CHIIR ’16, pages 131–140, 2016.
- [34] Wladimir C Brandão, Rodrygo L T Santos, Nivio Ziviani, Edleno S de Moura, and Altigran S da Silva. Learning to expand queries using entities. *JASIST*, 65(9):1870–1883, 2014.

-
- [35] Liora Braunstein, Oren Kurland, David Carmel, Idan Szpektor, and Anna Shtok. Supporting human answers for advice-seeking questions in CQA sites. In *Proceedings of the 38th European conference on Advances in Information Retrieval*, pages 129–141, 2016.
 - [36] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.
 - [37] Marc Bron, Krisztian Balog, and Maarten de Rijke. Ranking related entities: Components and analyses. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1079–1088, 2010.
 - [38] Razvan C Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, ACL '06, pages 9–16, 2006.
 - [39] 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*, ICML '05, pages 89–96, 2005.
 - [40] Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. Context-aware query classification. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 3–10, 2009.
 - [41] 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*, ICML '07, pages 129–136, 2007.
 - [42] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr., and Tom M Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI '10, pages 1306–1313, 2010.
 - [43] David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. ERD'14: Entity recognition and disambiguation challenge. In *ACM SIGIR Forum*, volume 48, pages 63–77, 2014.

-
- [44] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50, 2012.
 - [45] Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. Dexter: An open source framework for entity linking. In *Proceedings of the Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval*, ESAIR '13, pages 17–20, 2013.
 - [46] 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 and Knowledge Management*, CIKM '13, pages 139–148, 2013.
 - [47] Jing Chen, Chenyan Xiong, and Jamie Callan. An empirical study of learning to rank for entity search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 737–740, 2016.
 - [48] Gong Cheng, Thanh Tran, and Yuzhong Qu. RELIN: Relatedness and informativeness-based centrality for entity summarization. In *Proceedings of 10th International Semantic Web Conference*, ISWC '11, pages 114–129, 2011.
 - [49] Yen-Pin Chiu, Yong-Siang Shih, Yang-Yin Lee, Chih-Chieh Shao, Ming-Lun Cai, Sheng-Lun Wei, and Hsin-Hsi Chen. NTUNLP approaches to recognizing and disambiguating entities in long and short text at the ERD challenge 2014. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, ERD '14, pages 3–12, 2014.
 - [50] Kevyn Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *Proceedings of the 18th ACM International Conference on Information and Knowledge Management*, CIKM '09, pages 837–846, 2009.
 - [51] Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 249–260, 2013.

-
- [52] Marco Cornolti, Dipartimento Informatica, Massimiliano Ciaramita, and Stefan Rüd. The SMAPH system for query entity recognition and disambiguation. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, ERD '14, pages 1–6, 2014.
 - [53] Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Stefan Rüd, and Hinrich Schütze. A piggyback system for joint entity mention detection and linking in Web queries. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 567–578, 2016.
 - [54] W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines - Information Retrieval in Practice*. Pearson Education, 2009.
 - [55] W. Bruce Croft, Michael Bendersky, Hang Li, and Gu Xu. Query representation and understanding workshop. *SIGIR Forum*, 44(2):48–53, 2011.
 - [56] Silviu Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '07, pages 708–716, 2007.
 - [57] Silviu Cucerzan and Eric Brill. Spelling correction as an iterative process that exploits the collective knowledge of Web users. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, EMNLP '04, pages 293–300, 2004.
 - [58] Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 365–374, 2014.
 - [59] Arjen P. de Vries, Anne-Marie Vercoustre, James A. Thom, Nick Craswell, and Mounia Lalmas. Overview of the INEX 2007 entity ranking track. In *Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2007)*, pages 245–251, 2008.
 - [60] Gianluca Demartini, Tereza Iofciu, and ArjenP. de Vries. Overview of the INEX 2009 entity ranking track. In *Focused Retrieval and Evaluation, 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX 2009)*, pages 254–264, 2010.

-
- [61] Omkar Deshpande, Digvijay S Lamba, Michel Tourn, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Building, maintaining, and using knowledge bases: A report from the trenches. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 1209–1220, 2013.
 - [62] Andrea Dessi and Maurizio Atzori. A machine-learning approach to ranking RDF properties. *Future Gener. Comput. Syst.*, 54(C):366–377, 2016.
 - [63] Xin Dong, Evgeniy Gabrilovich, Jeremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A Web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601–610, 2014.
 - [64] Alan Eckhardt, Juraj Hreško, Jan Procházka, and Otakar Smrž. Entity linking based on the co-occurrence graph and entity probability. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, ERD '14, pages 37–44, 2014.
 - [65] Joe Ellis, Jeremy Getman, and Stephanie M Strassel. Overview of linguistic resources for the TAC KBP 2014 evaluations: Planning, execution, and results. In *Proceedings of TAC KBP 2014 Workshop, National Institute of Standards and Technology*, pages 17–18, 2014.
 - [66] Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. Overview of linguistic resources for the TAC KBP 2015 evaluations: Methodologies and results. In *Proceedings of TAC KBP 2015 Workshop, National Institute of Standards and Technology*, pages 16–17, 2015.
 - [67] Faezeh Ensan and Ebrahim Bagheri. Document retrieval model through semantic linking. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 181–190, 2017.
 - [68] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. Introducing Wikidata to the linked data Web. In *Proceedings of 13th International Semantic Web Conference*, ISWC '14, pages 50–65, 2014.
 - [69] Paolo Ferragina and Ugo Scaiella. TAGME: On-the-fly annotation of short text fragments (by Wikipedia entities). In *Proceedings of the 19th ACM*

- International Conference on Information and Knowledge Management*, CIKM '10, pages 1625–1628, 2010.
- [70] Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with Wikipedia pages. *CoRR*, abs/1006.3498, 2010.
- [71] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, ACL '05, pages 363–370, 2005.
- [72] John Foley, Brendan O'Connor, and James Allan. Improving entity ranking for keyword queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2061–2064, 2016.
- [73] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4: 933–969, 2003.
- [74] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29:1189–1232, 2001.
- [75] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0), 2013. URL <http://lemurproject.org/clueweb12/>.
- [76] Darío Garigliotti, Faegheh Hasibi, and Krisztian Balog. Target type identification for entity-bearing queries. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 845–848, 2017.
- [77] David Graus, Daan Odijk, Manos Tsagkias, Wouter Weerkamp, and Maarten de Rijke. Semanticizing search engine queries: The University of Amsterdam at the ERD 2014 challenge. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, ERD '14, pages 69–74, 2014.
- [78] David Graus, Manos Tsagkias, Wouter Weerkamp, Edgar Meij, and Maarten de Rijke. Dynamic collective entity representations for entity ranking. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 595–604, 2016.

-
- [79] Kalpa Gunaratna, Krishnaprasad Thirunarayan, and Amit P Sheth. FACES: diversity-aware entity summarization using incremental hierarchical conceptual clustering. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 116–122, 2015.
 - [80] Kalpa Gunaratna, Krishnaprasad Thirunarayan, Amit Sheth, and Gong Cheng. Gleaning types for literals in RDF triples with application to entity summarization. In *Proceedings of the 13th International Conference on The Semantic Web*, ESWC '16, pages 85–100, 2016.
 - [81] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 267–274, 2009.
 - [82] Stephen Guo, Ming-Wei Chang, and Emre Kiciman. To link or not to link? a study on end-to-end tweet entity linking. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL, pages 1020–1030, 2013.
 - [83] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R Curran. Evaluating entity linking with Wikipedia. *Artif. Intell.*, 194:130–150, 2013.
 - [84] Matthias Hagen, Martin Potthast, Benno Stein, and Christof Bräutigam. Query segmentation revisited. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 97–106, 2011.
 - [85] Harry Halpin, Daniel M Herzig, Peter Mika, Roi Blanco, Jeffrey Pound, Henry S Thompson, and Duc Thanh Tran. Evaluating ad-hoc object retrieval. In *Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010)*, 2010.
 - [86] Xianpei Han and Jun Zhao. Named entity disambiguation by leveraging Wikipedia semantic knowledge. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 215–224, 2009.
 - [87] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in Web text: A graph-based method. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 765–774, 2011.

-
- [88] Faegheh Hasibi. Indexing and querying overlapping structures. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, page 1144, 2013.
 - [89] Faegheh Hasibi and Svein Erik Bratsberg. Non-hierarchical structures: How to model and index overlaps? *CoRR*, abs/1408.1, 2014.
 - [90] Faegheh Hasibi, Darío Garigliotti, Shuo Zhang, and Krisztian Balog. Supervised ranking of triples for type-like relations (The Cress Triple Scorer at WSDM Cup 2017). In *WSDM Cup 2017*.
 - [91] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. A greedy algorithm for finding sets of entity linking interpretations in queries. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, ERD '14, pages 75–78, 2014.
 - [92] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Entity linking in queries: Tasks and evaluation. In *Proceedings of the 2015 ACM International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 171–180, 2015.
 - [93] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Exploiting entity linking in queries for entity retrieval. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 209–218, 2016.
 - [94] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. On the reproducibility of the TAGME entity linking system. In *Proceedings of the 38th European conference on Advances in Information Retrieval*, ECIR '16, pages 436–449, 2016.
 - [95] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Dynamic factual summaries for entity cards. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 773–782, 2017.
 - [96] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Entity linking in queries: Efficiency vs. effectiveness. In *Proceedings of the 39th European conference on Advances in Information Retrieval*, ECIR '17, pages 40–53, 2017.

-
- [97] Faegheh Hasibi, Krisztian Balog, Darío Garigliotti, and Shuo Zhang. Nordlys: A toolkit for entity-oriented and semantic search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1289–1292, 2017.
 - [98] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. Dbpedia-entity v2: A test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1265–1268, 2017.
 - [99] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenauf, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 782–792, 2011.
 - [100] Johannes Hoffart, Dragan Milchevski, and Gerhard Weikum. STICS: searching with strings, things, and cats. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 1247–1248, 2014.
 - [101] Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. Understanding user's query intent with Wikipedia. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 471–480, 2009.
 - [102] Jian Huang, Jianfeng Gao, Jiangbo Miao, Xiaolong Li, Kuansan Wang, Fritz Behr, and C Lee Giles. Exploring Web scale language models for search query processing. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 451–460, 2010.
 - [103] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.
 - [104] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, 2002.
 - [105] Hideo Joho, Lawrence Cavedon, Jaime Arguello, Milad Shokouhi, and Filip Radlinski. First international workshop on conversational approaches

- to information retrieval (CAIR'17). In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1423–1424, 2017.
- [106] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 387–396, 2006.
- [107] Rianne Kaptein and Jaap Kamps. Exploiting the category structure of wikipedia for entity ranking. *Artificial Intelligence*, 194:111–129, 2013.
- [108] Rianne Kaptein, Pavel Serdyukov, Arjen De Vries, and Jaap Kamps. Entity ranking using Wikipedia as a pivot. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 69–78, 2010.
- [109] Lyndon Kennedy, Roelof van Zwol, Nicolas Torzec, and Belle Tseng. Learning crop regions for content-aware generation of thumbnail images. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 30:1–30:8, 2011.
- [110] Jin Young Kim and W. Bruce Croft. A field relevance model for structured document retrieval. In *Proceedings of the 34th European conference on Advances in Information Retrieval*, ECIR '12, pages 97–108, 2012.
- [111] Jinyoung Kim, Xiaobing Xue, and W. Bruce Croft. A probabilistic retrieval model for semistructured data. In *Proceedings of the 31th European conference on Advances in Information Retrieval*, ECIR '09, pages 228–239, 2009.
- [112] Jinyoung Kim, Xiaobing Xue, and W. Bruce Croft. A probabilistic retrieval model for semistructured data. In *Proceedings of the 31th European conference on Advances in Information Retrieval*, ECIR '09, pages 228–239, 2009.
- [113] Wessel Kraaij and Martijn Spitters. Language models for topic tracking. In *Language Modeling for Information Retrieval*, pages 95–123, 2003.
- [114] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of Wikipedia entities in Web text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 457–466, 2009.

-
- [115] John Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 111–119, 2001.
 - [116] Dmitry Lagun, Chih-Hung Hsieh, Dale Webster, and Vidhya Navalpakkam. Towards better measurement of attention and satisfaction in mobile search. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 113–122, 2014.
 - [117] Victor Lavrenko. *A Generative Theory of Relevance*, volume 26. Springer Science & Business Media, 2008.
 - [118] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 120–127, 2001.
 - [119] Taesung Lee, Zhongyuan Wang, Haixun Wang, and Seung-won Hwang. Attribute extraction and scoring: A probabilistic approach. In *Proceedings of 29th International Conference on Data Engineering*, ICDE '13, pages 194–205, 2013.
 - [120] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.
 - [121] Rui Li, Linxue Hao, Xiaozhao Zhao, Peng Zhang, Dawei Song, and Yuexian Hou. A query expansion approach using entity distribution based on Markov Random Fields. In *Proceedings of 11th Asia Information Retrieval Societies Conference*, AIRS '15, pages 387–393, 2015.
 - [122] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 339–346, 2008.
 - [123] Yang Li, Chi Wang, Fangqiu Han, Jiawei Han, Dan Roth, and Xifeng Yan. Mining evidences for named entity disambiguation. In *Proceedings of the*

- 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1070–1078, 2013.
- [124] Jimmy J. Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chatopadhyaya, John Foley, Grant Ingersoll, Craig MacDonald, and Sebastian Vigna. Toward reproducible baselines: The open-source IR reproducibility challenge. In *Proceedings of the 38th European conference on Advances in Information Retrieval*, ECIR '16, pages 408–420, 2016.
- [125] Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Springer Science & Business Media, 2011.
- [126] Xitong Liu and Hui Fang. Latent entity space: A novel retrieval approach for entity-bearing queries. *Inf. Retr.*, 18(6):473–503, 2015.
- [127] Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. Evaluating question answering over linked data. *Web Semantics*, 21:3–13, 2013.
- [128] Chunliang Lu, Wai Lam, and Yi Liao. Entity retrieval via entity factoid hierarchy. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, ACL '15, pages 514–523, 2015.
- [129] Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.*, 1(4):309–317, 1957.
- [130] Yuanhua Lv and ChengXiang Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1895–1898, 2009.
- [131] Craig Macdonald, Rodrygo L Santos, and Iadh Ounis. The whens and hows of learning to rank for Web search. *Inf. Retr.*, 16(5):584–628, 2013.
- [132] Christos Makris, Yannis Plegas, and Evangelos Theodoridis. Improved text annotation with Wikipedia entities. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 288–295, 2013.
- [133] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press Cambridge, 2008.

-
- [134] Olena Medelyan, Ian H Witten, and David Milne. Topic indexing with Wikipedia. In *Proceedings of the AAAI WikiAI workshop*, pages 19–24, 2008.
 - [135] 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 Semant.*, 9(4):418–433, 2011.
 - [136] Edgar Meij, Wouter Weerkamp, and Maarten De Rijke. Adding semantics to microblog posts. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM '12*, page 563, 2012.
 - [137] Edgar Meij, Krisztian Balog, and Daan Odijk. Entity linking and retrieval for semantic search. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM '14*, pages 683–684, 2014.
 - [138] 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, I-Semantics '11*, pages 1–8, 2011.
 - [139] Donald Metzler and W. Bruce Croft. A Markov Random Field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '05*, pages 472–479, 2005.
 - [140] Donald Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Inf. Retr.*, pages 257–274, 2007.
 - [141] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the 16th ACM International Conference on Information and Knowledge Management, CIKM '07*, pages 233–242, 2007.
 - [142] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13*, pages 3111–3119, 2013.
 - [143] David Milne and Ian H Witten. Learning to link with Wikipedia. In *Proceedings of the 17th ACM International Conference on Information and Knowledge Management, CIKM '08*, pages 509–518, 2008.

-
- [144] Calvin N Mooers. The theory of digital handling of non-numerical information and its implications to machine economics. In *Association for Computing Machinery Conference*. University of Minnesota, Minneapolis, 1950.
- [145] Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. DBpedia SPARQL benchmark – performance assessment with real queries on real data. In *The Semantic Web – ISWC 2011: 10th International Semantic Web Conference*, pages 454–469, 2011.
- [146] Vidhya Navalpakkam, LaDawn Jentzsch, Rory Sayres, Sujith Ravi, Amr Ahmed, and Alex Smola. Measurement and modeling of eye-mouse behavior in the presence of nonlinear page layouts. In *Proceedings of the 22Nd International Conference on World Wide Web, WWW '13*, pages 953–964, 2013.
- [147] Robert Neumayer, Krisztian Balog, and Kjetil Nørkvåg. On the modeling of entities for ad-hoc entity search in the Web of Data. In *Proceedings of the 34th European conference on Advances in Information Retrieval, ECIR '12*, pages 133–145, 2012.
- [148] Robert Neumayer, Krisztian Balog, and Kjetil Nørkvåg. When simple is (more than) good enough: Effective semantic search with (almost) no semantics. In *Proceedings of the 34th European conference on Advances in Information Retrieval, ECIR '12*, pages 540–543, 2012.
- [149] Fedor Nikolaev, Alexander Kotov, and Nikita Zhiltsov. Parameterized fielded term dependence models for ad-hoc entity retrieval from knowledge graph. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16*, pages 435–444, 2016.
- [150] Thanapon Noraset, Chandra Sekhar Bhagavatula, and Doug Downey. WebSAIL Wikifier at ERD 2014. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation, ERD '14*, pages 119–124, 2014.
- [151] Daan Odijk, Edgar Meij, and Maarten de Rijke. Feeding the second screen: Semantic linking based on subtitles. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval, OAIR '13*, pages 9–16, 2013.

-
- [152] Paul Ogilvie and Jamie Callan. Combining document representations for known-item search. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '03, pages 143–150, 2003.
- [153] Aasish Pappu, Roi Blanco, Yashar Mehdad, Amanda Stent, and Kapil Thadani. Lightweight multilingual entity extraction and linking. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 365–374, 2017.
- [154] Jovan Pehcevski, James A Thom, Anne-Marie Vercoustre, and Vladimir Naumovski. Entity ranking in Wikipedia: Utilising categories, links and topic difficulty prediction. *Information Retrieval*, 13(5):568–600, 2010.
- [155] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From Freebase to Wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 1419–1428, 2016.
- [156] Jay M Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281, 1998.
- [157] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. Ad-hoc object retrieval in the Web of Data. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 771–780, 2010.
- [158] Roman Prokofyev, Gianluca Demartini, and Philippe Cudré-Mauroux. Effective named entity recognition for idiosyncratic Web collections. In *Proceedings of the 23rd International World Wide Web Conference*, WWW '14, pages 397–408, 2014.
- [159] Silvia Quarteroni, Marco Brambilla, and Stefano Ceri. A bottom-up, knowledge-aware approach to integrating and querying Web data services. *ACM Trans. Web*, 7(4):19:1–19:33, 2013.
- [160] Filip Radlinski and Nick Craswell. A theoretical framework for conversational search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, CHIIR '17, pages 117–126, 2017.

-
- [161] Filip Radlinski and Nick et al. Craswell. Search result driven query intent identification, 2011. US Patent App. 12/813,376.
 - [162] Hadas Raviv, David Carmel, and Oren Kurland. A ranking framework for entity oriented search using Markov Random Fields. In *Proceedings of the 1st Joint International Workshop on Entity-Oriented and Semantic Search, JIWES '12*, 2012.
 - [163] Hadas Raviv, Oren Kurland, and David Carmel. Document retrieval using entity-based language models. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16*, pages 65–74, 2016.
 - [164] Ridho Reinanda, Edgar Meij, and Maarten de Rijke. Mining, ranking and recommending entity aspects. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15*, pages 263–272, 2015.
 - [165] Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. Query segmentation for Web search. In *Proceedings of 12th International World Wide Web Conference, WWW '2003*, 2003.
 - [166] Giuseppe Rizzo, Marieke van Erp, and Raphaël Troncy. Benchmarking the extraction and disambiguation of named entities on the semantic Web. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC*, pages 4593–4600, 2014.
 - [167] Stephen E. Robertson. The probability ranking principle in IR. *Journal of documentation*, 33(4):294–304, 1977.
 - [168] Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.
 - [169] Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.
 - [170] Stephen E. Robertson, Hugo Zaragoza, and Michael Taylor. Simple BM25 extension to multiple weighted fields. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management, CIKM '04*, pages 42–49, 2004.

-
- [171] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System—Experiments in Automatic Document Processing*, pages 313–323, 1971.
 - [172] Daniel E Rose and Danny Levinson. Understanding user goals in Web search. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 13–19, 2004.
 - [173] Gerard. Salton. *Automatic Information Organization and Retrieval*. McGraw Hill Text, 1968.
 - [174] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.
 - [175] Michael Schuhmacher, Laura Dietz, and Simone Paolo Ponzetto. Ranking entities for Web queries through text and knowledge. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1461–1470, 2015.
 - [176] Prithviraj Sen. Collective context-aware topic models for entity disambiguation. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 729–738, 2012.
 - [177] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Building bridges for Web query classification. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 131–138, 2006.
 - [178] Milad Shokouhi and Qi Guo. From queries to cards: Re-ranking proactive card recommendations based on reactive search history. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 695–704, 2015.
 - [179] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, 2007.
 - [180] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: a large ontology from Wikipedia and WordNet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217, 2008.

-
- [181] Bin Tan and Fuchun Peng. Unsupervised query segmentation using generative language models and Wikipedia. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 347–356, 2008.
- [182] Andreas Thalhammer and Achim Rettinger. Browsing DBpedia entities with summaries. In *Proceedings of the 11th International Conference on The Semantic Web*, ESWC '14, pages 511–515. 2014.
- [183] Andreas Thalhammer, Nelia Lasierra, and Achim Rettinger. LinkSUM: Using link analysis to summarize entity data. In *Proceedings of 16th International Conference on Web Engineering*, ICWE '16, pages 244–261.
- [184] Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, Paolo Ferragina, Christiane Lemke, Andrea Moro, Roberto Navigli, Francesco Piccinno, Giuseppe Rizzo, Harald Sack, René Speck, Raphaël Troncy, Jörg Waitelonis, and Lars Wesemann. GERBIL: General entity annotator benchmarking framework. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1133–1143, 2015.
- [185] Srinivas Vadrevu, Ying Tu, and Franco Salvetti. Ranking relevant attributes of entity in structured knowledge base, 2016. US Patent 9,229,988.
- [186] David Vallet and Hugo Zaragoza. Inferring the most important types of a query: A semantic approach. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 857–858, 2008.
- [187] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 165–174, 2016.
- [188] Vasudeva Varma, Vijay Bharath Reddy, Sudheer Kovelamudi, Praveen Bysani, Santhosh Gsk, N Kiran Kumar, Kranthi Reddy B, Karuna Kumar, and Nitin Maganti. IIIT hyderabad at TAC 2009. In *TAC*, 2009.
- [189] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on*

- Research and Development in Information Retrieval*, SIGIR '94, pages 61–69, 1994.
- [190] Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. Learning to explain entity relationships in knowledge graphs. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, ACL '15, pages 564–574.
- [191] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.
- [192] Jörg Waitelonis and Harald Sack. Towards exploratory video search using linked data. *Multimedia Tools Appl.*, 59(2):645–672, 2012.
- [193] Qiuyue Wang, Jaap Kamps, Georgina Ramírez Camps, Maarten Marx, Anne Schuth, Martin Theobald, Sairam Gurajada, and Arunav Mishra. Overview of the INEX 2012 linked data track. In *CLEF Online Working Notes*, 2012.
- [194] Yue Wang, Dawei Yin, Luo Jie, Pengyuan Wang, Makoto Yamada, Yi Chang, and Qiaozhu Mei. Beyond ranking: Optimizing whole-page presentation. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 103–112, 2016.
- [195] Ian H Witten and David Milne. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy*, pages 25–30, 2008.
- [196] Qiang Wu, Christopher J Burges, Krysta M Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3): 254–270, 2010.
- [197] Chenyan Xiong and Jamie Callan. EsdRank: Connecting query and documents through external semi-structured data. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, CIKM '15, pages 951–960, 2015.
- [198] Chenyan Xiong and Jamie Callan. Query expansion with Freebase. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 111–120, 2015.

-
- [199] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Bag-of-entities representation for ranking. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 181–184, 2016.
 - [200] Chenyan Xiong, Russell Power, and Jamie Callan. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1271–1279, 2017.
 - [201] Danyun Xu, Gong Cheng, and Yuzhong Qu. Facilitating human intervention in coreference resolution with comparative entity summaries. In *The Semantic Web: Trends and Challenges*, pages 535–549. 2014.
 - [202] Jun Xu and Hang Li. Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 391–398, 2007.
 - [203] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. Natural language questions for the Web of Data. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 379–390, 2012.
 - [204] Ikuya Yamada, Tomotaka Ito, Shinnosuke Usami, Shinsuke Takagi, Hideaki Takeda, and Yoshiyasu Takefuji. Evaluating the helpfulness of linked entities to readers. In *Proceedings of the 25th ACM Conference on Hypertext and Social Media*, HT '14, pages 169–178, 2014.
 - [205] Emine Yilmaz, Manisha Verma, Rishabh Mehrotra, Evangelos Kanoulas, Ben Carterette, and Nick Craswell. Overview of the TREC 2015 tasks track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, (TREC 2015)*, 2015.
 - [206] ChengXiang Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2:137–213, 2008.
 - [207] Chengxiang Zhai and John Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 403–410, 2001.

-
- [208] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.
 - [209] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. A general boosting method and its application to learning ranking functions for Web search. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS’07, pages 1697–1704, 2007.
 - [210] Nikita Zhiltsov, Alexander Kotov, and Fedor Nikolaev. Fielded sequential dependence model for ad-hoc entity retrieval in the Web of Data. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’15, pages 253–262, 2015.

