Robert Neumayer

Semantic and Distributed Entity Search in the Web of Data

Thesis for the degree of Philosophiae Doctor

Trondheim, February 2013

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

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Computer and Information Science

© Robert Neumayer

ISBN 978-82-471-4208-0 (printed ver.)
ISBN 978-82-471-4209-7 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2013:56

Printed by NTNU-trykk

Abstract

Both the growth and ubiquitious character of the Internet have had a profound effect on how we access and consume ata and information. More recently, the Semantic Web, an extension of the current Web has come increasingly relevant due to its widespread adoption.

The Web of Data (WoD) is an extension of the current web, where not only documents are interlinked by means of hyperlinks but also data in terms of predicates. Specifically, it describes objects, entities or "things" in terms of their attributes and their relationships, using RDF data (and often is used equivalently to Linked Data). Given its growth, there is a strong need for making this wealth of knowledge accessible by keyword search (the de-facto standard paradigm for accessing information online).

The overall goal of this thesis is to provide new techniques for accessing this data, i.e., to leverage its full potential to end users. We therefore address the following four main issues: a) how can the Web of Data be searched by means of keyword search?, b) what sets apart search in the WoD from traditional web search?, c) how can these elements be used in a theoretically sound and effective way?, and d) How can the techniques be adapted to a distributed environment?

To this end, we develop techniques for effectively searching WoD sources. We build upon and formalise existing entity modelling approaches within a generative language modelling framework, and compare them experimentally using standard test collections. We show that these models outperform the current state-of-the-art in terms of retrieval effectiveness, however, this is done at the cost of abandoning a large part of the semantics behind the data. We propose a novel entity model capable of preserving the semantics associated with entities, without sacrificing retrieval effectiveness. We further show how these approaches can be applied in the distributed context, both with low (federated search) and high numbers (Peer-to-peer or P2P) of independent repositories, collections, or nodes.

The main contributions are as follows:

- We develop a hybrid approach to search in the Web of Data, using elements from traditional information retrieval and structured retrieval alike.
- We formalise our approaches in a language model setting.
- Our extensions are successfully evaluated with respect to their applicability in different distributed environments such as federated search and P2P.
- We discuss and analyse based on our empirical evaluation and provide insights into the entity search problem.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for partial fulfilment of the requirements for the degree of philosophiae doctor.

The PhD work summarised in this thesis has been carried out at the Department of Computer and Information Science, NTNU, Trondheim. The main supervisor was Kjetil Nørvåg. Both Jon Atle Gulla and Trond Aalberg were assigned co-supervisors. Krisztian Balog joined the team of supervisors later on.

Funding was provided by the Norwegian Research Council. The research presented in this thesis is financed by grant #183337/S10 (COMIDOR, Cooperative Mining of Independent Document Repositories).

Acknowledgements

First, I would like to acknowledge Kjetil's contribution. He always found time to discuss ideas and to comment on my writings and doings—both in a constructive and honest way. Thanks a lot for the vegan chocolate late at night, especially when I have not had anything else to eat before that, it always made my day.

Special thanks go to my co-supervisor Krisztian who was extremely helpful and contributed a lot not only in discussions and developing ideas but also in terms of implementation and writing, gg. Thanks also to all of my other co-authors and to my previous supervisor to get me involved in information retrieval in the first place.

Without exceptions, everyone at the department has been extremely helpful and always friendly. Thanks for supporting my every struggle—be it with forms I could not handle or with the copying machine; all of you provided a great working environment. I always enjoyed common coffee and lunch breaks, whenever I could be there.

I am also thankful to João—for providing the most comfortable office environment and for putting up with listening to my song of the day and lots of other music he never was interested in listening to, insights into various topics, and for the soy icecream which turned out not being good enough for himself. I really appreciate all these things.

Thanks to special people I encountered around here and friends and family (yes, this includes parents) for supporting and encouraging me from afar—and for not forgetting about me. Thanks a million to everyone who tried to be here with me in Norway and their struggle with both weather and culture. Thanks to Ifat for trying so long; it meant everything to me.

Contents

Part I

Introduction and Background

In the beginning of this thesis, we outline the motivation and introduce the main scenario for our work in Chapter 1. Here we also state our research questions and provide an overview of the remainder of the thesis. Chapter 2 summarises the most relevant related work and provides the foundation for the remaining parts.

Chapter 1

Introduction

This chapter gives a motivation and introduction to the main topic of the thesis. We give introductory examples pointing out some important limitations of the current state of the art. Further, we outline the relevant research areas and present current challenges therein. We continue with formulating research questions and put the main findings of the thesis in context. Finally we list the contributions in terms of publications and give an overview of the organisation of the rest of the thesis.

1.1 Motivation

Web search has become one of the main methods people use to find information. In short, web search describes the search process of a user issuing a query to satisfy her or his information need. Keyword queries transmitted via a single search box have become the de-facto standard in this context, simply because Web users have become used to this kind of query and are not familiar with more complex, structured query languages. The average length of web queries has shown to be between two and three words [108].

The multitude of possible information needs range from finding general facts about a certain topic to requesting very specific pieces of information. Information needs and their satisfaction strongly depend on the user, i.e., what one user finds relevant for a given query might differ from another one, based on their actual information need even though they might issue identical queries.

In general, web queries can be classified into the following three categories [63]:

- *Informational queries* describe broad topics for which there are many pages. Usually there is no single web page which provides all the information the user is looking for. The user expects a list of several relevant pages.

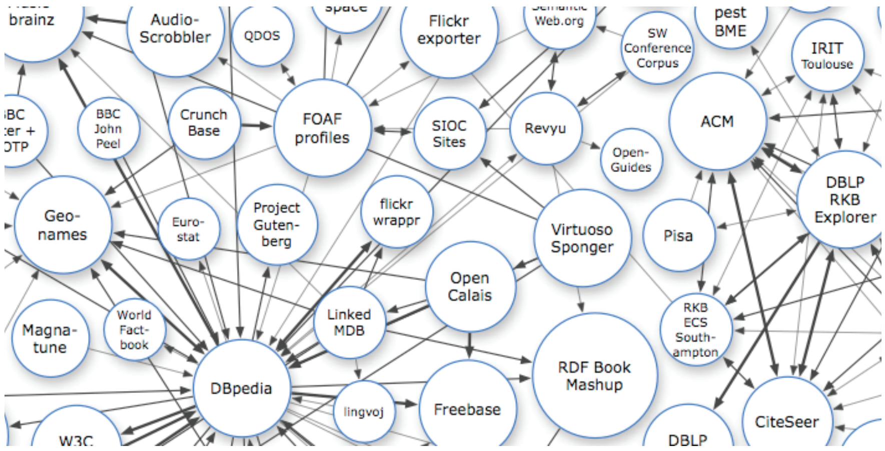


Figure 1.1: Overview of the data sets included in the Linked Open Data project (`http://linkeddata.org/`).

- *Navigational queries* denote searches for specific web pages or objects. The user expects the web page of the entity of interest to be the first result he is presented with (in fact, this is the only result she or he is interested in).
- *Transactional queries* are targeted at specific actions like purchasing a product or downloading a file. The search engine should list services providing interfaces for this action.

In this thesis our main focus—according to these definitions—lies on a subset of navigational and informational queries. More specifically we focus on the search for specific things or objects of interest users might search for.

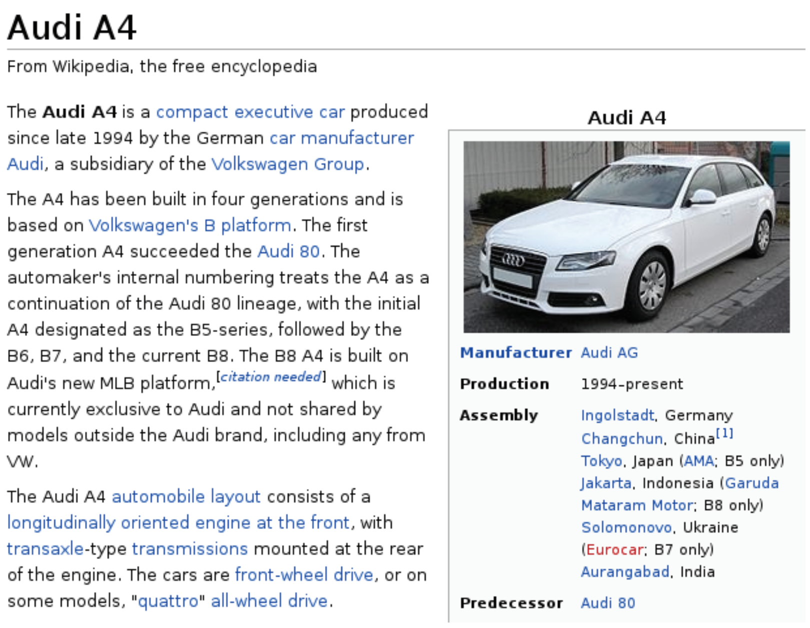
In our context, such objects or things are called *entities*. An entity can, amongst others, be a person, a location, a service, or a product as opposed to other sources of information such as newspaper articles or other documents covering multiple topics or describing multiple objects. According to recent research, a considerable amount of all web search queries target such entities [92].

1.1.1 Entity Search in the Web of Data

The increasing interest in entity-related search coincides with the fact that there is an increasing amount of information published as Linked Data, the foundation of the Web of Data.

The past three years in particular have seen a significant increase in the number of knowledge bases published as Linked Data (such as DBpedia, Freebase,[1] and others). At the same time, we observe an increase and in the availability of metadata

[1]`http://www.freebase.com`



Audi A4

From Wikipedia, the free encyclopedia

The **Audi A4** is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the Volkswagen Group.

The A4 has been built in four generations and is based on Volkswagen's B platform. The first generation A4 succeeded the Audi 80. The automaker's internal numbering treats the A4 as a continuation of the Audi 80 lineage, with the initial A4 designated as the B5-series, followed by the B6, B7, and the current B8. The B8 A4 is built on Audi's new MLB platform,[*citation needed*] which is currently exclusive to Audi and not shared by models outside the Audi brand, including any from VW.

The Audi A4 automobile layout consists of a longitudinally oriented engine at the front, with transaxle-type transmissions mounted at the rear of the engine. The cars are front-wheel drive, or on some models, "quattro" all-wheel drive.

Audi A4

Manufacturer	Audi AG
Production	1994–present
Assembly	Ingolstadt, Germany Changchun, China[1] Tokyo, Japan (AMA; B5 only) Jakarta, Indonesia (Garuda Mataram Motor; B8 only) Solomonovo, Ukraine (Eurocar; B7 only) Aurangabad, India
Predecessor	Audi 80

Figure 1.2: Example of a Wikipedia of an entity.

embedded inside web pages (RDF, RDFa, Microformats, and others) [92]. The combination of these two facts further drives research interest in the direction of entity search.

The Linking Open Data (LOD) initiative[2] provides a platform for publishing and linking open data sets. Currently their registry contains 326 open data sets.[3] This number went up from 12 data sets in 2007,[4] showing a strong increase in availability and popularity of Linked Data. A part of the LOD diagram is shown in Figure 1.1, including prominent WoD data sets such as *DBpedia*, *Geonames*, *DBLP*, or *Freebase*. From the figure it also becomes apparent that some data sets are more prominent than others, e.g., *DBpedia*, as it has a very high number of both incoming and outgoing links as shown by the strength of the arrows connecting it to others.

The growing amount of embedded semantic data such as RDFa and microdata is documented in the Web Data Commons project. Such embedded data is all the more important since it provides a good opportunity for user generated content to be a part of the WoD. The goal of the project is to provide aggregated data based on microdata embedded in a common web crawl.[5] Amongst others, the authors

[2]`http://linkeddata.org/`
[3]`http://thedatahub.org/group/lodcloud`
[4]`http://richard.cyganiak.de/2007/10/lod`
[5]`http://webdatacommons.org/`

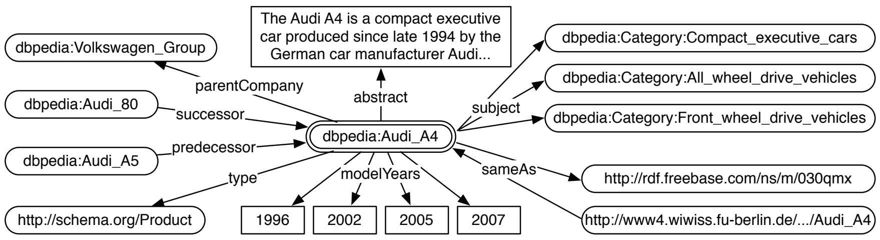

The Audi A4 is a compact executive car produced since late 1994 by the German car manufacturer Audi...
dbpedia:Volkswagen_Group
parentCompany
dbpedia:Audi_80
successor
abstract
dbpedia:Audi_A4
predecessor
dbpedia:Audi_A5
type
modelYears
http://schema.org/Product
1996
2002
2005
2007
subject
dbpedia:Category:Compact_executive_cars
dbpedia:Category:All_wheel_drive_vehicles
dbpedia:Category:Front_wheel_drive_vehicles
sameAs
http://rdf.freebase.com/ns/m/030qmx
http://www4.wiwiss.fu-berlin.de/.../Audi_A4

Figure 1.3: Linked Data representation of an entity.

documented a 23 percent rise in the amount of embedded RDFa data from 2010 to 2012.

In this thesis we assume an entity to be an element of the Web of Data (WoD). [6] The WoD is inherently organised around entities; each entity is identified by a unique URI and is described using a set of subject-predicate-object RDF triples (from which an adequate entity representation has to be created).

A prominent source of partly structured data is Wikipedia[7] or its machine-readable version DBpedia.[8] Figure 1.2 shows the wikipedia page of one entity, the Audi A4 car. Wikipedia offers structured information to some extent, as can be seen in the fact box on the right hand side of the figure. However, the most part of the information provided is free text and the relative amount of free text and structured info varies greatly (i.e., some pages contain lots of free text with little fact box info or the other way around). Machine-readability is one of the main features of semantic technologies, strongly contributing to its interestingness as a research topic: the automatic analysis of Linked Data is highly desirable due to its size. All data is organised by means of their URI and links in between entities. DBpedia offers all its data in the machine-readable RDF (resource description framework) format.[9] This format is a simple way of making information about resources available online. The basic idea is that every fact can be expressed as a triple:

```
(subject, predicate, object)
```

and all three elements are URIs or literals. For example, the triple:[10]

```
(dbpedia.org/Audi_A4, dbpedia.org/ModelYears, ‘‘2006’’)
```

denotes that the Audi A4 was manufactured in the year 2006. Figure 1.3 shows more elements of the DBpedia representation of the Wikipedia page shown be-

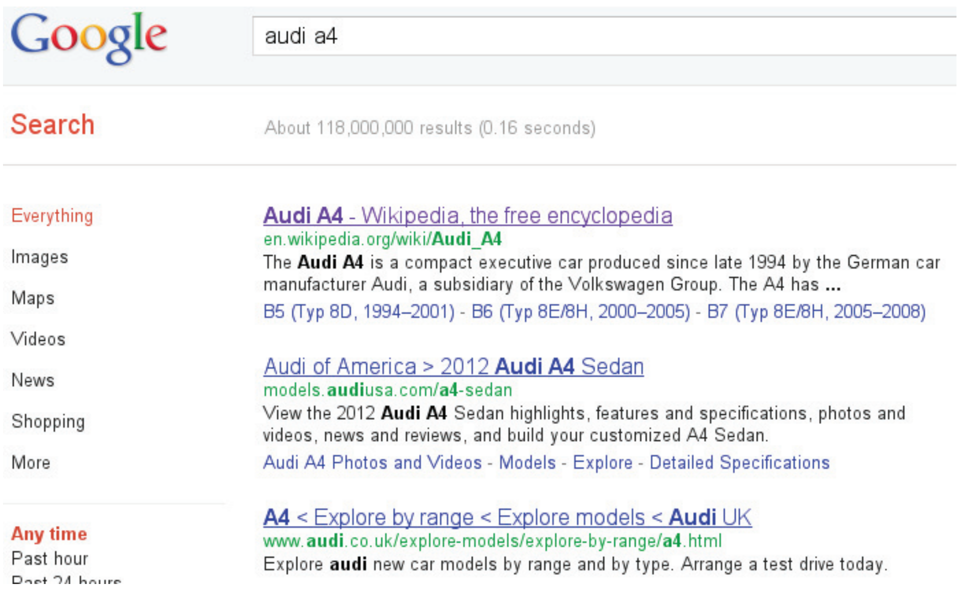
[6]This type of data is also referred to as Linked Data. In fact Linked Data is a publishing mechanism and one component of realising the Web of Data. In this thesis, even though we are aware of the slight semantic differences, we primarily use the term WoD, but in this thesis both terms can be treated as synonyms.

[7]`http://www.wikipedia.org`

[8]`http://dbpedia.org`

[9]`http://www.w3.org/RDF`

[10]For reasons of easier readability we omit the “http” prefixes.


audi a4

Search

About 118,000,000 results (0.16 seconds)

Everything
Images
Maps
Videos
News
Shopping
More

Any time
Past hour

Audi A4 - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Audi_A4
The **Audi A4** is a compact executive car produced since late 1994 by the German car manufacturer Audi, a subsidiary of the Volkswagen Group. The A4 has ...
B5 (Typ 8D, 1994–2001) - B6 (Typ 8E/8H, 2000–2005) - B7 (Typ 8E/8H, 2005–2008)

Audi of America > 2012 Audi A4 Sedan
models.audiusa.com/a4-sedan
View the 2012 **Audi A4** Sedan highlights, features and specifications, photos and videos, news and reviews, and build your customized A4 Sedan.
Audi A4 Photos and Videos - Models - Explore - Detailed Specifications

A4 < Explore by range < Explore models < Audi UK
www.audi.co.uk/explore-models/explore-by-range/a4.html
Explore **audi** new car models by range and by type. Arrange a test drive today.

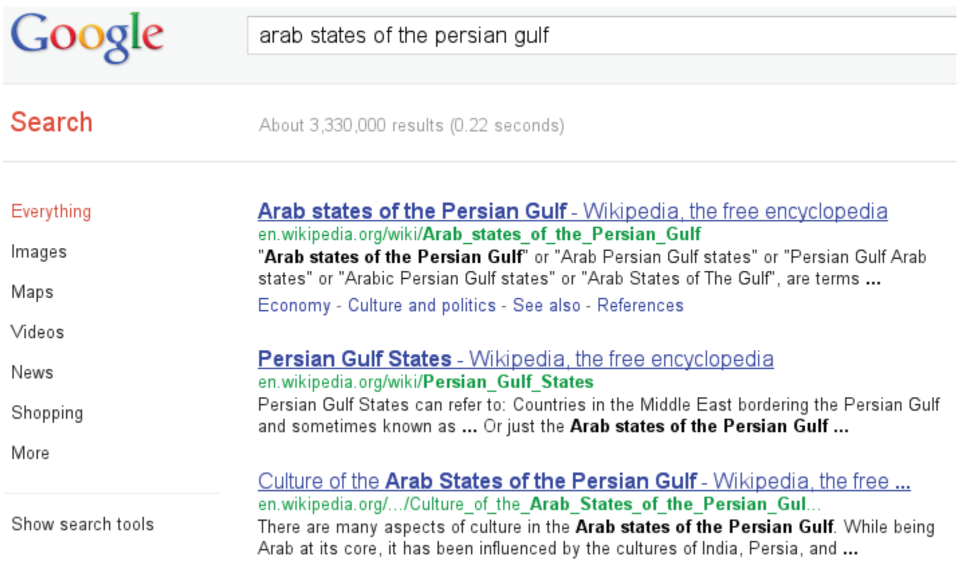
Figure 1.4: Google search for the "Audi A4" entity.

fore. The aforementioned "modelYears" relation is a literal predicate since it assigns string values to the entity. The "sameAs" predicates denote identity relations between different entities, each identified by an URI. Note that relation type predicates can be both in- and outgoing.

This figure shows in particular the importance of individual predicates. The "modelYears" predicate, for example, will be a good predicate to search in when looking for dates. However, ad-hoc search on the model year will not yield any results because it only contains dates. For this reason we stress the need for analysing which query terms are best matched against which predicates. Another benefit from following such a paradigm is that evolving data can be handled easier (e.g., when predicate names change or new predicates are added to an existing entity). As such, it is desirable to model individual predicates, i.e., keep the structure of the entity rather than only searching in its textual predicates.

1.1.2 Limitations of Current Web Search Engines

We show a screenshot from a traditional search engine for the "Audi A4" query in Figure 1.4. The search engine shows the correct result with respect to Wikipedia at rank one (the first result returned) and it shows a summary of the different models under the representative snippets, but fails to cover the structure of the Web of Data and does not highlight the most important predicates. Users might be most interested in the price of the car, or a particular model might be more popular


arab states of the persian gulf

Search

About 3,330,000 results (0.22 seconds)

Everything
Images
Maps
Videos
News
Shopping
More

Show search tools

Arab states of the Persian Gulf - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Arab_states_of_the_Persian_Gulf**
"**Arab states of the Persian Gulf**" or "Arab Persian Gulf states" or "Persian Gulf Arab states" or "Arabic Persian Gulf states" or "Arab States of The Gulf", are terms **...**
Economy - Culture and politics - See also - References

Persian Gulf States - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Persian_Gulf_States**
Persian Gulf States can refer to: Countries in the Middle East bordering the Persian Gulf and sometimes known as **...** Or just the **Arab states of the Persian Gulf ...**

Culture of the **Arab States of the Persian Gulf** - Wikipedia, the free **...**
en.wikipedia.org/.../Culture_of_the_**Arab_States_of_the_Persian_Gul**...
There are many aspects of culture in the **Arab states of the Persian Gulf**. While being Arab at its core, it has been influenced by the cultures of India, Persia, and **...**

Figure 1.5: Google search for the "Arab states of the Persian gulf" list search query.

than others—this type of information could be covered by exploiting the entity's predicate structure.

Whereas this is most likely not a real problem in terms of finding what a user is looking for (in this case maybe prices or reviews of the Audi A4 since its entity type is "car"), it clearly shows the limitations in terms of relations between entities which are neither displayed nor exploited. Furthermore, the "Audi A4" targets one specific entity. However, more complex queries make the limitations of web search engines even more evident as shown in the following.

List search denotes the task of searching for a list of entities specified by a query. One example is the query "Arab states of the Persian gulf". The correct result would encompass all of the six Arab states located at the Persian gulf (that is, Saudi Arabia, United Arab Emirates, Qatar, Kuwait, Bahrain and Oman) and their respective URIs. However, as shown in Figure 1.5, web search engines fail completely at this task. The result comprises good hits for pages containing the correct answers, but fails to list the individual states.

These shortcomings in combination with the increasing availability of knowledge bases has made search in the Web of Data an active research area, especially propelled by recent benchmarking initiatives, such as the introduction of the Semantic Search evaluation series [20, 46], or related INEX tasks [32, 50]. Benchmarking initiatives have further advanced research in this direction, by providing a common evaluation platform to empirically assess methods and algorithms devised for the task that has been termed *ad-hoc entity retrieval*: "answering arbitrary information needs related to particular aspects of objects [entities], expressed in unconstrained natural language and resolved using a collection of structured data" [92]. Since the WoD is inherently organised around objects or entities, having entities as the unit of retrieval follows naturally.

1.2 Scope of the Thesis

In the following we will point out some of the problems occurring in entity search, particularly focusing on the differences to traditional web search. We show some of the main shortcomings of web search with respect to linked data. The Web of Data itself has unique properties, by which it can be distinguished from traditional search in the Web of Documents:

- Entities are described in a semi-structured way instead of the dominant full text representation of the Web (with the exception of basic semi-structured information in terms of HTML).
- Relations between entities have types as opposed to the limitation to hyperlinks in the case of the WWW (i.e., the fact that only one type of link exists).
- The Web of Data consists of multiple collections (or sources) such as DBpedia or geonames.[11] Similarly the Web itself consists of many domains, the degree of linkage in the Web of Data, however, makes it particularly interesting to combine its sources.

These properties also constitute what we mean by "semantic" in the course of this thesis, namely considering the meaningful structures that exist within the data. These structures are the main difference to standard web search or the standard bag-of-words approach of IR and we will try to exploit their availability as good as we can. Whilst traditional web search engines have limited support for these unique features, the vast potential they hold has yet to be exploited.

The traditional way of searching textual data is to search the whole document disregarding any structural information which might be available. Structured data, in the simple case documents with a *title/content* distinction, or DBpedia entries with a possibly long list of predicates, require more advanced retrieval models. Structured retrieval has been established as a way to search structured or semi-structured text collections, combining retrieval scores over multiple fields (i.e., in the context of e-mails this would mean "subject" or "body" fields). We therefore have a strong focus on structured retrieval models in our research where we mainly are concerned with their applicability to the entity search use case.

The WoD is an extension of the WWW and as such inherently distributed. Linked Data sources lie on different servers, belong to different organisations, and adhere to different schemas. However, current state-of-the-art web search engines crawl the

[11]`http://www.geonames.org`

web and then perform search on a central index.[12] The strengths of the individual data sources or collections, though, lies in their familiarity with their own data and their capability to provide search functionality. In distributed approaches the main assumption is that the owners of collections know best how to search in their own data and how to provide adequate access. Therefore we also look at scenarios where we search several distributed collections and we present this in a separate part of the thesis. Especially with a continuing strong growth in both the size of existing knowledge sources and the addition of new ones, we consider the distributed aspect important and therefore will propose solutions for different scenarios. Further, current web search engines, do not support automated processing of results, i.e., their results are not machine-readable, which is one of the major hopes put in semantic search technology. If these answers could be automatically discovered and put in context to each other, semantic formats such as RDF can facilitate easy distribution and reusability. This point becomes more urgent with the continuing growth of the Web of Data.

To summarise, we will explore different aspects of the problem of entity search. We focus on the areas of advanced retrieval models of entity search and their application in the distributed context. In the following, we will give an overview of research questions we explored.

1.3 Research Questions

After having specified the overall context of our research work, we introduce the main research questions we seek to answer in the course of this thesis:

RQ1: A plethora of techniques have been developed in Information Retrieval over decades. Most of these techniques are tailored to flat text documents. The majority of them do not exploit the unique characteristics of WoD and entities do not have a direct textual representation.

How can traditional ad-hoc document retrieval techniques be applied in the context of the Web of Data?

RQ2: One straightforward approach to making use of structure is to create a representation based on multiple fields. We will provide a thorough investigation of the strengths of structured retrieval and show how it can be applied in the context of the WoD.

How can the structure of entities be exploited for the purpose of ad-hoc retrieval?

RQ3: Many well-performing methods in entity search make use of the fielded structure of the data as indicated by its predicate structure. This predicate structure can be used by assigning different weights to different types of predicates.

[12] On a conceptual level these indices are centralised, in order to deal with computational challenges, search engines make heavy use of parallelisation.

How does field weighting affect search quality?

RQ4: The Web of Data is inherently distributed. As such, distributed search and its mature retrieval models may be a viable option when deciding how to best query its data.

Can existing, standard federated search techniques be applied to entity search in the Web of Data?

RQ5: Federated search techniques are able to exploit local features. Subcollections might be able to provide better rankings (i.e., local rankings) than global models.

Can federated entity search benefit from improved entity modelling?

RQ6: Federated search is usually concerned with a rather low number of disjoint repositories (usually around 100), i.e., more large-scale approaches might be of interest given the current growth of the Web of Data. The switch from a federated search to a P2P scenario brings with it the change of the number of involved nodes from up to ten to up to several thousands. This also poses challenges in terms of scalability and dynamicity.

Is P2P search a viable alternative to broker-based (i.e., federated search) architectures for entity retrieval?

RQ7: P2P networks are highly distributed and dynamic systems. In such a scenario it is particularly difficult to provide global statistics (such as term frequencies) about the involved collections. Hierarchical aggregation of frequencies in P2P networks is a heuristic technique for estimating such statistics and has the advantage of providing accurate values. However, techniques based on gossiping might improve the results even though they provide less accurate numbers.

How can the proposed frequency estimation technique be further improved?

1.4 Contributions and Research Papers

The majority of the content of this thesis has been published in international conferences and journals in the course of the last 4 years. In the following we show a chronological overview and point out the relevance of the published articles, put it in relation to the aforementioned research questions, and specify where in the thesis they will be used.

P1 Robert Neumayer, Christos Doulkeridis, and Kjetil Nørvåg. Aggregation of document frequencies in unstructured P2P networks. In *Proceedings of 10th International Conference on Web Information Systems Engineering (WISE'09)*, pages 29–42, Poznan, Poland, October 5-7 2009. Springer
Relevance to this thesis: The findings of this paper will be used in Chapter 7. It covers an initial analysis of the data sharing problem in peer-to-peer text search (P2P). We answer research questions RQ4 and RQ5.

P2 Robert Neumayer, Christos Doulkeridis, and Kjetil Nørvåg. A hybrid approach for estimating document frequencies in unstructured P2P networks. *Information Systems*, 36(3):579–595, May 2011
Relevance to this thesis: The findings of this paper will be used in Chapter 7. In this paper we presented new ideas for the P2P problem also discussed in P1. Further, we provide an in-depth experimental evaluation. We provide further analysis of RQ6 and answer RQ7.

P3 Krisztian Balog, Marek Ciglan, Robert Neumayer, Wei Wei, and Kjetil Nørvåg. NTNU at SemSearch 2011. In *Proceedings of the 4th International Semantic Search Workshop of the 20th Int. World Wide Web Conference WWW2011)*, pages –, Hyderabad, India, April 19-21 2011
Relevance to this thesis: We use the contents of this paper in Chapter 3. We report the results of our submission to the semantic search challenge and put our approach in context to other submissions. As we show the applicability of IR approaches to entity search, it is an initial exploration of RQ1.

P4 Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. When simple is (more than) good enough: Effective semantic search with (almost) no semantics. In *Proceedings of the 34rd European Conference on Information Retrieval (ECIR'12)*, pages 540–543, Barcelona, Spain, April 1 - 5 2012
Relevance to this thesis: As a part of Chapter 4, we show how a rather straightforward model can outperform all of the state-of-the-art methods on one given benchmark data set. Further, we describe the fielded models which are the foundation of this work. We answer research questions RQ1, RQ2, and RQ3.

P5 Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. On the modeling of entities for ad-hoc entity search in the web of data. In *Proceedings of the 34rd European Conference on Information Retrieval (ECIR'12)*, pages 133–145, Barcelona, Spain, April 1 - 5 2012
Relevance to this thesis: Building on the results of P4, we show how additional structural elements can be taken into account, which is also included in Chapter 4. To this end we propose a novel retrieval model, considering both individual predicates and types of thes predicates. We answer research questions RQ2 and RQ3.

P6 Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. Ranking distributed knowledge repositories. In *Proceedings of the International Conference on Theory and Practice of Digital Libraries Research and Advanced Technology for Digital Libraries (TPDL'12)*, pages 486–491, Paphos, Cyprus, September 23 - 27 2012. Springer
Relevance to this thesis: In Chapter 5, we formalise the task of collection selection in a language modelling framework and propose baselines methods to handle this task. For evaluation, we introduce a test collection building on existing benchmark corpora. RQ4 is investigated and answered in this paper.

P7 Krisztian Balog, Robert Neumayer, and Kjetil Nørvåg. Collection ranking and selection for federated entity search. In *Proceedings of 18th International Symposium of String Processing and Information Retrieval (SPIRE'12)*, Cartagena, Colombia, October 21-25 2012. Springer. Accepted for publication
Relevance to this thesis: Based on the results from P6, we continue by proposing a new method for collection selection in the entity context in Chapter 6. We evaluate our new method by using the benchmark collection introduced in P6. We further investigate RQ5.

1.5 Additional Papers

The following papers were published in the course of this PhD, but are not included in the thesis because they are concerned with areas only marginally connected to its main topics:

P8 Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Data recovery from distributed personal repositories. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries (ECDL'09)*, Corfu, Greece, September 27 - October 2 2009. Springer

P9 Rudolf Mayer and Robert Neumayer. Multi-modal analysis of music: A large-scale evaluation. In *Proceedings of the Workshop on Exploring Musical Information Spaces, (WEMIS'09)*, pages 30–35, Corfu, Greece, October 1-2 2009

P10 Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Interacting with (semi-) automatically extracted context of digital objects. In *Proceedings of the Workshop on Context, Information And Ontologies (CIAO'09)*, pages 1–9, Heraklion, Greece, June 1 2009. ACM

P11 Rudolf Mayer, Robert Neumayer, Doris Baum, and Andreas Rauber. Analytic comparison of Self-organising Maps. In *Proceedings of the 7th International Workshop on Self-Organizing Maps (WSOM'09)*, pages 182–190, St. Augustine, FL, USA, 2009. Springer

P12 Robert Neumayer, Rudolf Mayer, and Kjetil Nørvåg. Combination of feature selection methods for text categorisation. In *Proceedings of the 33rd European Conference on Information Retrieval (ECIR'11)*, pages 763–766, Dublin, Ireland, April 19-21 2011

P13 Robert Neumayer, George Tsatsaronis, and Kjetil Nørvåg. TRUMIT: A tool to support large-scale mining of text association rules. In *Proceedings of the 10th European Conference on Machine Learning and Knowledge Discovery in Databases - European Conference, (ECML PKDD 2011)*, pages 646–649, Athens, Greece, September 5-9 2011. Springer

P14 Robert Neumayer and Kjetil Nørvåg. Evaluation of feature combination approaches for text categorisation. In *Proceedings of the 19th International Symposium on Methodologies for Intelligent Systems (ISMIS'11)*, pages 438 – 448, Warsaw, Poland, June 28-30 2011

P15 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)*, Maui, HI, USA, October 29 - November 2 2012. Poster, accepted for publication

1.6 Thesis Structure

This thesis is organised in four main parts. Part I gives an introduction into the main topics of the thesis and summarises relevant background in these areas. Then, in parts II and III, we describe the main research topics of the thesis. Finally, we draw conclusions and give an overview of future work in part IV. A more detailed outline of the contents is given in the following:

Part I Introduction and Background

Chapter 1 gives both motivation and introduction to the research laid out later in the thesis. We give examples for entity search and describe some of the main problems of the area.

Chapter 2 introduces the technical foundations for the later chapters. We introduce the most relevant techniques and position our work within the area of information retrieval and semantic search.

Part II Entity Search

Chapter 3 summarises our research on entity search. We outline the problem statement and propose solutions for dynamic search for entities in the Web of Data. We address research question RQ1.

Chapter 4 investigates more advanced retrieval models based on structural retrieval. These models are adapted for the entity search case. We also provide experimental evaluation to show the applicability of our models. We answer research questions RQ2 and RQ3.

Part III Distributed Aspects

Chapter 5 builds on the techniques introduced in Part II and suggests modifications to make them applicable to a distributed context. More specifically, we use a federated search architecture to show how entity search can work in the distributed case. This chapter takes a closer look at research question RQ4.

Chapter 6 makes use of the probabilistic federated search framework given in Chapter 5. We introduce extensions and new models for several federated search components. In this chapter, we investigate research questions RQ5 and RQ6.

Chapter 7 further extends to the area of peer-to-peer (P2P) search and shows how to tackle some of the problems introduced by that scenario. We in chapter we answer research question RQ7.

Part IV Discussion and Outlook

Chapter 8 sums up the contributions of the thesis, and gives an overview of possible follow up work. We further give a more general outlook on the field.

Chapter 2

Background

We gave an outline of the scenario of entity search in Chapter 1. In this chapter, we provide an overview of the background in related areas and mention its applicability to the entity search problem. The techniques we will introduce in this chapter are known to be well-working when applied to search in text documents. In the following sections we will make use of these approaches and show how they can be applied in the main parts II and III.

2.1 Overview

First, we introduce the area of information retrieval (IR) since we build on many basic techniques and because we face similar problems. IR research methods are summarised in Section 2.2. There, we also give a short overview of benchmarking initiatives in IR along with its evaluation measures and detail how they can and will be used to evaluate our methods in Section 2.2.2. We then continue to be more specific by introducing the task of "document retrieval" and take a closer look at the most typical units of retrieval, documents, in Section 2.3. In this section we also move on to presenting an overview of probabilistic retrieval models, in particular language models, which we use in many of the publications which are part of this thesis, in Section 2.3.1. In Section 2.3.2, we discuss structured retrieval approaches since these models will be instruments for semantic search in later chapters. We give an overview of distributed aspects within both IR and entity search in Section 2.4.1, as well as peer-to-peer (P2P) search in 2.4.2. Next, we introduce some basics of semantic search in Section 2.5. Finally, in Section 2.6, we then proceed to give an overview of the task of entity search and how it has recently been approached.

2.2 Information Retrieval

Information retrieval is a diverse area of computer science, which has heavily been researched for decades. The most basic description for IR is simply that it deals with the "search for information", often with a strong focus on users, their queries, and result documents, or according to the classic definition by Salton in [98]:

> *Information retrieval is a field concerned with the structure, analysis, organisation, storage, searching, and retrieval of information.*

In the following we give a short overview of these aspects and highlight methods which are specifically relevant to the later chapters of this thesis. While there exist a range of excellent books on the topic of IR, we mostly refer to one particularly helpful and recent one when explaining basics [63].

2.2.1 Research Methods in Information Retrieval

The field of information retrieval is known for a strong focus on experimental evaluation in terms of effectiveness. A wide range of benchmarking initiatives have emerged in that context. TREC is undoubtedly the most well-known of those initiatives [119].

The TREC initiative has been running since 1992 and is an ongoing series of workshops concerned with the evaluation of different aspects of IR. A strong focus lies on providing standardised test collections on which participants can run their algorithms. In general, TREC comprises different tracks, i.e., individual, focused competitions. These range from the Web track to the entity search track, which is most relevant with respects to techniques introduced later in this thesis. All of the tracks provide a data set of some kind (web documents for the Web track, Linked Data for the Entity track), a set of queries (also called topics or topic sets). Every participant sends a ranked list of documents/entities their algorithm has found to be relevant to the queries. Relevance judgements to evaluate against are created by the organisers or participants and made available at a later stage. Then, for each query, some judged documents are available, returned to the participants, and can be compared to the ranked results, according to standard evaluation measures like precision, recall, or Mean Average Precision. These results are then submitted and discussed at the actual workshop.

More recently, TREC has explicitly been focusing entity search [12, 14]. There exist initiatives similar in spirit but with a stronger focus on entity search, such as the Semantic Search evaluation series [20, 46], or related INEX tasks [32, 50].

Benchmarking initiatives play a vital role in information retrieval research, guaranteeing sound experimental evaluation methodology; this ensures the repeatability of experiments.

2.2.2 Information Retrieval Evaluation

Evaluation in IR is typically based on manual assessments of relevant answers to queries. Typically, this will include a list of documents and their relevance judgements for each query. In the simplest case these relevance judgements will be binary, i.e., relevant or not relevant. All IR benchmarking initiatives use such relevance judgements (not necessarily binary though) to assess submissions. Whenever researchers test their algorithms, they compare the rankings against the gold standard.

Given these basics, a range of measures has been proposed to assess the effectiveness of retrieval systems. The most basic and frequent measures are precision and recall. Precision (P) measures the quality of the retrieved results (i.e., how many of the retrieved documents are relevant). Recall (R) measures the coverage of the results (i.e., how large a fraction of all relevant documents in the collection they contain). Precision denotes the fraction of relevant items retrieved and the retrieved items:

$$P = \frac{\#(relevant\ items\ retrieved)}{\#(retrieved\ items)}. \tag{2.1}$$

Recall denotes the fraction of relevant items retrieved and the number of relevant items in the collection:

$$R = \frac{\#(relevant\ items\ retrieved)}{\#(relevant\ items)}. \tag{2.2}$$

Since it is easy to increase recall by returning more documents (e.g., returning all documents of the collection always equals to a recall of 1), it is desirable to not exclusively use it as a quality measure. To combine both measures into one, the F measure has been proposed, combining both precision and recall in one number:

$$F_\beta = \frac{(\beta^2+1)PR}{(\beta^2 P)+R}, 0 \leq \beta \leq +\infty. \tag{2.3}$$

When assigning equal weights to precision and recall we denote this as the F1 measure:[1]

$$F1 = \frac{2PR}{P+R}. \tag{2.4}$$

All three of these measures are set-based, which means they are computed on an unordered set of documents. When evaluating rank-based retrieval results, these measures are insufficient. One way of looking at ranked results is to look at the top k results. Precision and recall can be computed at each such level to give a

[1]Equal weights imply a β value of 1 in the general equation, leading to the name F1.

more complete overview (this is denoted by $P@k$ and $R@k$, respectively). These measures are computed for each query in turn, and then averaged over the whole query set. Especially average precision (AP) is used regularly.

One measure that computes precision at each recall level is (MAP)[2]. It is a single-figure measure for precision at all different recall levels, giving a more holistic view; MAP has become increasingly popular within the TREC community. The MAP for a given set of queries Q is defined as:

$$MAP = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}P(R_{jk}), \tag{2.5}$$

where Q is a set of queries or information needs and m_j is the number of relevant documents for query j. R_{jk} is the set of ranked retrieval results and $Precision(R_{jk})$ the precision at all of these levels. Then the value is averaged over the number of queries. The mean average precision gives a more complete picture of the precision of a given query searched for in a given collection.

Another measure, specifically handling the rank of the first relevant document is mean reciprocal rank (MRR):

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}, \tag{2.6}$$

where Q denotes a set of queries, and $rank_i$ is the rank of the first relevant answer to query i. This is averaged over all queries. A high reciprocal rank means that the first relevant answer is often ranked among the top results.

Another approach that has become increasingly popular is the cumulative gain (or more specifically the Normalised Discounted Cumulative Gain ($nDCDG$). This measure is designed for non-binary relevance judgements. To begin with, we compute a vector of cumulative gain, i.e., the relevance sum at each level k:

$$CG[k] = \sum_{i=1}^{k}G[i]. \tag{2.7}$$

$G[i]$ corresponds to the relevance given (i.e., between 0 and 3, where 0 is not relevant, 1 slightly and so on). In the next step we apply a discount function to reflect the relative lower importance of results at lower ranks. The discounted version DCG using $log_2(1+i)$ as discount function is computed as follows:

$$DCG[k] = \sum_{i=1}^{k}\frac{G[i]}{log_2(1+i)}. \tag{2.8}$$

[2]In other words the average precision across all recall levels.

Table 2.1: Overview of document retrieval notation.

Variable	**Explanation**
t	Term
d	Document
q	Query
$tf_{t,d}$	Term frequency of term t in document d
df_t	Document frequency of term t
N	Number of documents in the collection
idf_t	Inverse document frequency of term t
$\vec{d}$	Document vector of document d
dl	Document length
adl	Average document length
$P(A)$	Probability of event A
$P(A\|B)$	Conditional probability of A given B
$P(d\|q)$	Conditional probability of a document d given a query q
θ_d	Document language model of document d

Finally, these values can be normalised by the ideal DCG vector DCG'. This vector is computed for an ideal ranking where for example all documents with a relevance of 4 are ranked before all documents with a relevance of 3 and so on. This leads to an $nDCG$ value at k of:

$$nDCG[k] = \frac{DCG[k]}{DCG'[k]}. \tag{2.9}$$

$nDCG$ is typically computed as the mean over a set of queries and various retrieval depths such as *nDCG@5* or *nDCG@10*.

None of these measures alone is enough to evaluate a system, and which one is more desirable depends highly on the context (web users are mostly interested in the top 10 results, sometimes only the one highest result is of interest, and usually are not interested in high recall). Thus, in the context of web search measures such as precision at 1 (P@1) are more important. In the course of the experiments performed in this thesis we will mostly use *P@10*, *MRR*, *MAP*, and also list *nDCG* where applicable. This is to show a more complete picture of our results with respects to different requirements in different settings.

2.3 Document Retrieval

Ad-hoc document retrieval aims at adequately ranking documents with respect to their relevance to a given query, e.g., how to compute the similarity between queries and documents. The prevalent technique to do this is based on indexing the documents[3] and terms they contain to later match these representations with queries. Here, we do not consider the ordering or co-occurrences of terms, i.e., we use the bag of words approach.

In the most simple case, each document has binary weights for the terms it contains (i.e., 1 meaning the term does occur, 0 that it does not). This model is called the Boolean Model of Information Retrieval. Its obvious limitations are that the frequency of the terms in the individual documents is disregarded; whether a term occurs once or 100 times in one document does not make any difference with respect to similarity scoring, i.e., documents are scored based on occurrence or not-occurrence of terms alone.

The vector space model takes into account the frequencies of terms in documents ($tf_{t,d}$), the frequency of term t in document d). We provide an overview of variables and symbols used throughout this chapter in Table 2.1. This leads to an improved ranking mechanism. However, in the most basic case we still suffer from biased results for very common terms (when using raw frequencies only, think of the term "abstract" in scientific documents; almost every document will have an "abstract" headline). The document frequency (df_t), on the other hand, denotes the number of documents a term occurs in and would assign a high value to the "abstract" term from before. As such it is a good idea to weigh terms with their inverse document frequency:

$$idf_t = \log \frac{N}{df_t}, \tag{2.10}$$

where N is the total number of documents in the collection. The *idf* values of rare terms will be high;[4] the ones of common terms will be low. This means that a simple combination of *tf* and *idf* values will improve document scoring:

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t. \tag{2.11}$$

This allows us to treat every document in the collection as a vector, the indices of which correspond to the *tf-idf* weighting of terms. This is commonly referred to as the vector space model. Further, we can compute the simple overlap score of a document d given a query q as:

$$score(q,d) = \sum_{t \in q} tf\text{-}idf_{t,d}. \tag{2.12}$$

The similarity measure most commonly used is the cosine similarity between two documents (or document vectors) $\vec{d_1}$ and $\vec{d_2}$ (the documents can easily be substituted with one query q and a document d):

[3]An index is a compact representation which helps to access documents based on the terms they contain.

[4]Some terms, so-called stop words, are skipped at indexing time since they do not convey important information. This comprises extremely common words with little or no discriminative power such as "a", "and", "it" ,"he", or "of."

$$\begin{aligned} sim(\vec{d_1}, \vec{d_2}) &= \frac{\vec{d_1} \bullet \vec{d_2}}{|\vec{d_1}| \times |\vec{d_2}|} \\ &= \frac{\sum_{i=1}^{n} \vec{d_1}_i \times \vec{d_2}_i}{\sqrt{\sum_{i=1}^{n} \vec{d_1}_i^2}\sqrt{\sum_{i=1}^{n} \vec{d_2}_i^2}}. \end{aligned} \tag{2.13}$$

The cosine similarity is then calculated between the query and each of the documents (much like between documents), resulting in a ranking of documents.

2.3.1 Probabilistic Information Retrieval

In this section we will first review basic probability theory and then give an overview of the most influential ideas in probabilistic IR. Generally speaking, the probabilistic model is theoretically sound and can therefore be a viable alternative to vector space approaches.

Probability Theory Revisited

Many of the later chapters will rely on probabilistic models and language modelling techniques. Therefore, we will first give a short introduction to basic probability theory to give a foundation for the more advanced concepts to be described later on.

In general, probability theory is concerned with events and how probable they are to occur. An event can be represented by variable A and its likelihood or probability[5] is denoted by $0 \leq P(A) \leq 1$.

In the more interesting case of two or more events (let us say A and B), the probability of both occurring can be explained with their joint probability $P(A, B)$. The conditional probability $P(A|B)$, on the other hand, denotes the probability of event A given that event B occurred. The relationship between both is given by the *chain rule*:

$$P(A, B) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A). \tag{2.14}$$

The probability of a joint event can be expressed as the probability of one of the events multiplied by the conditional probability of the other. Another important rule is the *partition rule*, explaining that the probability of an event B is the sum of the probabilities of all disjoint subclasses:[6]

$$P(B) = P(A, B) + P(\overline{A}, B). \tag{2.15}$$

[5]The terms likelihood and probability are used as synonyms in this context. Likelihood usually denotes an observed set of events on which probability calculations are based on.

[6]This assumes that event B can be divided into an exhaustive set of disjoint subclasses.

These rules suffice to derive *Bayes' rule* for inverting conditional probabilities:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{2.16}$$

The two most important concepts here are *prior probability* and *posterior probability.* The *prior probability* $P(A)$ provides the probability of an event having no other information. The *posterior probability* of an event $P(A|B)$ denotes its probability after having seen the evidence B.

Probability Ranking

In the context of documents, collections and queries, we can assign a random variable $R_{d,q}$ representing the relevance of a document d given a query q. In the binary case, R has a value of 0 if a document is not relevant and a value of 1 if it is. The main idea behind probability is to rank documents according to their estimated probability of relevance $P(R = 1|d, q)$.

The Binary Independence Model (BIM) The binary independence model introduces some assumptions making the estimation of $P(R|d, q)$ feasible. Both documents and queries are represented by boolean vectors, 1 indicating a term is present in a document/query and 0 indicating it is not. This also means that a term independence assumption is made, reflecting the fact that the model does not recognise associations or relations in between terms. This assumption is far from correct but simplifies the process and provides satisfactory results in other applications such as probabilistic classification. The term independence assumption also correlates to how documents are modelled in the vector space model. The BIM is formalised as:

$$\begin{aligned} P(R = 1|\vec{x}, \vec{q}) &= \frac{P(\vec{x}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{x}|\vec{q})} \\ P(R = 0|\vec{x}, \vec{q}) &= \frac{P(\vec{x}|R = 0, \vec{q})P(R = 0|\vec{q})}{P(\vec{x}|\vec{q})} \end{aligned} \tag{2.17}$$

where $\vec{x}$ denotes the document vector of document x, $\vec{q}$ denotes the query vector of query q. $P(\vec{x}|R = 1, \vec{q})$ is the probability of $\vec{x}$ being the representation of a relevant document being retrieved (this is accordingly expressed for a non-relevant document being retrieved as $P(\vec{x}|R = 0, \vec{q})$). $P(R = 1|\vec{q})$ and $P(R = 0|\vec{q})$ denote the prior probabilities of query q retrieving relevant and non-relevant documents, respectively. Both of these probabilities sum up to 1.

OKAPI BM25

The binary dependence model was initially developed to rank documents of rather consistent length (i.e., the individual documents vary little in size). However, this assumption can not be made in many contexts, and document lengths should be taken into account in the retrieval model. BM25 is designed to handle varying document lengths. A comprehensive overview of the probabilistic relevance framework and BM25 methods is presented in [94]. BM25 uses the average document length component to score documents, distinguishing it from classic *tf-idf* scoring. Further, two internal parameters are needed, b and $k1$.

We assume no relevance information available. As such, a close approximation of the classical *idf* is commonly used, compensating for 0-values:

$$idf_t = \log \frac{N - df_t + 0.5}{df_t + 0.5}. \tag{2.18}$$

Additionally, soft document normalisation is introduced:

$$B := \left((1-b) + b\frac{dl}{avgdl} \right), 0 \leq b \leq 1, \tag{2.19}$$

taking into account the relative length of the current document (the $\frac{dl}{avgdl}$ component, referring of the ratio between the document length dl and the average document length $avgdl$). Setting $b = 0$ will not make use of normalisation at all; $b = 1$ performs full document length normalisation.

Substituting B from Eq. (2.19), we use the sum over all query terms t to score document d as follows:

$$score_{BM25}(q,d) = \sum_{t \in q} \frac{tf_{t,d}}{k1(1 - b + b\frac{dl}{avgdl}) + tf_{t,d}} idf_t, \tag{2.20}$$

where $k1$ is a tuning parameter calibrating the document term frequency scaling (a $k1$ value of 0 corresponds to a binary model, large values for $k1$ correspond to using raw term frequencies).

BM25 has a strong advantage over *tf-idf* based methods because it takes into account the document lengths, but also comes with disadvantages in that it is more difficult to tune the additional parameters $k1$ and b.

Language Models for Information Retrieval

Language models for information retrieval have become a popular and competitive way of performing document ranking. The underlying idea is that a document is a good match for a query if this document is likely to generate the query. The idea of "generating a query" stems from the ability of a finite automaton to generate language. While this is more out of tradition, similarities exist between the two concepts. A language model does put a probability measure over strings drawn from some vocabulary (single terms in the case of unigram language models). The resultant language model is a probability distribution, i.e., it adds up to 1. It should be noted that the event space, i.e., the set of all possible outcomes or occurrences of word sequences, is infinite since any natural language term can occur in any order.

This way, each term in a document is assigned a probability and the probabilities of sequences of terms can be computed simply by multiplying these individual probabilities (assuming a uniform language model ignoring the context of terms):

$$P_{uni}(t_1 t_2 t_3) = P(t_1)P(t_2)P(t_3). \tag{2.21}$$

The query likelihood language model is only one instance of the family of language modelling approaches. In this model, we construct a language model θ_d for each document in the collection. The main goal is to rank documents by $P(d|q)$; the probability of a document is interpreted as the likelihood that it is relevant to the query. Using Bayes' rule from (2.16), we have:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}. \tag{2.22}$$

$P(q)$ is the same for all documents, as such cannot influence rankings, and thus can be dropped. The prior probability of a document $P(d)$ is often treated as uniform over all documents, but could be assigned on a document level reflecting the importance of individual documents. The query likelihood can now be computed as

$$P(q|\theta_d) = \prod_{t \in q} P(t|\theta_d)^{tf_{t,q}}, \tag{2.23}$$

where $tf_{t,q}$ denotes the (raw) frequency of term t in the query.[7] The main open question now is how to estimate the probability of a term given a document language model $P(t|\theta_d)$. It is important to point out that this probability must never be 0 since that would lead to 0 as the total probability (which in turn would make partial matches impossible). This is where smoothing comes into play to guarantee that we never encounter zero-probabilities. We employ Bayesian smoothing using Dirichlet priors which has been shown to achieve superior performance on a variety of tasks and collections [125]:

$$P(t|\theta_d) = \frac{tf_{t,d} + \mu P(t|\theta_c)}{|d| + \mu}, \tag{2.24}$$

where $tf_{t,d}$ is the raw frequency of term t in document d and $|d|$ is the size of a document, i.e., $\sum_t tf_{t,d}$. $P(t|\theta_c)$ represents the global term probability, e.g., the

[7]For short keyword queries this virtually always equals to 1 since they hardly ever contain the same term more than once.

probability of the term occurring in the collection, i.e., $\sum_d \frac{tf_{t,d}}{\sum |d|}$. The smoothing parameter μ is subject to optimisation, but setting it to the average document length is usually a good starting point [59]. Practically, all terms are smoothed by their probability in the whole collection and therefore 0 values are avoided.[8]

Language Models vs. Classical Approaches

The language modelling approach has gained much interest and is theoretically sound and computationally tractable. On the other hand, the relations to traditional *tf-idf* models are significant [48]. The effect of smoothing by collection frequency can be compared to *idf* weighting. Even though LMs provide good performance, comparable or even better than other weightings such as BM25, it is not definitive that this fact is a general one, or that proper parameter tuning can not be used to tune traditional models to similar performance.

2.3.2 Structured Retrieval

Structured retrieval, searching when additional structural information is available, has become an active field within Information Retrieval research. It can denote to both structure in the document representation and the results (e.g., in XML retrieval where the unit of retrieval is parts or passages of documents rather than full documents [62]). In this context we use structured retrieval to refer to document structure in terms of multiple available fields while the unit of retrieval is not varied, i.e., we return entities. Many search scenarios can be presented as such, encompassing, e.g., rather straight-forward applications like search in e-mails (in the simple case where "subject," "from," and "body" fields are available). However, also more complex use-cases like search in the Internet Movie Database (IMDB[9]) can be approached by structured retrieval techniques. In this case it seems more intuitive to use its tens of different fields such as "director" or "releasedate" within the retrieval model to improve overall effectiveness. Other collections with structured information available are WoD collections such as DBpedia[10] where the number of fields quickly increases to several hundred.

Usually, structured retrieval is approached by fielded extensions of the retrieval models discussed in the previous subsection, such as BM25 and the language models (LM); we present these extensions in the next two subsections.

BM25F

BM25F is an extension of BM25 incorporating multiple fields [95]. In this case, both the soft normalisation B and tf_d need to be adjusted. We first compute a

[8]This assumes that terms which are not available in the collection are not considered and dropped while the query is parsed.
[9]`http://www.imdb.com/`
[10]`http://dbpedia.org`

normalised term frequency:

$$\widetilde{tf}_{t,d} = \sum_{f=1}^{F} tf_f \, \frac{tf_{t_{df}}}{B_f}, \tag{2.25}$$

where $tf_{t_{df}}$ denotes the term frequency of term t in field f of document d. The soft normalisation for a field f is given as:

$$B_f = \left((1 - b_f) + b_f \frac{dl_f}{avgdl_f}\right), 0 \leq b_f \leq 1. \tag{2.26}$$

In many cases, one b_f value is used for all fields. This is due to both the computational complexity and theoretical difficulty that a per field soft normalisation bring along (this is in addition to the field weight, i.e., would lead to two free weighting parameters per field). Rather than manually setting one b value per field, it is possible to adjust the b_f value based on the field's relative average length:

$$b_f = \frac{avgFl_f}{\sum_{i \in N} avgFl_i}. \tag{2.27}$$

The overall score for query q and document d is then given as:

$$score_{BM25F}(q, d) = \sum_{t \in q} \frac{\widetilde{tf}_{t,d}}{k1 + \widetilde{tf}_{t,d}} idf_t. \tag{2.28}$$

Mixture of Language Models (MLM)

Analogous to BM25F, language models allow for an estimation of individual language models per field [87]. These models can then be used together by a linear combination. A separate language model θ_d^f is estimated for each field f (associated with the given document):

$$P(t|\theta_d^f) = \frac{tf_{t,f,d} + \mu_f P(t|\theta_c^f)}{|f, d| + \mu_f}, \tag{2.29}$$

where $tf_{t,f,d}$ is the term frequency for t in field f of document d, and $|f, d| = \sum_t tf_{t,f,d}$. Essentially, we apply Dirichlet smoothing using a collection-wide background model $P(t|\theta_c^f)$ (that is, a maximum-likelihood estimate for a field from all documents in the collection). The smoothing parameter μ_f is set to the average field length in the collection.

The document model is a linear mixture of the field type language models ($P(t|\theta_d^f)$), weighted with the importance of that field type ($P(f)$):

$$P(t|\theta_d) = \sum_f P(t|\theta_d^f) P(f). \tag{2.30}$$

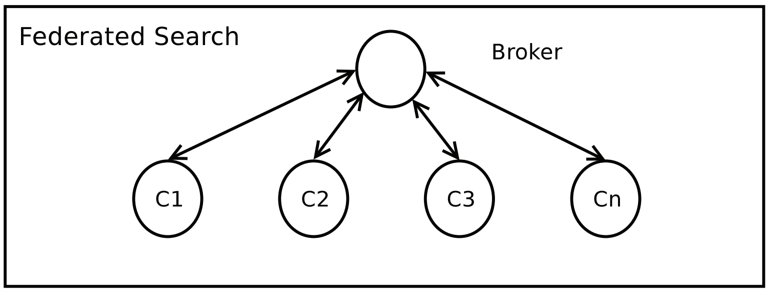


Figure 2.1: Example of a federated search setting.

Both BM25F and MLM show competitive performance in a range of settings [26, 53, 54, 87, 90, 95]. Again, a slight advantage of language models is their reasonably good performance with the standard settings (average field length and average document length as smoothing parameters).

2.4 Search in Distributed Environments

Search in distributed environments can broadly be categorised into broker-based and peer-to-peer (P2P) architectures. Federated search is typically broker-based, i.e., a central broker exists to perform the main tasks such as collection selection and result merging. An example of a broker-based architecture is given in Figure 2.1. P2P search, on the other hand, commonly has no central hub and each collection (or rather peer) is a valid entry point for search operations. In the following we continue by describing the basics of federated search and outline the foundations and special characteristics of P2P search as examples of both architectures.

2.4.1 Broker-based Architectures: Federated Search

Distributed IR is commonly called federated search. The assumption in this area is that—in comparison to the centralised use case—the document collection is stored in multiple locations and it is not feasible to copy all documents to a central location. The reasons for this might range from the sheer size of the collection to legal regulations not allowing documents to be processed outside a company's infrastructure. An excellent and up-to-date survey paper on federated search is given in [101].

Instead of expending effort to crawl all Web of Data sources—some of which may not be crawleable at all—*federated search* techniques directly pass the query to the search interface of multiple, suitable collections that are usually distributed across several locations [101]. For example, the query "entity retrieval" may be passed to a related collection, such as a bibliographical database for research articles dealing with information retrieval topics, while for the query “San Antonio” collections containing information about the city, such as geonames or DBpedia, might be more appropriate. Of course, there are also queries for which multiple databases can contain answers.

Federated search consists of three components:

1. Collection representation
2. Collection selection
3. Result merging

For each query which is sent to the system, we iterate all three steps. First, all the collections are ranked according to how likely they will contain relevant results to the query. Then, we need to select a small number of these collections. In order to comply with efficiency requirements this might range from very few to tens of collections. Once these collections are identified, the query is propagated to the selected collections and the broker needs to decide how to merge the results returned from each of them.

We adhere to these three steps in Chapter 5, where we provide a system for federated entity search. In the following, we discuss baseline techniques for these three steps.

Collection representation is concerned with how collections are represented at the central broker and as such plays a vital role in the process of collection ranking (in our context this step is dealt with by collection/entity modelling).

Collection Ranking and Selection

When deciding which collections to search in the first step is to rank all available collections according to some criterion. Then, some of these collections are selected as relevant sources (most commonly in terms of a fixed cutoff). Each collection is hereby treated as one big document (lexicon-based) or as consisting of individual documents (document-surrogate). We outline the common techniques for both approaches in the following.

In lexicon-based approaches, for example CORI, each collection is considered to be one big bag of words. Collections are ranked according to the similarity between a query and this representation.

CORI One of the best-known examples of lexicon-based collection selection is the CORI algorithm [24]. It uses a Bayesian inference network model and an adapted Okapi term frequency normalisation. The belief of the ith collection being associated with term t is calculated as:

$$\begin{aligned} T &= \frac{df_{t,i}}{df_{t,i} + 50 + 150 \times cw_j / avg\_cw} && (2.31) \\ I &= \frac{log(\frac{N_c + .5}{cf_t})}{N_c + 1} && (2.32) \\ P(t|c_i) &= b + (1 - b) \times T \times I, && (2.33) \end{aligned}$$

where $df_{t,i}$ is the document frequency of term t in collection i, cf_t the number of collections containing term t, N_c the number of collections. cw_i is the size of collection i in terms of collection length and avg_cw is the average length of all collections. The default belief b is usually set to 0.4. The resulting belief $P(Q|ci)$ is used to rank collections, commonly computed as the average beliefs of all query terms.

Document-surrogate methods were designed to be used in uncooperative environments, where central statistics are not available (they are still applicable to the cooperative use case). Document-surrogate methods also take into account a ranking of sampled documents from each of the individual collections.

ReDDE ReDDE (relevant document distribution estimation) [104] has the goal to select a small number of collections containing the most relevant documents by estimating the number of relevant documents per collection and then using this information for collection ranking. The number of relevant documents for query q in collection c is estimated by:

$$\mathcal{R}(c, q) = \sum_{d \in c} P(\mathcal{R}|d) P(d|c) |c|. \tag{2.34}$$

In this case $P(d|c)$ is the prior probability of a document in collection c and $P(\mathcal{R}|d)$ is the estimated relevance probability for document d. Sampling can be used to avoid computing relevance for all documents in all collections. The probability of a document being relevant is approximated with respect to its position in the ranked list of sampled documents. ReDDE uses a constant positive probability α for the top-ranked documents:

$$P(\mathcal{R}|d) = \begin{cases} \alpha, & \text{if } r_{CCI}(d) < \beta \sum_i |c_i|. \\ 0, & \text{otherwise} \end{cases} \tag{2.35}$$

$r_{CCI}(d)$ is the rank of document d in the ranking of all documents for collection i (this can be approximated by using the ranks of the sampled documents). β is a percentage threshold, set to 0.003 in prior research. A final goodness value is then computed by:

$$Goodness(c, q) = \frac{\mathcal{R}(c, q)}{\sum_i \mathcal{R}(c_i, q)}. \tag{2.36}$$

A collection is ranked according to the number of relevant documents it contains in the sample at the broker.

Centralised-rank Collection Selection Method (CRCS) As is the case with ReDDE, CRCS uses a centralised index of all sampled documents (CSI) to rank the collections [100]. One main difference to ReDDE, however, is that CRCS considers varying importance for documents according to their ranks. The contribution of a sampled document d depends on its position in the central ranking of all sampled documents, in the linear case, this results in:

$$P(R|d) = \begin{cases} \gamma - r_{CSI(d)} & \text{if } r_{CSI}(d) < \gamma. \\ 0, & \text{otherwise} \end{cases} \tag{2.37}$$

The parameter γ specifies how many top-ranked documents in CSI are considered, this was set to 50 in [100], where also an exponentially decreasing variant was introduced. r_{CSI} denotes the rank of document d in the CSI. The final goodness of each collection is calculated similarly to ReDDE:

$$Goodness(c, q) = \frac{|c_i|}{|c^{max}| \times |S_c|} \times \sum_{d \in S_c} R(d), \tag{2.38}$$

where the (sampled) collection sizes are normalised by the fraction of the size of each collection ($|c_i|$) and the maximum size of all involved collections ($|c^{max}|$) weighted by the number of documents sampled from collection c ($|S_C|$).

SUSHI In the spirit of ReDDE and CRCS, SUSHI first ranks the documents in CSI [112]. Then, SUSHI extracts the document ranking for each server in turn. After that, adjusted ranks are computed compared on the sizes of the individual collections. Curve fitting is then used to estimate the scores of unseen documents.

Result Merging

The third component of federated search systems is concerned with merging the results returned from the different collections. This becomes relevant after the collection selection step when all of the selected collections return ranked lists for the given query. There exist a range of techniques which can either be based on the scores assigned by the individual collections or based on the rank of the documents within these local rankings. CORI merging, for example is based on scores and applies a straightforward min-max normalisation before using a linear combination of collection selection scores and document scores. Other approaches include semi-supervised learning in [105] or using statistical fit [102].

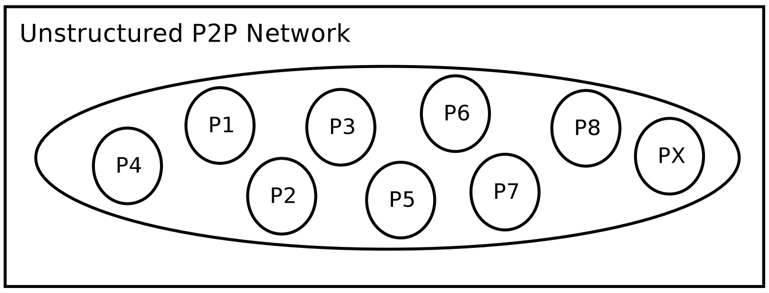


Figure 2.2: Example of an unstructured P2P network.

2.4.2 Search in P2P Networks

All the techniques described in the federated search scenario of the last section rely on a central broker, which has the task of coordinating search requests and to route each query to the collection best suited. However, this assumption can not always be made. Be it for privacy or sheer performance reasons, when a central broker might not always be able to handle the workload created by the network. P2P systems constitute a fundamentally different paradigm. Rather than requesting information via a central broker, all collections (or rather nodes or peers) can act as both client and servers in the network. The best-known example for a P2P application can be found amongst filesharing networks such as Napster.

Federated Search and P2P Search

In P2P, we change the scenario from the few nodes in the federated search context to hundreds or thousands. The typical federated search scenario consists of a few collections and a broker, which performs collection selection, query routing, and result merging. It is also the one link between all collections and as such clearly the bottleneck in terms of network load. On the other hand, communication goes over a maximum of two hops (e.g., between the broker and collection *C1* to *Cn*). An overview of this architecture is shown in Figure 2.1.

We show an overview of an unstructured P2P network in Figure 2.2. Instead of collections the network consists of nodes or peers (these terms are equivalent to the extent we investigate the issue here). For peer *P1* to communicate with peer *P8*, e.g., it is up to the self-organisation of the network to determine the communication path. Furthermore, each peer is a possible entry point for query routing, i.e., queries can originate from each node.

The two main types of P2P networks are structured and unstructured networks. In the former case, a globally consistent routing protocol exists, ensuring that each file (i.e., each document) can be found by all peers. Distributed hashtables are an example for this. In the latter case, the network is completely unstructured, i.e., in principle all peers in the network need to be accessed in order to find a particular document. The resource intensity of this so-called flooding process can be mitigated by using various types of overlay or meta networks within. In that case, each overlay is responsible for all peers in its network and it can suffice to contact one of the peers of the overlay network to find a document.

P2P Document Retrieval

Content-based search in P2P networks [97] is usually related to full-text search [60, 111, 126], with most approaches relying on the use of structured P2P networks. Some research focuses on providing P2P web search functionalities, like in [70], where MINERVA is presented, a P2P web search engine that aims at scalability and efficiency. In MINERVA, each peer decides what fraction of the web it will crawl and subsequently index. In further work, the authors also presented an information filtering approach relaxing the common hypothesis of subscribing to all information resources and allowing users to subscribe to the most relevant sources only [128].

Previous approaches regarding P2P web search have focused on building global inverted indices, as for example Odissea [109] and PlanetP [29]. In PlanetP, summaries of the peers' inverted indices are used to approximate TF-IDF. Inverse peer frequency (the number of peers containing the term) is used instead of IDF. It is questionable how this would scale in large P2P networks with dynamic contents, as also noted in [4]. In [6], super-peers are used to maintain DF for the connected peers. A similar approach is also used in [85]. Bender et al. study global document frequency estimation in the context of P2P web search in [18]. The focus is on overlapping document collections, where the problem of counting duplicates is immense. Their system relies on the use of an underlying structured P2P network. A similar approach is described in [88], which is quite different from our setup that assumes an unstructured P2P architecture.

A major shortcoming of all these approaches is that their efficiency degrades with increasing query length and thus they are inappropriate for similarity search. Recently, an approach has been proposed that reduces the global indexing load by indexing carefully selected term combinations [107].

The overview of an integrated system for P2P search is given in [96]. The authors propose a distributed architecture for P2P information retrieval called PHIRST. In this system the storage costs per node are reduced considerably by storing only a limited number of terms. Subsequently, a hybrid search model of both structured and unstructured search is employed at query time, for not all terms are stored within the system. However, their algorithms were tested on a test collection of moderate size and fall back to unstructured search.

P2P-based Digital Libraries (DL)

One area of research combining and applying several of the techniques introduced here are digital libraries. Several papers propose using P2P networks in a digi-

Table 2.2: Overview of term selection methods.

Term selection method	**Acronym**	**Equation**
Document frequency	DF	df_t
Collection frequency	CF	$\sum_{i=1}^{N} tf_{t,d_i}$
Collection freq. inverse document freq.	CFIDF	$CF_t \log \frac{N}{df_t}$
Term frequency document frequency	TFDF	$(n_1 n_2 + c(n_1 n_2 + n_2 n_3))$

tal library context [4, 37, 38, 91, 93]. In [5], a distributed indexing technique is presented for document retrieval in digital libraries [91]. Podnar et al. use highly discriminative keys for indexing important terms and their frequencies. In [93], the authors present *iClusterDL*, for digital libraries supported by P2P technology, where peers become members of semantic overlay networks (SONs).

Background on Term Selection

Term or feature selection denotes the process of selecting a subset of all available terms for further processing. These methods can generally be categorised as either supervised or non-supervised. Supervised methods use provided labels or class assignments for documents. The best or most discriminating terms are then selected according to their class labels and the occurrence of the term across classes (in the 20 newsgroups collection, for example, these labels indicate the newsgroup an article originally was posted to). In many cases, however, class labels are not available. In the context of distributed collections, such labels are particularly rarely available, due to reasons of missing common document types or the general ad-hoc character of the collections themselves. To perform term selection nevertheless, unsupervised techniques—even though there exist fewer than supervised ones—can be used. These methods mainly rely on frequency information of a term or term within a collection, in order to judge its usefulness.

Term Selection Methods Following the vector space model of information retrieval we use N as the number of documents in a collection (which can be either global, i.e., the whole collection, or local when only a subset of the collection is considered). Further we use df_t for the number of documents a term occurs in, also called the document frequency of term t. The number of occurrences of term t in document d is denoted to as the term frequency $tf_{t,d}$. In this context, we propose the usage of the unsupervised methods summarised in Table 2.2, as possible local term selection methods on each peer.

Document Frequency (DF). One of the most prevalent techniques is denoted as document frequency thresholding, i.e., the number of documents a certain term occurs in. The main assumptions underlying document frequency thresholding are that terms occurring in very many documents carry less discriminative information and that terms occurring only in very few documents will provide a strong reduction in dimensionality (even though they might be discriminative in some cases). In combination with an upper and lower threshold, term selection can be applied. This generally leads to results comparable to supervised techniques.

Collection Frequency (CF). The collection frequency of a term is given by the sum of all term frequencies for a given term (the total number of occurrences of a term in a collection):

$$CF_t = \sum_{i=1}^{N} tf_{t,d_i}. \tag{2.39}$$

Therefore, the collection frequency ranks highly terms which might occur only in few documents of the collection but have a high frequency in these documents.

Collection Frequency Inverse Document Frequency (CFIDF). The CFIDF is given by weighting the collection frequency values by the inverse document frequency of a term:

$$CFIDF_t = CF_t \log \frac{N}{df_t}. \tag{2.40}$$

This measure covers both aspects, the local document frequency and the total number of occurrences for a term.

Term Frequency Document Frequency (TFDF). Another, quite recent technique to exploit both the *tf* and *df* factors is presented in [122]:

$$TFDF_t = (n_1 n_2 + c(n_1 n_2 + n_2 n_3)), \tag{2.41}$$

where n_1 denotes the number of documents in which t occurs, n_2 the number of documents t occurs only once, and n_3 the number of documents containing t at least twice. An increasing weight c gives more weight for multiple occurrences. The setting the authors recommend to use because it gave the best results in their experiments is $c = 10$.

In [120], the authors examine the estimation of global term weights (such as the document frequency of a term, i.e., the number of documents a term occurs in for a given collection) in information retrieval scenarios where a global view of the collection is not available. Two alternatives are studied: either sampling documents or using a reference corpus independent of the target retrieval collection. In addition, the possibility of pruning term lists based on frequency is evaluated. The results show that very good retrieval performance can be reached when just the most frequent terms of a collection (an extended stop word list) are known, and all terms which are not in that list are treated equally. In this chapter, we do not consider how to actually determine (collect) and distribute this information. We will return to this particular problem in Chapter 7, when we turn our attention to

P2P search. We want to note that this research is directly applicable to the probabilistic model when document frequency estimation is substituted by collection frequency estimation.

2.5 Semantic Search

The vision of the Semantic Web is a machine-readable and machine-understandable version of the Web. Machine-readability can be guaranteed by common standards such as XML on the syntax level, or RDF on the data interchange level. The vision of the Semantic Web goes beyond mere data formats and includes layers for taxonomies and ontologies for better understanding of that data (inference models are an additional extension of these elements). Once this web is interconnected and semantically annotated (as envisioned in the WoD), many of today's tasks can be solved better and performing complex tasks will be better supported.

In our context, we mainly target search over RDF data and semantic search usually denotes exploiting the structure of the RDF graph. This means searching for related documents and considering the relationships between documents. One technique suggested for this type of search is SPARQL (SPARQL Protocol and RDF Query Language), a query language for RDF developed by the W3C consortium.[11] SPARQL allows for the specification of attributes and relation as part of a query. However, this makes the query syntax much more complex (especially when compared to straightforward keyword queries). We consider this quite a disadvantage for non-technical users, similar to the complexity of SQL queries (with which SPARQL shares common syntax elements). One of the main goals in our research is to find a viable compromise between the power of languages like SPARQL and the simplicity of keyword queries users expect to be able to issue. For this reason the focus of the later parts of this chapter is on Information Retrieval, more specifically ad-hoc (keyword) search. Many times, semantic search refers to search in data which is organised by means of an ontology. This ontology describes classes and hierarchies of objects and can additionally be used for inference (i.e., resolving existing and discovering new relationships based on an ontology).

On a more general level, search and retrieval has been the subject of intense research over decades. In general, we denote as search every user interaction with computer systems with the goal of finding information. Most commonly this is done by issuing some kind of textual query. The best-known search use case is web search, in the context of using any one of the large commercial search engines like Google, Yahoo, or the Microsoft equivalent Bing. The interfaces to all of these are quite similar in their simplicity. Users enter simple keyword queries (i.e., queries consisting of mostly only few keywords).

However, search interfaces can differ from that use case significantly. Not all scenarios are as unrestricted as web search. Search in libraries for example is different

[11] `http://www.w3.org/2001/sw/DataAccess/`

in that users want a certain set of fields to search in (ISBN, author, title), e.g., [43]; the same is true for search in the biomedical domain where users target chemical compounds rather than general information about a topic, an example of concept-based retrieval in the biomedical domain is given in [113].

Some work has been done in the area of user interfaces for semantic full text search. In [17], for example, a rich, interactive user interface for semantic search is presented. Starting from keyword queries, the system incorporates facets, proposals and breadcrumbs (location based clues). This work shows that supportive user interfaces can help users to refine their queries starting from keyword search. We see this line of research as complementary to improvements in ad-hoc search. In a more recent position paper semantic full-text search is advocated in [16].

We want to stress the importance of traditional retrieval models and the potential benefits of combining them with semantic technologies. In the context of this thesis, semantic search covers techniques going beyond simple keyword matching and considers contextual information and other extrinsic information. One very common semantic technique is for example word sense disambiguation, i.e., finding the right sense of a word in a given context (do users search for the Jaguar car or for the animal when simply entering "jaguar"?). Another one is to take into account synonyms and to not only present matches for the search term but also its synonyms (when users search for "bank" documents containing "financial institute" might be good matches too).

2.6 Entity Search

Entity search has been gaining increasing attention in the research community, as recognised by various world-wide evaluation campaigns. The TREC Question Answering track focused on entities with factoid questions and list questions (asking for entities that meet certain constraints) [119]. The TREC 2005–2008 Enterprise track [10] featured an expert finding task: given a topic, return a ranked list of experts on the topic. The TREC Entity search track ran from 2009 to 2011 [12], with the goal of finding entity-related information on the web, and introduced the related entity finding (REF) task: return a ranked list of entities (of a specified type) that engage in a given relationship with a given source entity. Between 2007 and 2009, INEX also featured an Entity Ranking track [32]. There, entities are represented by their Wikipedia page, and queries ask for typed entities (that is, entities that belong to certain Wikipedia categories) and may come with examples. Most recently, the Semantic Search Challenge (SemSearch) ran a campaign in 2010 [46] and 2011 [20] to evaluate the ad-hoc entity search task over structured data. The main task here is *ad-hoc entity retrieval*: "answering arbitrary information needs related to particular aspects of objects [entities], expressed in unconstrained natural language and resolved using a collection of structured data" [92].

Commonly, the ad-hoc entity retrieval task is approached by adapting standard document retrieval methods. A textual representation ("pseudo document") is built for each entity, and these representations can then be ranked using conventional IR models. The main challenge, of course, is how to obtain these textual representations from structured data. WoD conceptually forms a large, directed, labelled graph with nodes corresponding to entities and edges denoting relationships, and is described in the form of subject-predicate-object (SPO) triples of the RDF data model; Figure 1.3 shows a small excerpt from an RDF graph centred around a given entity.

A natural solution would be to represent each entity using a fielded structure, where fields correspond to predicates (i.e., arrows on Figure 1.3) and associated nodes (or rather, the text extracted from them) are used as field values. These representations can then be ranked using any fielded document retrieval model, such as BM25F [95] or the mixture of language models (MLM) [86]. However, with this approach the number of document fields soon becomes computationally prohibitive, making the estimation of field weights intractable. A commonly used workaround is to group predicates together into a small set of predefined categories, and as such, create documents comprising of only a handful of fields. This grouping (or "predicate folding") can be based on, for example, the type of predicates (attributes, in/out-relations, etc.) [90] or on their (manually determined) importance [21]. This leads to a data model where the optimisation of field weights is easily tractable, even using exhaustive search over the parameter space. While this approach seems to work well in practice, it seriously limits the semantic expressiveness of entity models, as it is no longer possible to access the content of individual predicates or might be too dependent on the data collection.

Part II

Entity Search

Having covered the basics, we continue by introducing the main focus of this thesis in Chapter 3 which is the ad-hoc entity search task (i.e., search for entities in the WoD). Therein, we give a thorough introduction to the problems we are concerned with and describe the basic techniques the later chapters will build on. In Chapter 4, we continue to describe more advanced models and incorporate semantic aspects in terms of structuring predicates by their types to improve retrieval efficiency.

Chapter 3

Entity Search in the Web of Data

In this chapter we give an overview of the task of entity search, and our participation in benchmarking initiatives. Our investigations in the field of entity retrieval started with our participation in the 2010 edition of the Semantic Search Challenge for which we studied the entity search and list search tasks. We first introduce the entity search task, based on an extension of [13]. This line of research was followed further in [27]. Further, our observations gave rise to specific research questions related to entity modelling which will be picked up in Chapter 4.

3.1 Introduction

Search for entities has become the most popular type of web search, second to navigational queries [92]. As such, the search for entities has attracted considerable amounts of research interest. We introduce approaches to both the classical entity search and list search (e.g., the "Arab states of the Persian gulf", introduced in 1.3, is a typical list search query). entity search task. With respect to the list search task we attempt to model human user behaviour when searching in Wikipedia. Both methods were evaluated in the Semantic Search Challenge of the Semantic Search 2011 Workshop. We then put our results in context with the other teams' results for the challenge tasks. Entity search denotes searches targeting entities instead of documents. Contrary to search in the Web of Documents, we search for entiteis and do so in RDF (Resource Description Framework) data or other types of structured data representation. Such structured representations, which provide the directions and types of links between entities, are often referred to as "Semantic Web" as envisioned by Tim Berners Lee [19].

At the same time, there is an increased amount of information published as Linked Data that is inherently organised around entities; each entity is identified by a unique URI and is described using a set of subject-predicate-object RDF triples. Querying these structured data sources by means of simple keyword search (as opposed to SPARQL-like languages) emerged as a genuine user need and has recently become an active topic of research [20, 21, 26, 90, 92]. The tasks we are studying in this chapter is *ad-hoc entity retrieval* (often referred to as *semantic search*) which we introduced in Section 2.6: "answering arbitrary information needs related to particular aspects of objects [entities], expressed in unconstrained natural language and resolved using a collection of structured data" [92], and list search, i.e., find a set of relevant answers for a given query.

In the context of semantic search this means that the classical information retrieval keyword search is extended by using RDF input data in the form of *(subject, predicate, object)*, where each component is described by a URI (Uniform Resource Identifier). Entities are represented by subjects and occur together with predicates and objects closer identifying this entity. For example the RDF triple *example.org/NTNU, example.org/hasLocation, example.org/Trondheim* implies that NTNU is located in Trondheim.

In this chapter we show our approaches to entity search and summarise our participation in and the results of the Semantic Search Challenge of the Semantic Search 2011 Workshop providing comprehensive evaluation of our approaches, using both the Billion Triple Challenge (BTC) and DBpedia[1] data sets. Overall we achieved the third place for entity search and the first for the list search task.

Our main emphasis for the entity search was on combining evidence from multiple knowledge sources, where each source is queried using a retrieval method tailored to its specific properties. With respects to list search, our goal was to mimic the behaviour of humans searching in Wikipedia for we believe much of the answers to list queries is available there, albeit not directly accessible. Finally, for both tasks, we exploited "sameAs" links extracted from DBpedia.

In the remainder of this chapter we first survey related work in Section 3.2. Next, we introduce the Semantic Search Challenge in Section 3.3. Then, in two largely independent sections, we discuss our approaches to both entity search and list search in Sections 3.4 and 3.5, respectively. Further, we give an overview of the results of the challenge in Section 3.6, putting our approaches in context with other submissions. We conclude and outline future directions in Section 3.7.

3.2 Related Work

The problem of providing natural language interfaces to semantic data, the area also addressed by this work, is currently in the centre of research attention, with several prototype systems already in existence. Some were built for a specific domain (e.g., [41]), others are domain-independent [51]. The dominant approach is to transform a user's keyword query to a formal semantic query by matching query segments

[1]`http://dbpedia.org`

to triples from the knowledge base [28, 51, 110]). One of the main challenges is the task of mapping segments of a free-text keyword query to entities of a given knowledge base (resources or an ontology). Different strategies to this task can be found in the literature, from pattern-matching and bag-of-words to deep linguistic analysis [114]. Work related to the problem of keyword queries analysis also include annotation of the free text with resources from a knowledge base [72], segmentation of the keyword queries [45], as well as semantic query suggestion [68].

Learning from the user interaction can be employed in order to improve performance of the semantic ad-hoc retrieval system, cf. Lopez et al. [57] and later by Damljanovic et al. [31], who extend work from [110] by allowing user feedback, query refinement, and query expansion. In contrast to most other systems, PowerAqua [58] (building on [57]) is able to work with multiple heterogeneous ontologies. It transforms the input keyword query into the intermediate triple form, similarly to the principle of other approaches, the intermediate format is then mapped to the candidate entities in distinct ontologies.

The authors in [71] and [92] study real query logs of major search engines from a semantic search point of view; their study allowed for classification of semantic query types and for definition of ad-hoc object retrieval from semantic data. This includes the ad-hoc semantic type retrieval that is the main focus of this chapter and is largely uncovered by the previous work. A methodology for the evaluation of ad-hoc retrieval is discussed in detail in [46].

The importance of the ad-hoc retrieval from the semantic data to the research community is evident also from the growing number of research challenges targeting this task, often from slightly different perspectives. The challenges include TREC Entity Track,[2] SemSearch challenge[3] and QALD challenge.[4]

Finally, we also mention that an effort similar to the ad-hoc semantic search is building natural language interfaces to databases. Here, instead of the semantic knowledge base, data is retrieved from relational schemas (cf., e.g., [73]).

3.3 Data Collection

The data collection used is the Billion Triple Challenge 2009 corpus; it comprises about 1.14 billion RDF statements collected by a Semantic Web crawler.[5] We address the following tasks: a) ad-hoc entity search: given a keyword query, targeting a particular entity, provide a ranked list of relevant entities, identified by their URIs; and b) list search: return a list of possibly all relevant results for a given query. The data set, queries, and relevance assessments are publicly available.[6]

[2]`http://ilps.science.uva.nl/trec-entity`
[3]`http://semsearch.yahoo.com`
[4]`http://www.sc.cit-ec.uni-bielefeld.de/qald-1`
[5]`http://km.aifb.kit.edu/projects/btc-2009/`
[6]`http://km.aifb.kit.edu/ws/semsearch{10|11}`

Table 3.1: BTC corpus statistics.

(a) Top 10 most frequent domains in the data collection.

Domain	**Frequency**
dbpedia.org	403,490,100
livejournal.com	177,194,000
rkbexplorer.com	155,367,100
geonames.org	131,639,700
mybloglog.com	101,977,700
sioc-project.org	82,271,100
qdos.com	35,620,700
kanzaki.com	35,259,400
hi5.com	33,224,700
dbtune.org	25,373,800

(b) Top 10 most frequent properties in the data collection.

Property	**Frequency**
dbpedia:wikilink	156,434,900
rdf:type	143,479,200
rdfs:seeAlso	53,852,300
foaf:knows	35,786,400
foaf:nick	32,979,500
foaf:weblog	23,239,200
dc:title	22,356,700
akt:has-author	19,541,900
sioc:links_to	19,228,400
skos:subject	18,280,600

Data set. The data collection used is the Billion Triple Challenge 2009 data set. It was mainly crawled during February/March 2009 and comprises about 1.14 billion RDF statements and describes entities from domains like *dbpedia.org*, *livejournal.com* or *geonames.org*.[7] In our evaluations we considered the 500 most frequent predicates. An overview of the top frequent domains along with their counts in terms of how many triples they occur in are shown in Table 3.1(a). We further show the top RDF properties in Table 3.1(b).

Topics and relevance assessments.

The 2011 search challenge consisted of two separate tasks:

- Entity search
- List search

The entity search task is a continuation of the previous year's task where participating teams are given keyword queries and the goal is to find the most relevant entities with respect to one particular entity (e.g., "YMCA Tampa") [46]. Two topic sets are available, consisting of 92 and 50 keyword queries for years 2010 and 2011, respectively. The queries were sampled from web search engine logs.

The list search task, on the other hand, contains more complex queries matching multiple entities (e.g., "Arab states of the Persian Gulf"). This is a task similar in spirit to the List Completion problem at the INEX Entity Ranking track [35] and to the Entity List Completion task of the TREC Entity track [14]). Each team was allowed to submit three runs, i.e., different setups, to the challenge. The best out of these would then be used to determine the teams' rankings. More details on the search challenge can be found in [20]. 50 list search queries were made available for the 2010 benchmark.

[7]`http://km.aifb.kit.edu/projects/btc-2009/`

Relevance judgments were obtained using Amazon's Mechanical Turk. Human assessors were presented with a simplified HTML summary of entities and had to judge relevance on a 3-point scale (excellent, fair, and irrelevant).

3.4 Entity Search Task

In this section we outline our approach to the entity search track. This corresponds to answering queries that refer to one particular entity.

We decided to incorporate both the given BTC collection and DBpedia data. There are two reasons for this: a) we want to put a stronger focus on DBpedia data because of its relative importance within the BTC collection, and b) a DBPedia dump can be considered "cleaner" and includes more additional information such as categories and template information. In general, we try to account for both short and long representations as we consider this an important aspect of user behaviour.

3.4.1 Retrieval Model

We formulate the entity search problem as follows. We rank candidate entities (e) according to their probability of being relevant given the query q: $P(e|q)$. Instead of estimating this probability directly, we use Bayes' rule and rewrite it to:

$$P(e|q) = \frac{P(q|e) \cdot P(e)}{P(q)}. \tag{3.1}$$

Next, we drop the denominator as it does not influence the ranking of entities. The term $P(e)$ could be used to express the *a priori* belief that an entity is relevant (to any query); in this work, we assume this probability to be uniform. Hence, we rank entities according to $P(q|e)$.

In order to incorporate multiple representations, to estimate $P(q|e)$ we consider a linear combination of three different entity representations: based on name only (N), based on DBpedia (D), and based on the BTC collection (B):

$$P(q|e) = \lambda_N P_N(q|e) + \lambda_D P_D(q|e) + (1 - \lambda_N - \lambda_D) P_B(q|e). \tag{3.2}$$

We take $P_N(q|e)$ to be either 0 or 1, based on strict string matching between the query and the name of the entity. $P_D(q|e)$, and $P_B(q|e)$ are estimated using a (fielded) Language Modelling approach. The mixture weights λ_N and λ_D were set to correspond with the importance of the individual sources. Intuitively, we assigned the following weights: $\lambda_N = 0.5$ and $\lambda_D = 0.3$. We detail the computation of these components in the next section.

3.4.2 Entity Representations

We assume the following two types of entity representation.

Name-only Representation For each entity, we collected all its name variants from DBpedia. Let e_N denote the set of name variants that belongs to e. Based on this representation we make a binary decision:

$$P_N(q|e) = \begin{cases} 1, & \exists n \in e_N : match(n,q) \\ 0, & \text{otherwise,} \end{cases} \tag{3.3}$$

where $match(n,q)$ is a strict, case insensitive string matching function that returns true iff n equals to q.

DBpedia Representation We rank entities in DBpedia using a fielded Language Modelling (LM) approach. Each entity e is represented as a multinomial probability distribution over terms: e. The likelihood of the query given this model is then computed as a product of individual term probabilities:

$$P(q|\theta_e) = \prod_{t \in q} P(t|\theta_e)^{n(t,q)}, \tag{3.4}$$

where $n(t,q)$ denotes the number of times term t occurs in the query. So far, this approach equals to the standard LM approach. We deviate from it in the estimation of the entity language model θ_e:

$$P(t|\theta_e) = \sum_{f \in \mathcal{F}} P(t|\theta_{e_f}) \cdot P(f), \tag{3.5}$$

where $\mathcal{F}$ is the set of DBpedia fields considered, $P(t|\theta_{e_f})$ the term's probability given a specific field f, and $P(f)$ is the importance of that field. We estimate field-specific term probabilities as a linear combination of field-level and entity-level term probabilities, both smoothed by Dirichlet priors:

$$P(t|\theta_{e_f}) = (1 - \lambda_e) \cdot P(t|e_f) + \lambda_e \cdot P(t|e), \tag{3.6}$$

where

$$P(t|e_f) = \frac{tf_{t,e_f} + \mu_f \cdot P(t|\theta_{c_f})}{|e_f| + \mu_f}, \tag{3.7}$$

and

$$P(t|e) = \frac{\sum_f tf_{t,e_f} + \mu \cdot P(t|\theta_c)}{\sum_f |e_f| + \mu}. \tag{3.8}$$

The components of Eq. (3.7) and Eq. (3.8) are as follows: tf_{t,e_f} is the number of times term t appears in field f of entity e, $|e_f|$ is the length of field f of e (i.e., $\sum_t tf_{t,e_f}$), μ_f is a smoothing parameter for field f, μ is the entity-level smoothing parameter, $P(t|\theta_{c_f})$ is a field-specific background language model, and $P(t|\theta_c)$ is the general language model for the collection. The smoothing parameter μ_f was set to the average field length of f, i.e., $\sum_e |e_f|/|e|$, where e is the number of entities. Similarly, μ was set to the average entity representation length. The background models $P(t|\theta_{c_f})$ and $P(t|\theta_c)$ were calculated using a standard maximum-likelihood estimate on the corresponding representation. We set the λ_e parameter in Eq. (3.6) to a fixed value of 0.5.

BTC Representation We consider two representations based on the BTC collection. The first approach (`BTC singlefield`) renders triples as single-field documents, and ranks them using a standard LM approach. Specifically, we use Eq. (3.4) for ranking, where the entity model is estimated using Eq. (3.8). The other variation (`BTC name+content`) distinguishes between *name* and *content* fields. Using the retrieval model introduced in the previous subsection, we set $\mathcal{F} = \{\text{name}, \text{content}\}$ and consider the two fields equally important. λ_e (in Eq. (3.6)) was set to 0.7 based on empirical results with last year's queries. Smoothing parameter estimation is as discussed before.

3.4.3 Exploiting "sameAs" Relations

Additionally, we experimented with exploiting "sameAs" relations extracted from DBpedia. We propagate a fraction of the original query-likelihood scores along "sameAs" links:

$$\text{score}(q|e) = P(q|e) + \lambda_S \cdot \sum_{e' \in e_S} |P(q|e') - P(q|e)|, \tag{3.9}$$

where e_S denote the set of "sameAs" variants of entity e. We set λ_S to 0.75.

3.4.4 Preprocessing and Indexing

Both collections we used (DBpedia and BTC) were sorted prior to indexing to facilitate the indexing on a per-entity basis; every subject in the collections was treated as an entity, where the corresponding predicate-object values constitute to the entity's representation. For the indexing part, we used Apache Lucene.[8] Preprocessing was based on Lucene's standard analyzer, including lowercase transformation and basic stop word filtering. We processed all queries with the Yahoo! Spelling Suggestion API[9] and applied the same transformations (lowercasing and stop word removal) as to entity documents. Table 3.2 summarises the indices we built; next, we discuss the collection-specific details.

[8]`http://lucene.apache.org/java/docs`
[9]`http://developer.yahoo.com/search/web/V1/spellingSuggestion.html`

Table 3.2: Indices and sizes used.

Index	#Fields	#Entities	#Size
DBpedia	7	7.8M	20GB
BTC singlefield	1	23.9M	42GB
BTC name+content	2	38M	35GB

DBpedia We used the most recent complete dump of DBpedia made available on the DBpedia homepage.[10] We performed additional preprocessing with respect to URI decoding and matching in order to be compatible with the BTC collection. Additionally, we filtered out those DBpedia URIs from the result list that do not exist in the BTC collection as a subject. To catch all dependencies we used reversed versions of the input files (with subject and object positions switched) for disambiguations, page links, and redirects. We did not perform exhaustive parameter tuning; we tried a few different configurations and used the one that performed best on the SemSearch 2010 queries. The list of fields used and their corresponding weights are shown in Table 3.3.

Table 3.3: Fields in the DBpedia index.

Field Name	Weight
Short abstracts	0.10
Long abstracts	0.10
Article categories	0.07
Disambiguations	0.20
Infobox properties	0.11
Labels	0.30
Wikipedia links	0.12

We also used DBpedia to find name variants for exact name matches: for each DBpedia URI we considered the title and titles of pages redirecting to that URI as the set of name variations.

BTC For the single field index (`BTC singlefield`) we ignored the predicate fields and concatenated all object fields that were literals. We filtered out subjects that had less than 50 characters of textual material associated with them. As to the multi-field index (`BTC name+content`) we manually identified predicates that hold names for the top 10 sources of the BTC collection. All other predicates were considered "content." Again, we limited ourselves to literal objects and filtered out subjects that had less than 50 characters worth of textual data in the content field.

[10]`http://wiki.dbpedia.org/Downloads36`

3.4.5 Submitted Runs

Table 3.4 presents an overview of the runs we submitted. For each, we used the same approach to identifying exact name matches and to ranking entities in DBpedia.

Table 3.4: Runs submitted to the entity search track.

RunID	BTC index	sameAs	MAP
NTNU-1	BTC singlefield	N	0.2072
NTNU-2	BTC name+content	N	0.2063
NTNU-3	BTC name+content	Y	0.2050

As seen in the table, there is little difference in score across our runs. This implies that none of our changes impacted ranking quality significantly, clearly leaving room for future research in entity modelling.

3.5 List Search Task

This section describes our approach to the list search task. We begin with a brief overview of our approach, followed by the description of the data sets we have used in Section 3.5.1. Finally, we present the procedure we used to generate answers for the input queries in Section 3.5.2.

Our approach to the list search task was inspired by the process that a human user would carry on to answer list queries were he asked to do so with the help of Wikipedia. This process would probably be to enter the query to the search field of the Wikipedia GUI and inspect the top k results, matching Wikipedia articles, for the correct answer. One could suspect that the items of the correct answer to the list query would be distinct Wikipedia articles themselves and would be linked from the top results of the full-text query issued against the index of Wikipedia articles. In this spirit, our approach relies on retrieving information from the index of long abstracts of Wikipedia articles; from the top k retrieved articles, we expand by hyperlinks to obtain the list of articles, representing candidate entities for the list query answer. We then check whether the candidate list contains Wikipedia article sets, if so, we boost the scores of members of these sets. Under the term Wikipedia article set, we understand a set of Wikipedia articles forming a semantic group; we describe Wikipedia sets used in this work in Section 3.5.1. If no sets are identified in the candidate list, we merely rely on boosting the score of the items from the candidate list related to the principal entity of the query.

3.5.1 List Search Data Sets

This section provides a list and description of the data sets we have used for the list search task.

Wikipedia article index Lucene index of the long abstracts of Wikipedia articles. More specifically, this index is used to retrieve the articles most related to the input query.

Wikipedia link graph This data set contains the network of Wikipedia articles and links between them. An article is a node in the network and links correspond to the hyperlinks connecting articles. Each node has several attributes—Wikipedia identifier, article title and list of identifiers of the sets the article belongs to. The data set is used to form the candidate list from the top k most related articles to the query.

Wikipedia sets This data set contains sets of Wikipedia articles that form a semantically related group. We used the membership in Wikipedia categories and the inclusion of Wikipedia templates to generate the Wikipedia sets. For example, Wikipedia articles belonging to the category "Category:Astronauts" form one set in Wikipedia set data set, whereas articles using the Wikipedia template "Template:Ancient_Cities_of_Cyprus" form another (derived from the template inclusion in Wikipedia). We have also constructed an index of the Wikipedia sets. For each set a document was created by concatenating the short abstracts of articles belonging to that set. Those documents were indexed and we use the index to check the relevance of the input query to the sets identified as potential answers.

Annotation dictionary This data set contains a mapping between strings and the Wikipedia articles referred to by those strings. We used the default dictionary used by Wikipedia miner,[11] containing the mapping between anchor texts and articles, and extended it by adding article names and names of the redirect pages.

3.5.2 List Search Process

Here, we describe the main components of our approach to the list search task. It consists of the following steps (each of which will be described in more detail below): query analysis, querying the article abstracts index, generating the list of candidate items, boosting of scores of entities related to the main entity of the query, and, finally, boosting of scores of items belonging to Wikipedia sets.

1. Query Analysis. Our first step is to analyse the query, which is facilitated by using the Wikipedia miner toolkit to annotate the query with Wikipedia topics. This gives us: a) query segmentation that we exploit in the full-text search step, b) entities (in form of Wikipedia article titles) that the query targets. We identify the principal entity of the query (the one with the highest relevance score from Wikipedia miner) and use it for query reformulation. If the query segment related to the principal entity is only slightly different from the entity article's title (the Levenshtein distance equals to 1), we try to

[11] `http://wikipedia-miner.sourceforge.net`

update the query by replacing the given query segment with the title of the entity article. We run both the original and the reformulated query against the index of Wikipedia abstracts. If the sum of the scores of top 10 items of the reformulated query is significantly higher (at least 2 times) than the sum of scores of the top 10 items from the original query, we use the reformulated query in the following steps. In the SemSearch Challenge data set, the query reformulation was used instead of the original for two queries.

2. **Querying the article abstracts index.** We run two queries against the article abstracts index. The first one is evaluated by the Language Modelling approach described in 3.4.2. The second query is a boolean query with the following constraint: the terms from the text query segment related to the principal entity (identified in step 1) must be present in the target document. As the two result sets use different scoring functions, we merge the results by normalisation; from each result set, we take the first 100 items, and normalise their scores (so that their sum is equal to 1), we then merge the results. Let T_k be the set of top k results of the merged ranks, and members of the set are Wikipedia articles. Let $r(i); i \in T_k$ be the rank of the item i in the merged result ranking.

3. **Generating the list of candidate items.** In this step, we take the top k (in the submitted runs $k = 10$) items from the previous step (T_k). We generate the list of candidate entities by taking those top k results and expanding from related articles by Wikipedia links. Every item added to the candidate list is assigned a score proportional to the rank of the item from T_k. If the item already exists in the candidate list the score is added to its existing score. In this way, the items referred by multiple links from the result set from the previous step will receive a higher score. More formally, let $G = V, E$ be the link graph of Wikipedia. Let $e(i, j) = 1 \Leftrightarrow e_{i,j} \in E$ and $e(i, j) = 0 \Leftrightarrow e_{i,j} \notin E$. We hence denote a candidate set C as:

$$C = \{j; j \in V \land \exists i \in T_k : e(i, j) = 1\}. \tag{3.10}$$

We set the score of item $c \in C$ to be:

$$w(c) = \sum_{i \ inT_k} e(i, c) \times (1 - ((r(i) - 1) \times (1/k))). \tag{3.11}$$

4. **Boosting of items related to principal entity.** We take the principal entity P identified in the query analysis (Step 1) and boost the scores of the entities in the candidate list that both are: a) linked to by the principal entity and b) link to the principal entity (in the link graph of Wikipedia). We boost the score of such an item I by computing the cosine similarity of the principal entity and the given item. In this case we represent Wikipedia articles as vectors constructed from their adjacency lists in the link graph. If $w(I)$ is the original score of the item, the boosted score $wb_1(I)$ is defined as

follows: $wb_1(I) = w(I) \times sim(P, I) \times b_1$, where b_1 is the boost constant (in our experiments we used $b_1 = 100$) and $sim(P, I)$ is the cosine similarity of the adjacency vectors.

5. **Boosting of scores of Wikipedia set items.** In this step, we identify Wikipedia sets that have more than p fraction of their members in the candidate list. For each such set S we compute the similarity score of the query and the set document $D(S)$ (see Section 3.5.1), using the standard Lucene scoring function. We boost scores for the items from the candidate list according to:

$$wb_2(I) = w(I) \times b_2 \times \sum_{I \in S : |S \cap C| \geq p \times |S|} sim(q, D(S)). \tag{3.12}$$

   Here, b_2 is the boost constant for sets, and $sim(q, D(S))$ is the similarity of the set document $D(S)$ and query q; in our runs we have used $p = 0.7$.

6. **Postprocessing** In the postprocessing step, we sort the items in the candidate list in descending order according to their scores and we map results back to subjects in the BTC collection.

3.5.3 Submitted Runs

We submitted three runs for the list search task. The first one followed the process as described in Section 3.5.2, only omitting Step 5.—boosting of the Wikipedia sets items. The second run exactly followed the approach presented in Section 3.5.2. The third run differed in the postprocessing step. For the items with a score higher than a defined threshold, have also added entities linked by “sameAs” relations.

3.5.4 Discussion

For the list search task, we limited our approach to the Wikipedia data set. The reason for this was purely pragmatic—while a part of the team was processing the BTC data set to a usable form, the rest of the team was experimenting with the list search task on the Wikipedia data set (which, in the form of DBpedia, is also covered in the BTC collection). Due to time constraints and the quality of pre-submission results we achieved, we decided to keep using only the Wikipedia data set. However, our approach is fully transferable to the BTC collection or any other to RDF data set in general, as we exploit textual descriptions of entities, links between them, and type information (in the form of categories/templates). One important observation from the results is that by exploiting Wikipedia sets and boosting sets’ members we can achieve substantial improvements compared to our baseline. Considering “sameAs” variants of high scoring entities led to further performance improvements; see Table 3.5.

Table 3.5: Runs submitted to the list search track.

RunID	Wikipedia set boosts	sameAs	MAP
NTNU-1	N	N	0.1625
NTNU-2	Y	N	0.2594
NTNU-3	Y	Y	0.2790

3.6 Results of the Challenge

A total of four teams participated in the entity search track. An overview of the results is given in Table 3.6. Sindice's submission achieved the highest scores in terms of MAP, second to the university of Delaware (Udel) [56]. Sindice makes use of multi-valued field normalisation in an extension to BM25F [33]. This is achieved by introducing additional parameters to the retrieval model. All three of NTNU's setups were competitive and got the respective third, fourth, and fifth rank.

Table 3.6: Competition results for the entity search track.

Rank	Participant	Run	MAP
1	9-Sindice	2	0.2346
2	13-UDel	2	0.2167
3	3-NTNU	1	0.2072
4	3-NTNU	2	0.2063
5	3-NTNU	3	0.2050
6	13-UDel	1	0.1858
7	9-Sindice	1	0.1835
8	9-Sindice	3	0.1635
9	5-IIIT Hyd	1	0.0876
10	5-IIIT Hyd	2	0.0870

For the list search track, a total of five teams submitted (all teams participating in the entity search track plus an additonal team, the Ambani Institute of Information and Communication Technology (DAA-IICT). NTNU's approach clearly outperformed the other submissions as shown in Table 3.7. Sindice and Delaware came second and third, however, with a substantially lower MAP score. We attribute the clear win to our strategy of dividing our resources between both tasks early in the process and targeting strong submissions in both categories. The other teams used only slight modifications with respect to query processing over their submissions to the entity search track. Also all other teams used the BTC data whereas we used mainly Wikipedia.

Table 3.7: Competition results for the list search track.

Rank	Participant	Run	MAP
1	3-NTNU	3	0.2790
2	3-NTNU	2	0.2594
3	3-NTNU	1	0.1625
4	9-Sindice	1	0.1591
5	9-Sindice	3	0.1526
6	9-Sindice	2	0.1505
7	13-UDel	1	0.1079
8	13-UDel	2	0.0999
9	5-IIIT Hyd	1	0.0328
10	5-IIIT Hyd	2	0.0328
11	15-Daiict	1	0.0050

3.7 Conclusions

In our participation we focused on integrating evidence from multiple sources for the entity search task: we employed a fielded Language Modelling approach to rank entities in the BTC collection and in DBpedia. Additionally, we considered strict name matches based on a dictionary of entity name variants extracted from DBpedia. As to the list search task we attempted to model human user behaviour when searching in Wikipedia. Our approach includes a query analysis step to identify the principal entity in the query. In the ranking phase we utilise the Wikipedia link graph and semantically related article sets, defined by Wikipedia categories and templates.

With our best entity search run ranked third among all submissions and our list search runs were placed first, second, and third, of all runs, we consider our approaches competitive and intend to further improve on them.

Possible directions for future research include an exhaustive success and failure analysis based on the full system evaluations. As our approaches employ a number of parameters, most of which were set intuitively in the lack of time and—in case of the list search task—in the lack of training data, we believe that there is much to gain by adjusting these settings. Inspired by the relatively small difference between our runs, we see further potential in improving the entity modelling component. We omitted the list search task from future work, mostly due to the relatively good results we achieved on it. Besides, all list search approaches will benefit from improvements on the entity search task to a certain degree.

Chapter 4

Semantic Entity Search

In this chapter we continue to investigate fielded or structured retrieval models for entity search based on our observations in Chapter 3. Starting from the simple fielded model introduced in the previous chapter and the experimental results we obtained so far, we propose a generalised framework for fielded entity retrieval in the WoD. We first introduce a strong baseline based on a straight-forward model, building on the findings in [81]. Further, we introduce extensions based on both attributes and types of the entities in the data set.

4.1 Introduction

In this chapter, we follow up on the general entity search task described in Section 2.6. The overall research question we address is how to represent entities in the Web of Data for the purpose of text-based retrieval. We put our work in a broader context and survey related work in Section 4.2.

We then start from a standard document retrieval approach and consider simple extensions: (1) extended preprocessing, a heuristic for extracting textual content from URI descriptors, (2) a two-field representation, distinguishing between title and content, and (3) entity importance, assigning more weight to entities from trusted, high-quality sources. This is presented in Section 4.3.

We show that these extensions lead to improvements and that they add up. In fact, our approach outperforms all previously reported results on both query sets, despite that those were generated by far more complex systems. We observe that the extended URI preprocessing accounts for the majority of the improvements.

Based on the observation that preserving structure can improve retrieval, we propose models taking this structure into consideration to a stronger degree in Section 4.4. First, in Section 4.4.1 we formalise two entity modelling strategies within a generative language modelling framework. One approach (Unstructured Entity

Model) collapses all text associated with the entity into a single flat-text representation. The other approach (Structured Entity Model) groups predicates together into a small number of categories and considers their weighted combination. We perform an experimental evaluation of the two models using the 2010 and 2011 test sets of the Semantic Search Challenge evaluation campaign in Section 4.4.2. We find that these models outperform the current state-of-the-art in terms of retrieval effectiveness on these collections. However, this is done at the cost of abandoning a large part of the semantics behind the data. Subsequently, in Section 4.4.3, we propose a novel entity model (Hierarchical Entity Model) capable of preserving the semantics associated with entities. It uses the idea of having a two-level hierarchy for entity representation, one based on the predicate types, another based on the individual predicates. We report on experiments using our hierarchical model in Section 4.4.4 and show that modelling individual predicates of a given type is more effective than folding their contents into a flat representation. Finally, we conclude with a discussion of our findings and outline our plans for future research in Section 4.5.

4.2 Related Work

It has been shown that that over 40% of web search queries target entities [92]. Following up on this trend, a range of commercial providers now support entity-oriented search, dealing with a broad range of entity types such as people, companies, services, or locations. This shows that the problem studied in this chapter, searching entities in structured semantic data, "has direct relevance to the operation of web search engines, which increasingly incorporate structured data in their search results pages" [21]. A range of evaluation benchmarks have been organised, for details of this we refer back to the overview given in Section 2.6.

The ad-hoc entity retrieval task over RDF data we studied in this chapter was proposed and formalised in [92]. This task bears some resemblance to keyword search in relational databases [89] and to XML retrieval [50]. The former has no direct relevance as methods developed for structured databases are not directly applicable to RDF data. As for the latter, Kim et al. [54] presented a probabilistic retrieval model for semi-structured data (PRM-S) that allows for a weighted mapping of query terms to (entity) attributes. Dalton and Huston [30] tested this model on the SemSearch 2010 data set and found that a key limitation of the PRM-S approach is that "it assumes a collection with a single or very few clearly defined entity types". Unlike the IMDB and Monster collections used in [54], the BTC-2009 corpus is very heterogeneous. Additionally, two specific problems were identified: (1) computing the query term to attribute mapping probabilities suffers from attribute sparsity, and (2) mapping probabilities are estimated for each query term independently. Finally, Ogilvie and Callan [87] considered modelling documents with a hierarchical structure for XML retrieval. Our Hierarchical Entity Model has been developed in a similar spirit, but hierarchy in [87] is defined by document structure (markup), while in our case it is organised around semantic relationships. Both works estimate language models on the component levels and incorporate evidence from multiple levels within the hierarchy, but the task addressed in [87] (XML component retrieval), and hence the actual models, are different from ours.

4.3 Fielded Models for Entity Search

We start with a baseline retrieval system that constructs pseudo documents from RDF triples and introduce three extensions: preprocessing of URIs, using two-fielded retrieval models, and boosting popular domains. Using the query sets of the 2010 and 2011 Semantic Search Challenge, we show that our straightforward approach outperforms all previously reported results, some of which were generated by far more complex systems.

We address the ad-hoc entity search task in RDF data: given a keyword query, targeting a particular entity, return a ranked list of relevant entities identified by their URIs. Each entity is described by a set of subject-predicate-object RDF triples. For each entity, we build a textual representation by considering all triples where it stands as the subject; we use only the object's (string) value from the triple and refer to it as *object value*.

Baseline Retrieval. All object values are concatenated together into a flat text representation. We perform standard tokenisation and stopword removal; no stemming is applied. We use standard retrieval models: BM25 from Eq. 2.20 and Language Models (LM) from Eq. 2.24.

Fielded Representation. We use a simple heuristic to identify predicates that hold title values: these end with "name", "label", or "title". Object values belonging to title-type predicates are concatenated into an additional *title* field. This is the only part where we have some (limited) semantics captured in our approach. Given this title+content representation, we use fielded versions of BM25 and LM, specifically, BM25F [95], as shown in Eq. (2.28), and the Mixture of Language Models [86], referred to as LMF, as given in Eq. (2.29) and Eq. (2.30).

Entity Importance. Entities from trusted, high-quality sources are considered more important and receive an extra query-independent weight in their retrieval score. In case of BM25, this is incorporated as a multiplication factor; for LM, we use the document (entity) priors for this is purpose. In our experiments, we illustrate the effects of this component by boosting DBpedia, which is a central hub in the Linked Data cloud.

Extended Preprocessing. For all settings we introduced before, we apply a heuristic to extract (additional) textual content from URIs. We do so by using the string part of the URI after the last slash as the object value. Additionally, we make sure that characters like underscores, dashes, brackets, etc. are all treated as whitespaces.

All data is stored in a Lucene index.[1] We further try to improve the baseline results obtained when using standard Lucene ranking by using advanced retrieval models. More specifically, we apply BM25 and a language modelling approach to the single-fielded and multi-fielded data.

4.3.1 Experiments

In this section we first introduce our experimental setup, then perform an evaluation of our entity models.

Experimental Setup

We use the test suites of the 2010 and 2011 editions of the Semantic Search (SemSearch) Challenge [20, 46]. For a more thorough introduction of the data set, we refer to Section 3.3.

Evaluation metrics. We use standard IR evaluation metrics: Mean Average Precision (MAP), Precision at rank 10 (P@10), and Normalised Discounted Cumulative Gain (NDCG).[2] To check for significant differences between runs, we use a two-tailed paired t-test and write $^{\dagger}/^{\ddagger}$ to denote significance at the 0.05/0.01 levels, respectively.

Resource resolution. If a relation (predicate targeted at a URI rather than a literal value) points to an entity within the collection, we replace the URI with that entity's name. Otherwise, we extract terms from the relative part of the URI.

We report on Mean Average Precision (MAP), the main metric used at SemSearch, a more in-depth explanation can be found in Section 2.2.2. Significance testing is performed using a two-tailed paired t-test.

Table 4.1 reports on a series of experiments we performed using two different retrieval models (LM and BM25) and two different parameter settings. For the default setting, shown in columns 4 and 5, no training material is used; we take values suggested in the literature or values that intuitively seem reasonable. For LM, we use the average document/field length (*avgdl*) as the smoothing parameter μ [86]. For BM25, we use $k1$=1.2 and b=0.25; we use the same b value for all fields in the fielded variant BM25F, analogous to [86] and [26]. We use a weighting of 0.2/0.8 for the title/content fields. The optimised parameter setting, displayed in column 6, is only for the 2011 query set. We use relevance assessments from the previous year as training material; these were also available to SemSearch 2011

[1]`http://lucene.apache.org`

[2]Note that the two editions used different gain values for computing NDCG scores: 3 and 2 (2010) vs. 2 and 1 (2011) for excellent and fair, respectively. We use these values unchanged.

Table 4.1: Retrieval results. (Rows 1-12): results obtained in our work; (Rows 13-14): best results taken from the literature. Best scores for each column are in boldface.

Run	URI Preproc.	Retrieval Model	2010 MAP	2011 MAP	2011 (opt) MAP
Baseline (content)	-	LM	0.1832	0.1840	0.1840
	-	BM25	0.1888	0.1970	0.2154
	+	LM	$0.2388^‡$	$0.2445^‡$	$0.2445^‡$
	+	BM25	$0.2464^‡$	$0.2502^‡$	$0.2702^‡$
title+content	-	LMF	0.1832	0.1840	0.1840
	-	BM25F	0.1888	0.1970	0.2154
	+	LMF	$0.2900^‡$	$0.2618^‡$	$0.2765^‡$
	+	BM25F	$0.2621^‡$	$0.2625^‡$	$0.2937^‡$
title+content	-	LMF	$0.1836^‡$	$0.1846^‡$	$0.1846^‡$
+ *dbpedia.org* boosting	-	BM25F	$0.1909^‡$	$0.2031^‡$	$0.2166^‡$
	+	LMF	$\mathbf{0.2914}^‡$	$0.2651^‡$	$0.2756^‡$
	+	BM25F	$0.2631^‡$	$\mathbf{0.2642}^‡$	$\mathbf{0.2991}^‡$
Best at SemSearch [20, 46]			0.1919		0.2346
Best reported since [26]			0.2805		

participants. The best found parameter settings are: $\mu{=}avgdl$ for LM, $\mu{=}2 \cdot avgdl$ for LMF, and $k1{=}0.4$ and $b{=}0.4$ for BM25/BM25F.

First, in rows 1-4, we use standard retrieval models with flat text representation. we see large differences depending on the URI preprocessing; all results using the advanced preprocessing in rows 3-4 for URIs are significantly different from the baselines without preprocessing in rows 1-2. Next, in rows 5-8, we use fielded variants of these models, with two fields: title and content. The results in rows 5-6 equal rows 1-2; this is because the title field cannot contribute to the entity representation without URI preprocessing. The results in rows 7 and 8, however, outperform their counterparts in rows 3 and 4; assigning higher weight to the title field clearly benefits retrieval when title values are extracted correctly. Finally, in rows 9-12, we boost entities coming from high-quality trusted sources, in our case DBpedia. In columns 4 and 5 we use the boosting value of 1.5, indicating that all scores of DBpedia entities are multiplied by that value. In column 6, we show results for the boosting factor that showed the best results on the 2010 queries (a value of 2.2). However, this leads to a performance decrease in row 11 compared to row 7; we attribute this to the fact that there are more relevant answers from DBpedia for 2010 than for 2011.

We chose to use both BM25 and LM to investigate if both retrieval models display the same behaviour with respect to the techniques we applied. We find that this is indeed the case, but we also discovered two interesting differences. First, with default parameter settings, LM performs better on the 2010 queries while BM25 does slightly better on 2011. Second, BM25 benefits more from parameter optimisation.

We achieve the highest MAP value for the 2010 queries using the fielded language models and using BM25F for the 2011 queries (row 11 and 12). We explain this by the nature of the queries and the fact that BM25 is able to better fit these differences using parameter optimisation. LMF, on the other hand, performs reasonably well with the standard settings in many cases.

We want to point out a correlation between query sets and retrieval models used. We consistently obtained better results for the 2010 queries with the language modelling approach, whereas BM25 performed best on the 2011 queries. The 2010 query set comprises 92 queries with an average query length of 2.6 terms and an average number of unique terms per query of 2.25). In the 2011 query set 50 queries were used with an average query length of 2.7 terms and 2.62 unique terms per query. In other words the 2010 query set overall contains more duplicate terms, a fact which is best handled by the language modelling approach. Furthermore the fraction of relevant documents in the relevance judgements stemming from *dbpedia.org* is more than double in the 2010 query set.

In comparison to other approaches, we outperform all published results for both years' queries as shown in the last two rows of Table 4.1. Campinas et al. [26] report improved results for the 2010 query set and achieve an MAP of 0.2805; a large fraction of their improvements can be attributed to additional query, attribute, and entity weighting. Blanco et al. [21] report a MAP score of 0.2705 on the 2010 queries using a manual grouping and weighting of predicates. Both works use BM25F.

4.4 Advanced and Adaptive Entity Modelling in the Web of Data

In previous section, we showed that simple fielded models outperform the current state-of-the-art in terms of retrieval effectiveness, however, this is done at the cost of abandoning a large part of the semantics behind the data. In this section, we propose a novel entity model capable of preserving the semantics associated with entities, without sacrificing retrieval effectiveness.

4.4.1 Baseline Entity Models

In this section we formalise and draw upon two existing entity modelling approaches within a generative language modelling framework.

Retrieval Framework

We study the entity retrieval problem in a generative language modelling setting. Language models (LMs) are attractive because of their solid theoretical foundations that couples with good empirical performance [124]. LMs have been successfully applied to a wide range of entity-related search tasks; see, e.g., [9, 30, 39, 54]. This is in line with our observations in Section 4.3. A general overview is given in Sections 2.3.1 and 2.3.2 for the fielded variants. The following notation deviates from the background section in that it regards entities instead of documents.

We rank candidate entities (e) according to their probability of being relevant given query q: $P(e|q)$. Instead of estimating this probability directly, we apply Bayes' rule and drop the denominator as it does not influence the ranking (for a given query): $P(e|q) = \frac{P(q|e)P(e)}{P(q)} \overset{\text{rank}}{=} P(q|e)P(e)$. Here, $P(e)$ is the prior probability of choosing a particular entity e, that we subsequently attempt to draw the query q from, with probability $P(q|e)$. Here, we assume that $P(e)$ is uniform, thus, does not affect the ranking. Entity priors could be used to incorporate query-independent features into the ranking, for example, based on the RDF graph structure [26, 33].

Each entity e is represented by a multinomial probability distribution over the vocabulary of terms. The entity model θ_e is used to predict how likely the entity would produce a given term t, that is, $P(t|\theta_e)$. Assuming that query terms are sampled identically and independently, the query likelihood is obtained by taking the product across all the terms in the query, such that:

$$P(q|\theta_e) = \prod_{t \in q} P(t|\theta_e)^{tf_{t,q}}, \tag{4.1}$$

where $tf_{t,q}$ is the (raw) frequency of term t in the query. Note that $P(t|\theta_e) > 0$ must be ensured for all vocabulary terms, otherwise the product might end up being zero.

The main question we will be concerned with for the remainder of this chapter is the estimation of the entity model, i.e., the probability distribution, $P(t|\theta_e)$.

Unstructured Entity Model

The simplest approach to constructing entity models is to fold all text associated with the entity into one “bag-of-words”; see Table 4.2(a) for an illustration. Following the standard language modelling approach to document retrieval, we implement the entity model as a Dirichlet-smoothed multinomial distribution:

$$P(t|\theta_e) = \frac{tf_{t,e} + \mu P(t|\theta_c)}{|e| + \mu}, \tag{4.2}$$

where $tf_{t,e}$ is the raw frequency of term t in the representation of e and $|e|$ is the size of this representation, i.e., $\sum_t tf_{t,e}$. The smoothing parameter μ is set to the

Table 4.2: Examples of baseline entity models corresponding to Figure 1.3. URIs are resolved (i.e., replaced with the name of the corresponding entity).

(a) Unstructured Entity Model.

Text
Audi A4
1996 2002 2005 2007
The Audi A4 is a
compact executive car...
Volkswagen Group
Compact executive cars
All wheel drive vehicles
Front wheel drive vehicles
Product
Audi A4
Audi 80
Audi A5
Audi A4

(b) Structured Entity Model.

Pred. type	**Value**
Name	Audi A4
Attributes	1996 2002 2005 2007 The Audi A4 is a compact executive car...
OutRelations	Volkswagen Group Compact executive cars All wheel drive vehicles Front wheel drive vehicles Product Audi A4
InRelations	Audi 80 Audi A5 Audi A4

average entity representation size in the collection. The term generation process is shown in Figure 4.1 (Left).

Several SemSearch participants employed variations of this approach: considering only triples where the entity stands as a subjects (thereby ignoring incoming relations) [13, 30, 56] or extracting text only from the subject itself [36].

Structured Entity Model using Predicate-folding

Instead of folding all predicates together, they might be grouped into multiple categories in order to preserve some of the original structure. Prior work presents examples of such grouping based on the type of the predicates [13, 30, 47, 90] or based on their manually determined importance [21]. We group RDF triples into four main predicate types p_t for a given entity e as follows:

- *Name*: the subject is e, the object is a literal, and the predicate comes from a predefined list (e.g., *foaf:name* or *rdfs:label*) or ends with “name”, “label”, or “title”.
- *Attributes*: the subject is e, the object is a literal, and the predicate is not of type name.
- *OutRelations*: the subject is e and the object is a URI. We resolve the URI by replacing it with the name of the corresponding entity; see Section 4.3.1 for details.

- *InRelations*: e stands as the object; the subject entity URI is resolved.

Figure 4.2(b) presents an example. A separate language model $\theta_e^{p_t}$ is estimated for each predicate type from all predicates of that type (associated with the given entity):

$$P(t|\theta_e^{p_t}) = \frac{tf_{t,p_t,e} + \mu_{p_t} P(t|\theta_c^{p_t})}{|p_t, e| + \mu_{p_t}}, \tag{4.3}$$

where $tf_{t,p_t,e}$ is the sum of the term frequencies for t for all predicates of type p_t associated with e, and $|p_t, e| = \sum_t tf_{t,p_t,e}$. Essentially, we concatenate all text from the predicates of type p_t and then apply Dirichlet smoothing using a collection-wide background model $P(t|\theta_c^{p_t})$ (that is, a maximum-likelihood estimate from all predicates of that type in the collection). The smoothing parameter μ_{p_t} is set to the average predicate type representation length in the collection.

Subsequently, the entity model is a linear mixture of the predicate type language models ($P(t|\theta_e^{p_t})$). This is additionally weighted with the importance of that predicate type ($P(p_t)$):

$$P(t|\theta_e) = \sum_{p_t} P(t|\theta_e^{p_t}) P(p_t). \tag{4.4}$$

Figure 4.1 (Middle) illustrates the term generation process. Viewing entities as documents and predicate types as document fields, this model is equivalent with the Mixture of Language Models approach by Ogilvie and Callan [86].

4.4.2 Evaluation of Baseline Entity Models

In this section, we provide an evaluation of both our baseline entity models in turn.

Unstructured Entity Model

Recall that this model creates an unstructured entity representation by collapsing text associated with the entity into one bag-of-words. In order to show the impact of the different predicate types on retrieval effectiveness we perform three sets of experiments, and report the results in Table 4.3. First, identified by the 'ALL' row, we consider all four predicate types (*Name*, *Attributes*, *OutRelations*, and *InRelations*). Next, shown in the 'field type-only' rows, we use only a single predicate type at a time. Finally, in the rows named 'ALL-but-field type,' we present results by omitting one of the types in turn.

Our findings are as follows. The absolute performance of the ALL model (that simply uses all text associated with the entity) is remarkable; on the 2010 topic set it outperforms the best SemSearch 2010 submission (MAP=0.1919) [46], while on the 2011 set it would have ranked third (best was MAP=0.2346) [20]. As for the individual predicate types, *Name* and *Attributes* perform best, with only minor

Table 4.3: Retrieval results using the Unstructured Entity Model. Significance is tested against the ALL setting (first line). Best scores for each topic set are typeset boldface.

	2010			**2011**		
Predicate types	**MAP**	**P@10**	**NDCG**	**MAP**	**P@10**	**NDCG**
ALL	0.207	0.314	0.383	**0.207**	0.188	**0.295**
Name-only	0.205†	0.304	0.397	0.200	0.190	0.310
Attributes-only	0.206‡	0.339†	0.392	0.220†	0.190	0.333†
OutRelations-only	0.087‡	0.176‡	0.216‡	0.094‡	0.102‡	0.167‡
InRelations-only	0.096‡	0.197‡	0.226‡	0.069‡	0.098‡	0.154‡
ALL-but-Name	0.157‡	0.262‡	0.321‡	0.156‡	0.144‡	0.247‡
ALL-but-Attributes	0.182‡	0.286‡	0.342‡	0.164‡	0.160‡	0.242‡
ALL-but-OutRel.	0.203‡	0.311	0.369‡	0.200‡	0.180	0.283
ALL-but-InRel.	**0.213**	**0.320**	**0.385**	0.204‡	**0.190**	0.283

differences between the two; in fact, either of these predicates types alone is on a par with the ALL model. The scores for incoming and outgoing relations are much lower. Looking at the combinations of all but one predicate types, *Name* contributes the most and *Attributes* comes second; without either, the scores drop substantially. Removing either incoming or outgoing relations has hardly any overall impact; in the 'ALL-but-InRelations,' 2010 case the scores marginally improve, but the difference is not statistically significant. Note that—with the aforementioned exception—all results in the bottom block of Table 4.3 are significantly different from the ALL combination in terms of MAP.

Structured Entity Model

Our second baseline model employs a weighted combination of four language models, corresponding to predicate types. We consider three different configurations in Table 4.4: (1) equal weights on all types, (2) all weights allocated to *Name* and *Attributes*, in equal proportion, and (3) most of the weights assigned to *Name* and *Attributes*, again, but this time taking relations too into account. In the lack of training data or any background knowledge about the collection or queries, (1) is a natural choice. (2) and (3) are motivated by the results from the previous subsection; however, knowing from the task definition that each query targets a particular entity, one could intuitively argue for such weight distributions. Note that when a single predicate type is used, the Structured Entity Model is equivalent to the unstructured case (Table 4.3, middle block).

All settings improve significantly and substantially over the unstructured baseline. We find, somewhat surprisingly, that the uniform weighting performs best of all; these numbers are the highest that were ever reported for either topic sets (which are: MAP= 0.2705 for 2010 [21] and MAP=0.2346 for 2011 [20]). Even though

Table 4.4: Retrieval results using Structured Entity Models. Significance is tested against the ALL setting of the Unstructured Entity Model (Table 4.3, first line). Best scores are typeset boldface. OutR stands for outgoing, InR for ingoing relations.

Pred. type weights				**2010**			**2011**		
Name	**Att**	**OutR.**	**InR.**	**MAP**	**P@10**	**NDCG**	**MAP**	**P@10**	**NDCG**
0.25	0.25	0.25	0.25	**0.282**‡	0.399‡	**0.494**‡	0.260‡	**0.242**‡	0.397‡
0.50	0.50			0.249‡	0.366‡	0.461‡	0.251‡	0.230‡	0.385‡
0.35	0.35	0.15	0.15	0.280‡	**0.404**‡	0.493‡	**0.261**‡	0.230‡	**0.397**‡

Name and *Attributes* were shown to be the two most ‘useful’ predicate types, using them without relational information is clearly suboptimal (row 2 vs. rows 1 and 3). The third setting (row 3) suggests that as long as all predicate types are considered, the model is robust with respect to the actual weight distribution used. We experimented with other configurations too, but none of them improved significantly over the uniform weighting.

4.4.3 Hierarchical Entity Model

In this section we introduce a novel entity modelling approach. Before presenting our proposal, we briefly review the considerations leading to the choice of this particular model. (1) In a heterogeneous environment the number of distinct predicates is huge. It is not feasible to optimise their weights directly (because of the computational complexity and because of the enormous amounts of training material it would require). (2) Entities are sparse with respect to the different predicates, i.e., most entities have only a handful of distinct predicates associated with them. (3) As we have shown with the Structured Entity Model, folding predicates based on their type is a viable alternative that works well in practice. Nevertheless, we need a different solution if we wish to preserve the semantics in the entity model, i.e., keep individual predicates and their contents accessible (and possibly exploit this information in the retrieval model). Users can only be offered to make use of single predicates if they are preserved individually, such kind of faceted search may not improve effectiveness in the context of the Semantic Search Challenge, but might be a requirement given from the user side.

The main idea behind our model is to organise information belonging to a given entity into a hierarchy of two levels, with predicate types (i.e., name, attributes, incoming and outgoing relations) on the first level and individual predicates of that type on the second level. This preserves the original structure associated with the given entity, and allows for setting the importance individual predicates conditioned on their type and on the entity. Formally, this is expressed as follows:

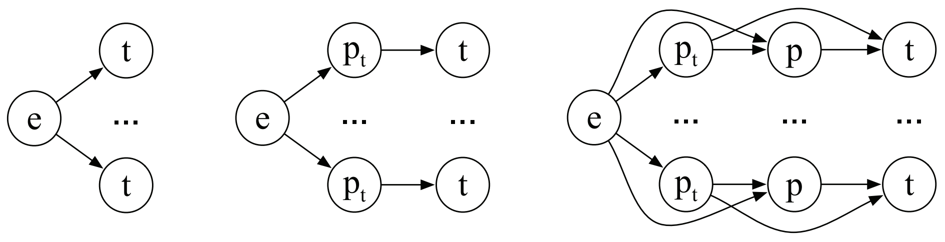


Figure 4.1: Graphical representations of entity models: (Left) Unstructured Entity Model, (Middle) Structured Entity Model using Predicate-folding, (Right) Hierarchical Entity Model.

$$P(t|\theta_e) = \sum_{p_t} P(t|p_t, e)P(p_t|e) \tag{4.5}$$

$$= \sum_{p_t} \big(\sum_{p \in p_t} P(t|p, p_t)P(p|p_t, e)\big) P(p_t|e). \tag{4.6}$$

The term generation process under this model is shown in Figure 4.1 (Right). Next, we discuss the three components from Eq. (4.6): term generation ($P(t|p, p_t)$), predicate generation ($P(p|p_t, e)$), and predicate type generation ($P(p_t|e)$).

Term generation. The importance of a term is jointly determined by the predicate in which it occurs as well as all other predicates of that type associated with the entity:

$$P(t|p, p_t) = (1 - \lambda)P(t|p) + \lambda P(t|\theta_e^{p_t}), \tag{4.7}$$

where $P(t|p)$ is a maximum-likelihood estimate (i.e., the relative frequency of term t in predicate p) and $P(t|\theta_e^{p_t})$ is the Dirichlet-smoothed LM for predicate type p_t, estimated using Eq. (4.3). The parameter $\lambda \in [0..1]$ controls the influence of the predicate type model. For the sake of simplicity, we set λ to 0.5 in our experiments.

Predicate generation. The importance of a given predicate p is conditioned on the type of the predicate p_t and the entity e: $P(p|p_t, e)$. We consider four natural options:[3]

- *Uniform* All the predicates of the same type are treated as being equally important: $P(p|p_t, e) = 1/n(e, p_t)$, where $n(e, p_t)$ is the number of predicates of type p_t assigned to e.[4]

[3]It is by design that predicate importance is independent of the query; this way other, possibly computationally heavy, alternatives for setting this probability could be estimated offline.

[4]Since predicates encoding the entity name are all equally important, we do not vary their importance, hence we always use the uniform distribution for $p_t = name$.

- *Length* The probability mass is allocated to predicates proportional to their length: $P(p|p_t, e) = |p, e|/|p_t, e|$, where $|p, e|$ and $|p_t, e|$ are the lengths of p and p_t, respectively, measured in the number of terms they contain.
- *Average length* We use the average length of the predicate p in the collection relative to the average length of all predicates of type p_t.
- *Popularity* We regard popular predicates more important, where popularity is measured in terms of the number of triples that have the given predicate: $P(p|p_t, e) = n(p)/n(p_t)$, where $n(p)$ and $n(p_t)$ are the total number of triples in the collection with predicate p and predicate type p_t, respectively. This is independent of the entity e

Predicate type generation. The model in Eq. (4.6) allows us to set predicate type importance on a per-entity basis. To simplify matters, however, we make the conditional independence assumption between predicate types and entities, hence $P(p_t|e) = P(p_t)$. This allows us to use estimation methods (or the actual values) from the Structured Entity Model, as this component is identical in the two models (cf. Eq. (4.4)).

4.4.4 Evaluation of the Hierarchical Entity Model

In order to evaluate the hierarchical entity model, we first show the results obtained for each of the predicate types in turn, analogous to the second block in Table 4.3. This is also the baseline we test against. Table 4.5 presents the results for each of the predicate types and different weighting models for individual predicates. As indicated, improvements are significant in the majority of cases. We show improvements over the results reported for the unstructured model in every category, illustrating that the additional level of normalisation can cover the structure of the entities. The "uniform" weighting strategy performs best for all types with "popularity" coming second. Therefore we omit the results for the remaining predicate weighting strategies in Table 4.6, where we show the effect of using multiple field types and combine them. We can not improve results here, which we attribute to both the unexpectedly strong baseline and a certain inability of the model to exploit the semantic information. This is somewhat surprising since we showed we can cover for individual predicate types and as such can contribute to better predicate modelling (as shown in Table 4.5).

4.5 Conclusions

We have addressed the task of ad-hoc entity retrieval in the Web of Data: returning a ranked list of RDF resources that represent an entity described in the keyword query.

Table 4.5: Retrieval results using Hierarchical Entity Models using a single predicate type.

Predicate	**Predicate**		**2010**			**2011**	
type	**weighting**	**MAP**	**P@10**	**NDCG**	**MAP**	**P@10**	**NDCG**
Attributes	Uniform	0.235‡	0.326	0.440‡	0.242‡	0.214	0.347
	Length	0.212‡	0.287‡	0.415	0.229‡	0.206	0.348
	AvgLength	0.213‡	0.286‡	0.414	0.229‡	0.206	0.351
	Popularity	0.232‡	0.310†	0.439‡	0.251‡	0.214	0.379
OutRelations	Uniform	0.119‡	0.178	0.261	0.122‡	0.122†	0.205‡
	Length	0.092‡	0.166	0.212	0.098‡	0.092	0.165
	AvgLength	0.091‡	0.153†	0.213	0.096‡	0.094	0.168
	Popularity	0.107‡	0.167	0.244	0.123‡	0.126†	0.207‡
InRelations	Uniform	0.107‡	0.205†	0.239	0.080‡	0.116†	0.175‡
	Length	0.094‡	0.179†	0.219	0.070	0.094	0.159
	AvgLength	0.092‡	0.170‡	0.221	0.078‡	0.098	0.175‡
	Popularity	0.112‡	0.202	0.245†	0.089‡	0.124†	0.190‡

Table 4.6: Retrieval results using Hierarchical Entity Models and uniform term weighting. Significance is tested against the Structured Entity Model (corresponding row in Table 4.4). Best scores for each year are typeset boldface. OutR stands for outgoing, InR for ingoing relations.

	Pred. type				**2010**			**2011**	
Name	**Att**	**OutR**	**InR**	**MAP**	**P@10**	**NDCG**	**MAP**	**P@10**	**NDCG**
0.25	0.25	0.25	0.25	0.235‡	0.326‡	0.439‡	0.242‡	0.214‡	**0.374**†
0.50	0.50			0.224‡	0.336‡	0.413‡	0.204‡	0.200‡	0.320‡
.35	.35	.15	.15	**0.256**‡	**0.364**‡	**0.461**‡	**0.244**‡	**0.224**	0.372†

Starting from a baseline using standard document retrieval techniques, we introduced three expansions: (1) a heuristic for extracting textual content from URI descriptors, (2) a two-field representation, based on title and content, and (3) boosting entities from trusted domains. We showed that our approach is highly competitive and—to the best of our knowledge—outperforms all previously reported results on these data sets. The extent of our improvements is somewhat surprising because our approach is straightforward in terms of transforming RDF triples into a flat structure and applying known IR techniques. We observe that the extended URI preprocessing component accounts for the majority of the improvements.

The main research question we have been concerned with is the modelling of entities, described in the form of subject-predicate-object triples, for text-based retrieval. Prior work suggested two main directions: (1) collapsing all text associated with the entity into a single flat-text representation and then using standard IR models for document retrieval, and (2) grouping predicates together into a small number of categories, representing them as document fields, and applying fielded extensions of document retrieval methods. We formalised both strategies using generative language models and termed them *Unstructured Entity Model* and *Structured Entity Model*. Experimental evaluation was performed using the 2010 and 2011 test sets of the Semantic Search Challenge evaluation campaign. The structured approach was shown to outperform a very strong baseline provided by the unstructured model.

Specifically, we grouped predicates based on their types into four categories: name, attributes, outgoing, and incoming relations. Out of these, name and attributes proved to be the most useful ones for our task. The combination of multiple predicate types failed in the unstructured case; incorporating relations did not bring in any improvements over using names or attributes alone. The reason for this behaviour is that there is more textual content for relation type predicates (35.5 and 13.3 terms on average for in- and out-relations, respectively) than for name (3.95) or attributes (26.6), and this way the entity language model may lose the focus from the entity itself. The structured model represents predicate types as language models and considers their weighted linear combination. Taking all types with equal weights delivers very strong results and outperforms the existing state-of-the-art. Importantly, we showed that relations can contribute to overall performance under this approach.

While the above two models perform well empirically, they suffer from a severe limitation: they abandon a large part of the semantics behind the data. We propose a novel approach, referred to as *Hierarchical Entity Model*, that is capable of preserving the semantics associated with entities. It organises predicates into a two-level structure with predicate types on the top level and individual predicates on the bottom level. The weight of predicate types can be set similarly to the *Structured Entity Model* model, while the importance of individual predicates can be estimated in an unsupervised way. Our experiments showed that modelling individual predicates of a given type is more effective than folding their contents into a flat representation. When multiple predicate types are combined for the entity, the *Hierarchical Entity Mpdel* delivers substantially higher results than the *Unstructured EntityModel* baseline, but fails to outperform the *Structured Entity Model*. One issue for future investigation is to find out why the combination does not benefit from the improved component models. It may be the case that the entity model is now more semantically informed but the query representation and retrieval model are yet unable to exploit this fact. Future work will mainly be concerned with extending the current approach by segmenting the query and mapping its components to individual predicates within the hierarchical entity model.

Recap: Part II

In the **Entity Search** part of the thesis we introduced entity search in the Web of Data. More specifically, we described two tasks, entity search (ad-hoc keyword search in the WoD) and list search (search for a set of results answering a given query).

We started our investigations in this context of the Semantic Search Challenge and described our submissions in Chapter 3. Our main focus there was to use both BTC and DBpedia and achieve competitive results based on probabilistic structured retrieval models using a simple fielded representation. Motivated by our experimental results, we then focused on the entity modelling component, which we identified as promising direction.

In particular we investigated fielded models. To this end, we proposed new models based on structured search in Chaper 4. First, we provided strong baselines based on advanced structured retrieval models. In the course of that chapter we proposed a new model, the *Hierarchical Entity Model*. This model takes into account both the strong full-text background model and the structured nature of the WoD. Experimental results showed the competitiveness of our proposed solutions.

Part III

Distributed Aspects

After having looked at entity search in a centralised setting in Part II; that is, we assumed all entities to be located on on central server. We now proceed to investigate distributed aspects motivated by the inherently distributed characteristics of the Web of Data (i.e., the WoD sources are located on different servers as indicated by the domain names of the individual entities). Distributed information retrieval is commonly denoted as federated search, which we will use as a basis for our investigations. We first introduce the federated search use case in Chapter 5 and describe a framework for entity search in this context. We then continue with describing advanced techniques in Chapter 6. There, we exploit the semantic structure of the WoD to improve entity retrieval in a federated search environment. Finally, we go one step further by adapting the entity search use case to a P2P environment in Chapter 7.

Chapter 5

Federated Entity Search

In this chapter we extend the techniques of Chapter 3 and Chapter 4 to a distributed scenario. To this end, we formalise the federated search problem in a language model setting and investigate the fundamentals of entity search in federated settings. The main questions we ask in this context is: How can we apply the entity search techniques developed for the centralised case to a distributed one? Further, we provide baseline results for all components of federated search.

5.1 Introduction

In this chapter we continue to focus on the *ad-hoc entity retrieval* we defined in Section 2.6. There is a growing body of work on the subject, including indexing structures [34], retrieval models [21, 39, 80], and evaluation methodology [46, 92].

All existing work on entity search, however, assumes that a centralised index, encompassing the contents of all individual data sources, is available. Instead of expending effort to crawl all Web of Data sources—some of which may not be crawleable at all—*federated search* or *distributed information retrieval (DIR)* techniques[1] route the query to the search interfaces of suitable collections that are usually distributed across several locations [101]. For example, the query "entity retrieval" may be passed to a bibliographical database of research articles (e.g., DBLP[2]), while for the query "San Antonio" collections containing information about the city, such as GeoNames[3] or DBpedia,[4] might be more appropriate.

Federated search is a well studied subject in the context of document retrieval. Three major challenges are distinguished in this area: (i) *collection representation*,

[1]In the course of this chapter, we will use federated search and distributed IR as synonyms.
[2]`http://dblp.org/`
[3]`http://www.geonames.org/`
[4]`http://dbpedia.org/`

i.e., how to represent the contents of each collection, (ii) *collection selection*, i.e., how to select a subset of collections that are most likely to contain relevant documents, and (iii) *result merging*, i.e., how to merge the results returned from the selected collections into a single ranked list. Each of these questions have been researched extensively and we base our approach on insights gained from these developments. The main distinctive element, however, that sets our efforts apart from prior work is that our unit of retrieval is entities, not documents. To be able to apply existing DIR methods, we need to create textual representations of entities that are described in terms of RDF triples. The simples solution is to create a flat-text representation from the entities; this corresponds to the Unstructured Entity Model from Section 4.4.2.

In this chapter, we seek to answer the following research questions: *How does distributed information retrieval perform compared to centralised retrieval on the ad-hoc entity search task in the Web of Data? How can federated search be formalised in language modelling framework?*

We chose a generative language modelling setting to study the entity retrieval problem, i.e., we base our work in this Chapter on the Unstructured Entity Model introduced in Section 4.4.2. Our motivation for using language models is twofold: (i) their solid theoretical foundations coupled with good empirical performance [124], and (ii) the availability of a great deal of prior work on both distributed information retrieval [40, 99, 106] and entity retrieval [8, 11, 22] that we can build upon. LMs have also been successfully applied to a wide range of entity-related search tasks; see, e.g., [9, 30, 39, 54, 80, 81]. Further, this way we open up for integrating the techniques introduced in Chapters 3 and 4.

The remainder of the chapter is organised as follows. We shortly survey related work in the area in Section 5.2 and show an overview of the federated entity search architecture we work with in Section 5.3. We formalise methods for the main components of federated search systems in a language modelling framework in Section 5.4. We continue by outlining our experimental setup for federated search in Section 5.5. In Section 5.6, we present the results of our experimental evaluation and finally draw conclusions in Section 5.7.

5.2 Related Work

In the following we will provide an overview of related research before we continue by introducing the typical federated search setting in Section 5.3.

Federated search has advanced to a mature research area dealing with querying multiple, geographically distributed information repositories; we outlined the federated search scenario in Section 2.4.1. Both term weighting and normalisation are identified as major problems when participating collections change more frequently [117], for both require global document frequency information. Viles and French study the impact of document allocation and collection-wide information in distributed archives [118]. They observe that even for a modest number of sites, dissemination of collection-wide information is necessary to maintain retrieval effectiveness, but that the amount of disseminated information can be relatively low. In a smaller scale distributed system, it is possible to use a dedicated server for collecting accurate term-level global statistics [69]. However, this approach is clearly not appropriate for large-scale systems.

Query-based sampling is applied to the resource selection problem in [2, 3]. Predictive Likelihood is used to assess the adequacy of an acquired resource description, as opposed to other approaches, where a fixed number of samples is drawn from each collection. The authors show that their technique minimised overheads while maintaining selection performance. A critical setting in this context is how to set the sample size to find the balance between good sample levels and effectiveness of the sampling process as well as to consider the size of the collection at hand. We will return to these issues later on in Section 5.5.3 when we investigate the underlying setting (cooperative/uncooperative) and put the sampling problem in context with our work.

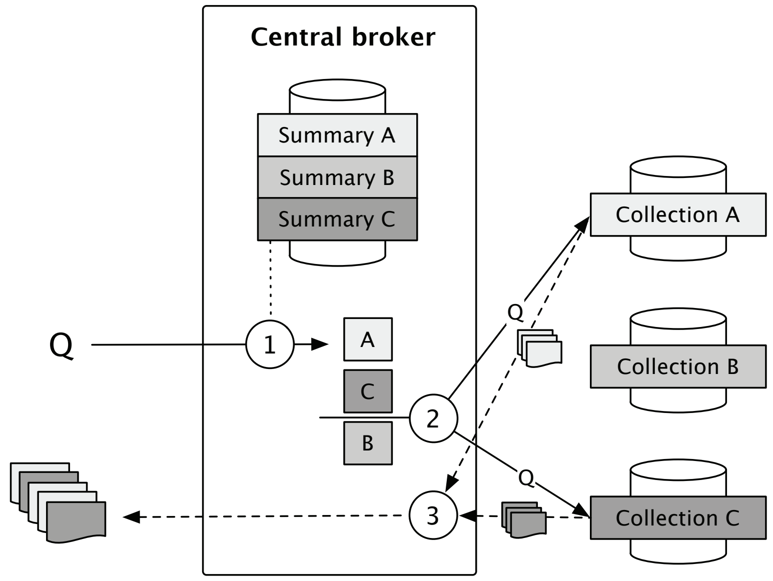


Figure 5.1: Schematic overview of a typical broker-based distributed information retrieval system.

5.3 An Architecture for Federated Entity Search

We now present a high-level overview of the typical broker based architecture in which the retrieval process is coordinated by a central server in Figure 5.1. Its

Table 5.1: Variable naming conventions for probabilistic approaches in the distributed entity search scenario.

Variable	**Gloss**
Q, E, C	Query, entity, collection
D_E	Document representing entity E
$t \in Q$	Query term
C_E	Collection that entity E belongs to
$tf_{t,Q}$	Number of times term t occurs in Q
$P(t\|D_E)$	Maximum-likelihood entity model
$\mathcal{C}$	Set of all collections
$P(t\|\theta_E)$	Smoothed entity model
$\|C\|$	Number of entities in collection C
$P(t\|G)$	Global background language model

main components are a central broker and individual collections, named A, B, and C. For now we assume the broker can send queries to all involved collections and store the resultant documents. However, different access levels are possible ranging to full access via the broker in a cooperative environment. A user query (Q) is first issued to the broker, which is responsible for forwarding it to the right collections and collecting the individual ranked results, as shown on the left side of the figure. To the end of deciding which collections are the best ones to answer a certain query, the broker has the task of modelling the individual collections via an adequate *collection representation* (1). Based on this representation, a ranking of collections is created first at the broker, then some of these collections are selected (2); this step is called *collection selection*. The query is then forwarded to all of these collections and the broker requests them to generate results for the input query. In our example, collections A and C process the query and return ranked lists of results. In a final step, these are combined in the *result merging* step (3) in order to return a single result set to the user. All three steps are depicted as numbers in circles in Figure 5.1.

We continue with introducing baseline versions for each of these components (i.e., the three steps depicted as numbers in circles) in Section 5.4. Note that we will put a special focus on how cooperative and uncooperative environments can be supported, a topic which we will further investigate in Section 5.5.3.

5.4 Baseline Models

In this section, we introduce the baseline models we use throughout the remainder of the this chapter and Chapter 6. We further refer to Table 5.1 for the notation which will be used.

5.4.1 Entity Modelling

The baseline entity model we apply is rather straight-forward: every unique subject found in the collection is an entity. We assign the text value of the object of every outgoing predicate as an entity's content field. Further, we employ the Unstructured Entity Model from Section4.4.2. To make sure this chapter is self-contained, we briefly introduce this model here.

Candidate entities (E) are ranked according to their probability of being relevant given query Q: $P(E|Q)$. Instead of estimating this probability directly, we apply Bayes' rule and drop the denominator as it does not influence the ranking (for a given query):

$$P(E|Q) = \frac{P(Q|E)P(E)}{P(Q)} \overset{\text{rank}}{=} P(Q|E)P(E). \tag{5.1}$$

$P(E)$ is the prior probability of choosing a particular entity E, that we subsequently attempt to draw the query Q from, with probability $P(Q|E)$. Here, we assume that $P(E)$ is uniform, thus, does not affect the ranking. Entity priors could be used to incorporate query-independent features into the ranking, for example, based on the RDF graph structure [26, 33].

Each entity E is represented by a multinomial probability distribution over the vocabulary of terms. The entity model θ_E is used to predict how likely the entity would produce a given term T, that is, $P(T|\theta_E)$. Assuming that query terms are sampled identically and independently, the query likelihood is obtained by taking the product across all the terms in the query, such that:

$$P(q|\theta_E) = \prod_{t \in Q} P(t|\theta_E)^{tf_{t,Q}}, \tag{5.2}$$

where $tf_{t,Q}$ is the (raw) frequency of term t in the query. Note that $P(t|\theta_E) > 0$ must be ensured for all vocabulary terms, otherwise the product might end up being zero.

The simplest approach to constructing entity models is to fold all text associated with the entity into one “bag-of-words”. Following the standard language modelling approach to document retrieval, we implement the entity model as a Dirichlet-smoothed multinomial distribution:

$$P(t|\theta_E) = \frac{tf_{t,E} + \mu P(t|\theta_c)}{|E| + \mu}, \tag{5.3}$$

where $tf_{t,E}$ is the raw frequency of term t in the representation of E and $|E|$ is the size of this representation, i.e., $\sum_t tf_{t,E}$. The smoothing parameter μ is set to the average entity representation size in the collection, which, based on our experience, is a sensible setting.

5.4.2 Collection Representation

In this section we present our approach for representing and ranking collections (i.e., Step 1 in Figure 5.1. We formulate this task in a generative probabilistic framework and rank collections based on their likelihood of containing entities relevant to an input query. That is, given a query Q, we estimate $P(C|Q)$ for each collection C. Instead of estimating this probability directly, we apply Bayes' rule and rewrite it to:

$$P(C|Q) \propto P(Q|C)P(C). \tag{5.4}$$

Note that $P(Q)$ has been dropped as it is the same for all collections, thus does not affect their ranking. According to Eq. 5.4 the score assigned to each collection has two components:

- *Query generator* ($P(Q|C)$): the probability of a query being generated by collection C; this can be interpreted as the collection's relevance to the query.
- *Collection prior* ($P(C)$): the *a priori* probability of selecting collection C; this is a query-independent component that tells us how likely the collection is to contain the answer to any arbitrary query.

We propose two models for estimating the query generator by drawing upon existing strategies to collection ranking and formalise them within a language modelling framework. According to our first approach (Collection-centric model) a separate representation is built for each collection from its contents, then these representations are ranked using a standard language modelling approach. In our second approach (Entity-centric model) we compute the relevance of each individual entity within the collection, then aggregate these scores to determine the collection's relevance. Our choices for the collection prior are discussed in §5.4.2. Also keep in mind that our focus throughout this chapter is on modelling collections (as opposed to modelling entities). Therefore, we assume here that for each entity E, a document representing that entity, D_E, has already been created.

Collection-centric Model

One of the simplest approaches to resource selection is to treat each collection as a single, large document [24]. Once such a pseudo-document is generated for each collection, we can rank collections much like documents. In a language modelling setting this ranking is based on the probability of the collection generating the query. Assuming independence between query terms, we put:

$$P(Q|C) = \prod_{t \in Q} P(t|\theta_C)^{tf_{t,Q}}, \tag{5.5}$$

where $tf_{t,Q}$ is the number of times term t is present in the query Q and $P(t|\theta_C)$ is the probability of term t in the collection's language model, computed using a mixture model:

$$P(t|\theta_C) = (1 - \lambda_G)P(t|C) + \lambda_G P(t|G), \tag{5.6}$$

where $P(t|C)$ and $P(t|G)$ are the probabilities of term t given the collection and the a global (cross-collection) language models, respectively, and λ_G is the smoothing parameter (to ensure that $P(t|\theta_C)$ is always greater than zero). To estimate $P(t|C)$ we aggregate the term probabilities from all entities in the collection using the following equation:

$$P(t|C) = \sum_{E \in C} P(t|D_E) P(E|C). \tag{5.7}$$

Here, $P(t|D_E)$ is the maximum likelihood estimate (i.e., the relative frequency) of term t in the document representation of entity E. Note that individual entities do not necessarily have to contribute evenly to the probability mass representing the collection; their relative importance (with respect to the collection) is controlled by $P(E|C)$, which we will come back to at a later stage. The global background language model is also a maximum likelihood estimate; $P(t|G)$ is set to the relative frequency of term t across all collections.

Putting our choices together, the final formula for ranking collections under the Collection-centric approach is:

$$P(Q|C) = \prod_{t \in Q} \Big\{ (1 - \lambda_G) \big(\sum_{E \in C} P(t|D_E) P(E|C) \big) + \lambda_G P(t|G) \Big\}^{tf_{t,Q}}. \tag{5.8}$$

In comparison with previous work on resource ranking in DIR, this model is similar in spirit to the CORI algorithm, which creates pseudo-documents for each collection using corpus term frequency statistics [23]. Our approach is also similar to the language model based resource selection approach in [106]; in fact, they are identical up until Eq. 5.6. The difference is in how the collection representation $P(t|C)$ is obtained; Si et al. [106] employ query-based sampling, while we take a weighted average of (entity) document term probabilities. Our model also bears resemblance to the *large document model* proposed by Elsas et al. [40] for blog feed search. The main difference lies in the actual estimation of the collection model; [40] use a term dependence model, while we assume full term independence (motivated by the need for computational efficiency). Moreover, in [40] all documents are weighted equally, while our model can incorporate document (in our case: entity) importance.

Entity-centric Model

Instead of creating a direct term-based representation of collections, our second approach models and queries individual entities, then aggregates their relevance estimates as follows:

$$P(Q|C) = \sum_{E \in C} P(Q|E, C) P(E|C). \tag{5.9}$$

This generative model has two components: (1) the probability of the query being generated by the entity and the collection, $P(Q|E, C)$, simply put, the entity's relevance to the query, and (2) the probability of the entity given the collection,

$P(E|C)$, that can be interpreted as the entity's importance within the collection. We estimate the query generation probability using the following mixture model (in which we assume conditional independence between the query and the collection given the entity):

$$P(Q|E,C) = \prod_{t \in Q} \big((1-\lambda_G)P(t|\theta_E) + \lambda_G P(t|G)\big)^{tf_{t,Q}}, \tag{5.10}$$

where $P(t|G)$ is the global background language model and $P(t|\theta_E)$ is the probability of term t in the entity's language model.

It is worth noting that Eq. 5.10 employs smoothing on two levels: (1) on the entity's level, by smoothing the entity document with the collection , and (2) on the collection level, by mixing with the global background model using coefficient λ_G. Therefore, this query-likelihood component enables us to combine evidence from the entity, from the collection, and from the global background model. By substituting Eq. 5.10 back into Eq. 5.9, the final estimation for the Entity-centric model is as follows:

$$P(Q|C) = \sum_{E \in C} P(E|C) \prod_{t \in Q} \big((1-\lambda_G)P(t|\theta_E) + \lambda_G P(t|G)\big)^{tf_{t,Q}}. \tag{5.11}$$

This model resembles the relevant document distribution estimation (ReDDE) collection selection algorithm [104]. While algebraically being very similar, the fundamental difference is that ReDDE assumes an uncooperative environment and relies on a sampling mechanism to estimate document relevance scores. Also, ReDDE directly incorporates the collection size into the scoring formula as a multiplication factor, while we can accommodate it (alone or in combination with other query-independent factors) in the form of a collection prior. Our high-level approach (Eq. 5.9) is equivalent to the *small document* blog feed search model of Elsas et al. [40], but we differ in the estimation of its components. Specifically, [40] employ a full dependence query model for $P(Q|E,C)$ and use $P(E|C)$ to measure the "centrality" of a document (this concept is specific to the blog feed search task they are solving).

Common Components

To complete our collection ranking models, two probabilities remain two be defined; these are common to both the Collection-centric and Entity-centric approaches.

Entity importance. The probability $P(E|C)$ expresses the importance of a given entity E within the collection C (cf. Eqs. 5.8 and 5.11). It can be used to incorporate query-independent features (for example, based on link structure or popularity) to favour certain entities over others. To remain focused, we assume here that all entities within a collection are equally important, i.e., we set $P(E|C) = 1/|C|$.

Collection priors. To estimate the *a priori* probability of a collection, $P(C)$, we consider two alternatives. The simplest choice is to assume that all collections are equally important: $P(C) \propto 1$. We refer to this as the *uniform* prior. Intuitively, larger collections are more likely to contain relevant entities to any information need. According to the *collection size* prior, we set the $P(C) \propto |C|$. In the interest of readability, we do not include the normalisation factors in the above equations; we normalised $P(C)$ so that $\sum_{C \in \mathcal{C}} P(C) = 1$.

Practical Considerations

The Collection-centric method represents each collection by a single term distribution, $P(t|C)$. This probability can be pre-computed and possibly stored in the index. If we put smoothing with the global collection aside, computing the query likelihood amounts to a single lookup for each query term, and then multiplying these probabilities. This makes this approach extremely efficient. On the flip side, since the collection is modelled as a whole, changes concerning the representation of entities (e.g., considering a fielded representation instead of flat-text) cannot necessarily be integrated easily, as these would need to be lifted to the collection level.

The Entity-centric model, on the other hand, can directly benefit from any improvements on the entity level, as it aggregates entity relevance scores. Therefore, a better ranking of entities should in principle lead to a better ranking of collections. A serious disadvantage of this approach, however, is that is has to iterate through all entities matching the query, to be able to determine the collection's score. Applying this model in practice, especially when low latency is required, would involve additional investments in engineering effort; one possible solution for improving efficiency is to apply heuristic cut-offs and to look only at the top k relevant entities within each collection.

5.4.3 Collection Selection

In the previous subsection we established mechanisms for ranking collections in order of relevance to a query. Next, we need to identify a set of collections that are likely to contain most relevant entities; this corresponds to Step 2 in Figure 5.1. The problem is generally addressed by choosing a fixed cutoff ahead of time; for example, Si and Callan [104] use 5 to 20. We refer to this method as *top-K collection selection.* (SUSHI [112] offers an alternative selection strategy; it is briefly discussed in Section 2.4.1.)

Formally, let $r(c, q)$ be the rank of collection c for query q according to the collection ranking component (where the highest ranked collection has rank value 0). The set of selected collections, $S_c(q)$, is then defined as follows: $S_c(q) = \{c|r(c, q) < K\}$, where K is a fixed cutoff value.

5.4.4 Result Merging

Language Model Result Merging

A major difficulty imposed by the distributed nature of the environment is that the entity query-likelihood scores ($P(Q|E)$), computed in Eq. 5.2) are not comparable across collections, as the $P(t|C)$ component is based on local corpus statistics. There can be several magnitudes of difference in the absolute values of this probability between collections, depending on their size. The problem of merging results with incomparable scores from individual data collections has been studied extensively in the literature.

Si et al. [106] introduced an elegant and theoretically sound solution for result merging as part of their language modelling framework for DIR. While we differ in the actual estimation of the underlying components, we can still adopt their approach unchanged. To remove the bias within the original entity scores caused by the different collection statistics, the final ranking of entities is computed using the following expression:

$$\log P(Q|E) \propto \log P(Q|E, C_E) - \log(\frac{\alpha}{1-\alpha} P(C_E|Q) + 1), \tag{5.12}$$

where C_E denotes the collection to which entity E belongs, and $\alpha \in [0..1]$ is a parameter that controls the relative importance of Q and E. The component $P(Q|E, C_E)$ is estimated using Eq. 5.10. As to the term $P(C_E|Q)$, we rewrite it using Bayes' rule:

$$P(C_E|Q) = \frac{P(Q|C_E)P(C_E)}{\sum_{C'} P(Q|C')P(C')}, \tag{5.13}$$

and use the $P(Q|C)$ and $P(C)$ values computed in the collection ranking step in Section 5.4.2. For more details and for the derivation of Eq. 5.12, we refer the reader to [106]. It is worth noting that in our Web of Data setting the same entity (identifier) might be present in multiple sources. Therefore, we need to deal with the issue of duplicate detection in this merging phase. If the same entity has multiple occurrences, we only return the one with the highest log-likelihood score.

5.5 Experimental Setup

In this section we detail our experimental setup. As no standard test collection exists for the scenario we are targeting, we describe the distributed testbed we have developed for evaluation purposes in Section 5.5.1.

5.5.1 Distributed Environment

Our setup is based on the test suites of the 2010 and 2011 editions of the Semantic Search (SemSearch) Challenge [20, 46]. The data collection used there is the Billion

Triple Challenge 2009 (BTC-2009) data set. It comprises about 1.14 billion RDF statements and describes entities from domains like *dbpedia.org*, *livejournal.com*, or *geonames.org*.[5] The task addressed at SemSearch is ad-hoc entity search: given a keyword query, targeting a particular entity, provide a ranked list of relevant entities, identified by their URIs. There are two topic sets, consisting of 92 and 50 keyword queries for years 2010 and 2011, respectively. The queries were sampled from web search engine logs. Relevance judgments are provided on a 3-point scale (excellent, fair, and irrelevant) and were collected using crowdsourcing.

By relations we mean RDF triples that have the entity as the subject and another URI (i.e., entity) as their object. So far, these objects were treated (and parsed) as regular terms. Our intuition is that it is more meaningful to replace these entity identifiers with the name of the entity. One way of doing that would be to look up the object URI in the corresponding resource and use the name attribute from the entity description obtained (if available). With this solution, two difficulties arise, specifically, when the URI points to an external collection: (i) it requires communication between collections, and (ii) it requires the knowledge of the schema of the target collection (i.e., the label of the predicate that holds the name attribute). A simple heuristic we settled for is to use the string part of the URI after the last slash as the entity name. Additionally, we make sure characters like underscores, dashes, brackets, etc. are treated as whitespaces.

To create a distributed environment, we have chosen the top 100 largest second-level domains from BTC-2009, in terms of the number of entities they contain, and indexed them as 100 separate collections. As with typical federated search testbeds, the collections are disjoint, they do not overlap [101]. The number of partitions we use follows standard practice, see, e.g., [100, 121], and is considered sufficiently large. We use all SemSearch queries (that is, from both 2010 and 2011) that contain at least one relevant result from one of the top 100 domains; this amounts to a total of 136 queries. We restricted the corresponding relevance judgments to our set of selected collections, but apart from the filtering we use the SemSearch assessments unchanged. We will refer to this test set throughout the paper as *BTC*.

The distribution of collection sizes for the top 100 domains, shown in Figure 5.2, resembles a Zipfian distribution, where the three largest domains account for almost 30% of the collection. While this is not at all unexpected or unusual, a somewhat unique property of this test set is that relevant documents do not follow the same distribution, but are very highly biased towards the biggest collection, DBpedia. In fact, 73% of all relevant results originate from DBpedia (note that this is not due to our selection of top 100 collections, as DBpedia holds 59% of all known relevant results, without any domain restrictions). To ensure that our findings are not misguided because of this anomaly, we created two more test sets, representing distributed environments with different characteristics.

BTC\DBpedia is the same as the BTC test set, but the DBpedia collection is excluded. This set, therefore, contains 99 separate collections. Consequently, DBpedia results have also been removed. There are 20 queries for which all relevant

[5] `http://km.aifb.kit.edu/projects/btc-2009/`

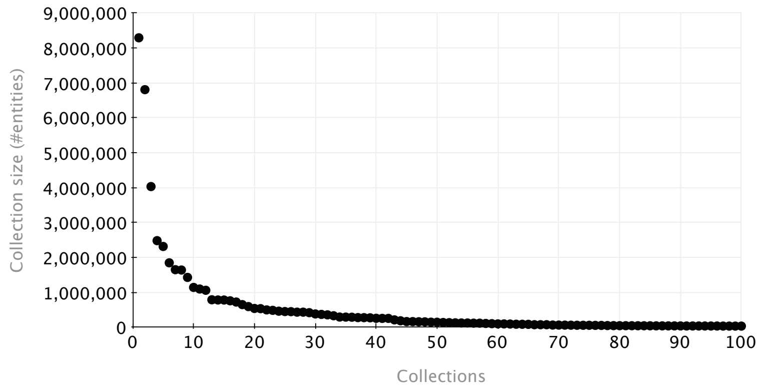


Figure 5.2: Distribution of collection sizes in BTC-2009.

results come from DBpedia; these have been omitted from the query set, leaving 116 queries in total. Here, the distribution of relevant results is more evenly distributed—this collection represents a typical linked data collection.

We also look at the *DBpedia* subset, on its own. Instead of using the version that is part of BTC, we considered the full version. Specifically, we used its most recent dump in version 3.7 and indexed all infobox predicates as well as labels and short abstracts of its 8.8M entities. The reason for doing so is that the BTC collection is based on a Web crawl, including some degree of noise. DBpedia, on the other hand, can be considered a more "clean" and uniform collection. We randomly distributed the DBpedia data set into 100 individual collections of equal size. The resulting collections are by no means organised (like topically or temporally) which implies that also the relevant documents are randomly distributed across all collections. Due to its random distribution, we expect it to be the most difficult setup in this context with respect to collection selection (all sub-collections are very similar to each other).

Table 5.2 presents descriptive statistics of the three test collections we developed and which will be used in the Chapters 5 and 6 alike.[6]

5.5.2 Ground Truth and Evaluation Metrics

To evaluate *collection ranking*, we use standard IR evaluation metrics: Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Normalised Discounted Cumulative Gain (NDCG). We obtained ground truth from the original SemSearch relevance assessments as follows. For the metrics that work with binary judgments, a domain is considered relevant if it contains at least one entity that was judged

[6]The queries and relevance judgments we derived from the SemSearch data set, as well as the DBpedia splits are available at `http://krisztianbalog.com/resources/spire-2012/`

Table 5.2: Overview of test collections.

	BTC	**BTC \DBpedia**	**DBpedia**
#Entities	68.8M	60.5M	8.8M
#Collections	100	99	100
#Queries	136	116	130
Avg. #rel. entities /query	14.9	4.8	10.1
Avg. #rel. collections /query	3.4	2.8	9.4

relevant for the query. When computing NDCG, we set the gain for each collection to the number of relevant documents the collection contains.

For evaluating *collection selection*, we introduce two metrics, $\mathcal{P}_K$ and $\mathcal{R}_K$, which are rough analogues of the classical precision and recall measures and consider the effectiveness of the collection selection method alone. We base our definitions on the metrics proposed in [42], and use the variant by Thomas and Shokouhi [112] for $\mathcal{R}_K$. If K collections are selected, $\mathcal{P}_K$ is the fraction of collections that contain (any) relevant entities, while $\mathcal{R}_K$ is the ratio of (all) relevant entities held by these collections. Both metrics range from 0 to 1, where higher values are desired. We also measure the average number of collections selected; this serves as our metric of efficiency.

5.5.3 Cooperative and Uncooperative Environments

In uncooperative environments collections do not publish their term statistics; their contents can only be accessed via their (public) query interfaces. The broker then must gather information by sending probe queries to the collection and analysing the returned documents (entities), a technique called *query-based sampling* (QBS) [25]. QBS has been widely used in federated search (see, e.g., [83, 84, 104, 112]). The general process is as follows:

(1) Select an initial probe query. In the simplest case the query is only a single term, one that is likely to generate many results.

(2) Run the query against the collection and download the top n returned documents.

(3) Update the collection's representation with the downloaded (and previously unseen) documents.

(4) Repeat the process, i.e., select another probe query and go to Step (2), until the stopping criterion is met. The stopping criterion is usually defined in terms of the total number of unique documents sampled. The probe query can be chosen from a reference dictionary or from the documents already sampled.

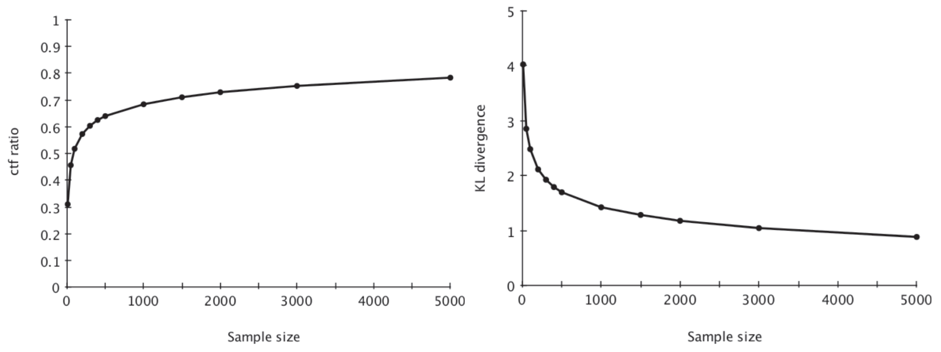


Figure 5.3: Impact of sample size on the quality of resource descriptions. (Left) ctf ratio, (Right) KL divergence.

We use a widely-accepted method by Callan and Connell [25]; the probe queries are single terms. The initial query term is drawn randomly from a reference collection (i.e., the background language model) and subsequent probe queries are chosen randomly from the documents already sampled. Callan and Connell [25] suggest to use $n = 4$ and continue the sampling until $300 - 500$ unique documents have been downloaded from the collection. While this setting is widely used (e.g., [83, 84, 104, 112]), it has been shown that 300 unique documents are insufficient for larger collections [103]. We wish to revisit the setting of this parameter under the WoD environment. We perform two sets of experiments: (1) in intrinsic evaluation, we directly measure the quality of collection representations, and (2) in extrinsic evaluation, we measure how sample sizes impact the performance of collection ranking.

To measure the extent to which resource descriptions are representative of their original collections, we employ two separate metrics. The first metric, *ctf ratio*, proposed by Callan and Connell [25], measures the vocabulary correspondence, that is, the proportion of terms in the collection (C) that are covered in the sample (S_C):

$$ctf_{S_C,C} = \frac{\sum_{t \in S_C} tf_{t,C}}{\sum_{t \in C} tf_{t,C}}, \tag{5.14}$$

where $tf_{t,C}$ is the frequency of term t in the collection. It can be seen from Eq. (5.14) that ctf values are between 0 and 1, and that frequent terms contribute more than infrequent (albeit maybe more representative) ones.

The second metric compares the language models of representation sets with that of the original collections. One commonly used method for comparing term distributions is the Kullback-Leibler divergence (KL):

$$KL(\theta_{S_C}||\theta_C) = \sum_{t \in C} P(t|\theta_{S_C}) \log \frac{P(t|\theta_{S_C})}{P(t|\theta_C)}. \tag{5.15}$$

Here, θ_{S_C} and θ_C are language models of the sampled and the original collections, respectively. KL values range from 0 to ∞, where 0 means that the two distributions are identical.

The results are shown in Figure 5.3; the numbers reported here are averages over 10 runs. The shapes of these curves are similar to those reported in [25], but substantially more documents need to be examined in our case. Specifically, Callan and Connell [25] report a *ctf ratio* of over 80% achieved with a sample of 300 documents on the TREC-123 collection, comprising of one million documents. In our case, we cannot reach 80% *ctf* even with a sample of 5000 documents. The impact of the sample size on the coverage was also investigated in [103]. The authors state that larger sample sizes may be necessary, particularly when other evaluation criteria are used. Our results indicate that our setting requires more extensive sampling of data than "traditional" text collections do. In the next chapter, we propose an alternative collection representation strategy that can help to overcome this concern (see Section 6). Besides, the results show that a varying sample size has a profound impact on the individual language models. This in turn shows that methods working well in cooperative settings can be adapted to the uncooperative setting to a very large degree by increasing the sample size alone. For this reason we chose to report results for cooperative settings in the course of this chapter and the next, and leave the issue of sample size estimation to future work.

5.6 Experimental Evaluation: Baseline Models

In this section, to evaluate our entity modelling approaches, and to provide an upper bound for retrieval performance on the end-to-end task, we first report results using a single centralised index (5.6.1). We further perform and evaluate each step of the DIR process, i.e., collection representation (5.6.2), collection selection (5.6.3), and result merging (5.6.4).

5.6.1 Centralised Retrieval

The top three rows of Table 5.3 correspond to centralised counterparts of the three distributed settings we employ throughout the paper. These results constitute numbers comparable to SemSearch submissions, to guarantee comparability even further, we present the numbers for the single years in the following rows; these can be directly compared to SemSearch results and show that we operate with strong baselines. The last two rows represent the numbers when only the most frequent domains are considered for evaluation (i.e., non-frequent domains are discarded for our evaluation). Essentially, these results represent an upper bound retrieval performance limit for our distributed approaches, i.e., an optimal output that can be produced from a centralised index when the same domains are considered that we use for distributed experiments.

Table 5.3: Entity retrieval results on a centralised index.

Test set	**Queries**	**MAP**	**MRR**	**P@10**	**NDCG**
All domains/splits					
BTC	ALL	0.1940	0.5046	0.2599	0.3675
BTC\DBpedia	ALL	0.1310	0.2468	0.0880	0.2450
DBpedia	ALL	0.2156	0.4730	0.1900	0.3910
Disregarding non-frequent domains					
Test set	**Queries**	**MAP**	**MRR**	**P@10**	**NDCG**
BTC	2010	0.1860	0.5265	0.2848	0.3860
BTC	2011	0.2088	0.4642	0.2140	0.3418
Disregarding non-frequent domains					
Test set	**Queries**	**MAP**	**MRR**	**P@10**	**NDCG**
BTC	ALL	0.1698	0.4107	0.2141	0.3291
BTC\DBpedia	ALL	0.1943	0.3419	0.1197	0.2997

Recall that our approach to modeling entities is a rather simple one; yet we achieve comparable results to the submissions to the SemSearch challenge (when our results are broken down to individual years; the results of the challenge are provided in Table 3.6).

5.6.2 Collection Representation and Collection Ranking

A key issue in collection selection is how to acquire summaries of the contents of collections (also referred to as *representation sets* or *resource descriptions*). In the cooperative case it is assumed that collections provide the broker with comprehensive information about their contents. In the uncooperative case, collections do not publish such information and representation sets have to be obtained based on a sample of documents downloaded from the collection. In our experiments we assume that for the cooperative case the broker has complete knowledge about the contents of each collection; while this is arguably an over-idealised setting, it allows us to uncover the full potential of the two collection representation strategies and to make a fair comparison between them, using these results as an upper bound. These results can be put in context in light of our results on the impact of sample sizes in 5.5.3.

We present the results for our baseline methods and the collection ranking task in Table 5.4. The results reported are for both methods (collection-centric and entity-centric), all three test collections (BTC, BTC\DBpedia, DBpedia) and with and without Collection priors. For reference, we also included retrieval results for well-known representatives of the two families of methods: CORI [23] for a lexicon-based approach, and ReDDE [104] and two variants of CRCS [100] for document-

Table 5.4: Collection ranking results in a cooperative environment. The second column indicates the collection representation method used. The third column shows whether collection priors (set proportional to the collection size) were used.

Test set	**Meth.**	**Priors**	**MAP**	**MRR**	**P@10**	**NDCG**
BTC	CC	N	0.3018	0.5170	0.1699	0.5327
	EC	N	0.5007	0.7013	0.2213	0.6370
	CORI	N/Y	0.3416	0.4751	0.1824	0.4993
	ReDDE	N	0.6442	0.9198	0.2412	0.8782
	CRCS(l)	N	0.6512	0.9240	0.2434	0.8818
	CC	Y	0.5450	0.9001	0.1978	0.8475
	EC	Y	0.6339	0.9079	0.2360	0.8718
	ReDDE	Y	0.6738	0.9590	0.2434	0.9092
	CRCS(l)	Y	0.6800	0.9648	0.2441	0.9144
BTC\DBpedia	CC	N	0.2277	0.3582	0.1111	0.4173
	EC	N	0.3826	0.5401	0.1684	0.5568
	CORI	N/Y	0.2828	0.4123	0.1239	0.4690
	ReDDE	N	0.4244	0.5935	0.1718	0.5856
	CRCS(l)	N	0.4399	0.5859	0.1735	0.5848
	CC	Y	0.2005	0.2774	0.1188	0.4025
	EC	Y	0.2990	0.3759	0.1632	0.4867
	ReDDE	Y	0.4473	0.6139	0.1735	0.5811
	CRCS(l)	Y	0.4535	0.6189	0.1726	0.5846
DBpedia	CC	N/Y	0.1391	0.2259	0.1069	0.4003
	EC	N/Y	0.1349	0.2071	0.1031	0.3929
	CORI	N/Y	0.1429	0.2486	0.1023	0.4060
	ReDDE	N/Y	0.1406	0.2590	0.0977	0.4032
	CRCS(l)	N/Y	0.1317	0.2090	0.0908	0.3901

surrogate methods. For these, we use default parameter settings as suggested in the corresponding publications, most importantly the cutoff value of 50, e.g., [100]. These methods are presented in greater detail in Section 2.4.1.

The first observation is that our language modelling based baselines (CC and EC) are as good as with existing methods from the literature. The most important finding is that both methods perform well in all settings, with an MAP value of over 0.20 for the first two collections. The results for the DBpedia collection alone are arguably much lower across all methods because of its subcollections being chosen randomly.[7] Nevertheless, both methods provide results with an MAP of more than 0.13, i.e., making it much better than the random baselines. Our next observation is that collection priors boost our results significantly for both BTC collections.

[7]The fact that they are of equal size also explains why collection priors can not be used in this setting.

5.6.3 Collection Selection

We show experimental results for the collection selection step in Table 5.5. To evaluate our results we refer to the evaluation criteria outlined in Section 5.5.2. We provide results for varying cutoff values (ranging from 1 to 15), i.e., the number of top collections selected for each method and report $\mathcal{P}_K$ as the fraction of collections that contain (any) relevant entities and $\mathcal{R}_K$ as the ratio of (all) relevant entities held by these collections. The first three blocks report results without using collection priors, i.e., all collections have an equal apriori weight. Results using collection priors are reported in the remaining two blocks (note that collection priors are not available for the DBpedia collection due to its random distribution). Our results show that collection priors have a very large positive impact on the results for the BTC collection. This can be attributed to the fact that DBpedia is the single most important collection there having an relatively big impact. Further, increasing the cutoff values almost always has a negative impact on the results (with the exception of the DBpedia collection without priors and the BTC\DBpedia collections when priors are used). We observe these results due to the fact that we already achieve a very high reciprocal rank in the collection ranking, i.e., correct collections are very often returned early.

5.6.4 Result Merging

We show our experimental results for the result merging step in Tables 5.6 and 5.7, for the BTC collections and DBpedia, respectively. We show results for both collection ranking methods and a varying number of k, i.e., the number of collections selected. For the BTC and BTC\DBpedia collections we show the results for very low thresholds, i.e., 1, 3, and 5, whereas we report for thresholds of 50, 75, and 100 for the DBpedia collection. We chose rather low numbers for k in the BTC case because the fact that each split represents a domain and its implication that the collections are very distinct. The higher settings for DBpedia are motivated by the increased difficulty of the task due to the random distribution of entities to collections, i.e., we need to select more collections to achieve performance that is comparable to that of the centralised case.

As shown in Table 5.6, the result merging results follow the same pattern as the collection ranking shown previously, namely that collection priors give a large boost to the collection-centric ranking. Without collection priors, the entity-centric method provides vastly superior results.

Considering the different thresholds, we observe that the best results are achieved by a k value as low as 2 without using collection priors, and a cutoff at 2 or 3 when using priors (depending on whether DBpedia is included or not). This implies that both collection ranking methods are competitive and manage to rank relevant collections early on in their rankings.

The results for the settings based on DBpedia only are shown in Table 5.7. These results clearly reflect the increased complexity of finding the correct collection due

Table 5.5: Collection selection results in a cooperative environment for settings with and without collection priors.

Test set	**Meth.**	$K = 1$		$K = 3$		$K = 5$		$K = 10$		$K = 15$	
		$\mathcal{P}_K$	$\mathcal{R}_K$	$\mathcal{P}_K$	$\mathcal{R}_K$	$\mathcal{P}_K$	$\mathcal{R}_K$	$\mathcal{P}_K$	$\mathcal{R}_K$	$\mathcal{P}_K$	$\mathcal{R}_K$
Without Collection Priors											
BTC	CC	0.3382	0.2541	0.2819	0.4156	0.2382	0.5322	0.1699	0.6997	0.1221	0.7510
	EC	0.5662	0.3971	0.4191	0.5484	0.3368	0.6922	0.2213	0.7742	0.1608	0.7966
BTC\DBpedia	CC	0.1795	0.1538	0.1709	0.2396	0.1453	0.2927	0.1111	0.4292	0.0809	0.4701
	EC	0.3419	0.3063	0.2934	0.4153	0.2308	0.4790	0.1684	0.6270	0.1271	0.6888
DBpedia	CC	0.0846	0.0654	0.0974	0.0899	0.1092	0.1143	0.1069	0.1583	0.0995	0.1804
	EC	0.0769	0.0538	0.0949	0.0718	0.1046	0.0933	0.1031	0.1440	0.0938	0.1743
With Collection Priors											
BTC	CC	0.8382	0.8088	0.4412	0.7850	0.3162	0.8046	0.1978	0.8563	0.1461	0.8797
	EC	0.8529	0.8098	0.5074	0.8228	0.3750	0.8475	0.2360	0.9023	0.1765	0.9242
BTC\DBpedia	CC	0.0427	0.0393	0.1510	0.2083	0.1453	0.2978	0.1188	0.4642	0.0957	0.5475
	EC	0.0769	0.0735	0.2308	0.3301	0.2154	0.4607	0.1632	0.6420	0.1311	0.7396

Table 5.6: Result merging results in a cooperative environment for settings with and without collection priors.

Test set	**Meth.**	$K = 1$				$K = 3$				$K = 5$			
		MAP	**MRR**	**P@10**	**NDCG**	**MAP**	**MRR**	**P@10**	**NDCG**	**MAP**	**MRR**	**P@10**	**NDCG**
Without Collection Priors													
BTC	CC	0.0476	0.2039	0.0702	0.0950	0.0668	0.2191	0.0922	0.1398	0.0693	0.2118	0.0879	0.1539
	EC	0.0816	0.3504	0.1085	0.1539	0.0668	0.2191	0.0922	0.1398	0.1021	0.2855	0.1149	0.2054
BTC\DBpedia	CC	0.0439	0.1018	0.0243	0.0570	0.0719	0.1634	0.0418	0.0989	0.0661	0.1491	0.0440	0.1045
	EC	0.1147	0.2564	0.0573	0.1494	0.1497	0.3140	0.0745	0.2066	0.1290	0.2648	0.0723	0.1903
With Collection Priors													
BTC	CC	0.1305	0.4514	0.1901	0.2617	0.1233	0.3388	0.1345	0.2066	0.1090	0.3025	0.1190	0.2162
	EC	0.1323	0.4663	0.1979	0.2688	0.1515	0.3498	0.1606	0.2697	0.1367	0.3411	0.1408	0.2506
BTC\DBpedia	CC	0.0276	0.0562	0.0304	0.0452	0.0764	0.1890	0.0312	0.1059	0.0876	0.2048	0.0475	0.1261
	EC	0.1018	0.1667	0.0654	0.1284	0.1396	0.2775	0.0603	0.1818	0.1597	0.3009	0.0773	0.2125

to the random distribution model we chose. MAP values, e.g., stay under the 0.1 mark until a k value of 50 for the collection-centric method. However, this scenario is the most difficult one, where no assumptions about the document distribution can be made whatsoever, yet, we achieve reasonable results around the cutoff of 50, which is a large improvement over considering all 100 collections. For the entity-centric case we observe similar behaviour. However, the results are higher even for lower values of k, e.g., at a k value of 5, we already achieve an MAP value of 0.1049. This is in line with our other observations.

5.7 Conclusions

In this chapter we introduced the federated search setting and formalised the entity search task within a probabilistic retrieval model. We discussed all components of federated search and provided baseline models for them. Another important aspect of our analysis is the investigation of sample size and quality for federated entity search.

Our findings suggest that the amount of sampling needed for collections of this size is prohibitive. Additionally, our experiments show that sufficient coverage of the collection is much more difficult to achieve than for other collections which have been worked with in the past. There are several reasons for this behaviour: the mere size of the collection, the structure of entity data, and the nature of entity queries. We specifically want to point out the fact that entity queries typically contain (or exclusively consist of) an entity's name. This fact can be exploited in terms of collection representation as will be shown in the next chapter.

Table 5.7: Result merging results in a cooperative environment for the DBpedia collection.

Test set	**Meth.**	**MAP**	**MRR**	**P@10**	**NDCG**	**MAP**	**MRR**	**P@10**	**NDCG**	**MAP**	**MRR**	**P@10**	**NDCG**
		$K = 50$				$K = 75$				$K = 100$			
DBpedia	CC	0.1375	0.4105	0.1423	0.2666	0.1661	0.4339	0.1700	0.3172	0.1699	0.4129	0.1685	0.3205
		$K = 5$				$K = 50$				$K = 75$			
	EC	0.1183	0.4432	0.1238	0.2045	0.1771	0.4471	0.1769	0.3317	0.1783	0.4334	0.1746	0.3361

Chapter 6

Advanced Models for Federated Entity Search

Having introduced the main building blocks of federated search systems in Chapter 5, we proceed to presenting advanced models exploiting the unique features of entity data and how they can help improving effectiveness in the entity search use case. Based on our findings when experimenting with baseline methods in the previous chapter, we propose propose a new collection ranking and selection method for entity search, called AENN. The key underlying idea is that a lean, name-based representation of entities can efficiently be stored at the central broker in a cooperative environment, which, therefore, does not have to rely on sampling. This takes into account the unique property of entities being well-described by their name.

6.1 Introduction

We focus on queries that target specific entities, mentioned by their name. While this is a rather specific scenario, Pound et al. [92] estimate that over 40% of web search queries are like this. Therefore, we study a significant problem with practical utility.

In this chapter, we consider a cooperative distributed environment and focus on two sub-problems: collection ranking and collection selection. We discuss state-of-the-art distributed document retrieval techniques that can be applied to the case of entities in a straightforward manner. For collection ranking, we build on two main families of approaches (lexicon-based and document-surrogate methods) in a unified language modelling framework. This allows for a fair comparison between approaches. For collection selection, we use top-K selection, where K is a fixed rank-based cutoff.

We introduce our novel approach, AENN, in Section 6.4. The key underlying idea is that instead of relying on sampling, the central broker maintains a complete dictionary of entity names and identifiers. Based on this lean, name-based representation, we generate not only a ranking of collections but also an *expected* ranked list of entities (that is, an approximation of the final results). This can then aid us in the collection selection step to dynamically adjust the number of collections selected, moreover, it allows for orientating the selection towards high precision, high recall, or a balanced setting.

Our experimental evaluation, reported in Section 6.5, demonstrates that AENN has merit and provides a viable alternative. On collections where names are available for entities—a reasonable precondition for our approach—AENN's effectiveness (measured in terms of precision and recall) is comparable to that of an idealised centralised approach that has full knowledge of the contents of all collections, while achieving gains in efficiency (i.e., selecting fewer collections).

6.2 Related Work

The present work lies in the intersection of entity retrieval and distributed information retrieval. Most basic techniques have been introduced in Chapter 5. In the following we will briefly reiterate the most prevalent techniques for collection selection, and put them in context with the main ideas guiding our own approach.

Several collection selection techniques rely on a central sample of all collections. This so called centralised index of all sampled documents (CSI) is then used to rank individual collections. ReDDE (Relevant Document Distribution Estimation) [104] aims at selecting a small number of collections containing the most relevant documents by estimating the number of relevant documents per collection and then using this information for collection ranking. The Centralised-rank Collection Selection Method (CRCS), similarly to ReDDE, uses a CSI [100]. One main difference to ReDDE, however, is that CRCS considers varying importance for documents according to their ranks. The contribution of a sampled document d depends on its position in the central ranking of all sampled documents. Another similar collection selection method is SUSHI, which first ranks the documents in CSI [112]. Then, SUSHI extracts the document ranking for each server in turn. After that, adjusted ranks are computed compared on the sizes of the individual collections. Curve fitting is then used to estimate the scores of unseen documents.

Similar to these mentioned approaches, we will rely on a central index, the main difference between these and our own approach, which we will present in the following, lies in the representation of entities. Instead of indexing full documents, we will rely on an index constructed purely from entity names. This is partly motivated by our findings in [81] and [80].

6.3 Baseline Methods

We refer back to the high-level overview of the distributed approach given in Chapter 5. In this chapter, we assume a cooperative environment, in which the retrieval process is coordinated by a central broker. As such, we focus on improving the first two steps of this pipeline (as shown in Figure 5.1), as these are the components where our contributions take place. Result merging is a research topic on its own; to stay focused (and also due to space considerations) we do not specifically focus on that step in this chapter. We note, however, that—assuming a reasonable results merging mechanism—improved collection selection also leads to better overall results on the end-to-end task. For matters of convenience, we sum up our baseline methods, which are explained in detail in Chapter 5, formalised in a probabilistic language modelling context.

6.3.1 Collection Ranking

In the collection ranking phase (Step 2 in Figure 5.1), we need to score collections based on their likelihood of containing entities relevant to the input query. We present two main families of approaches for this task. *Lexicon-based* methods treat and score each collection as if it was a single, large document [23, 106]. *Document-surrogate* methods, on the other hand, model and query individual documents (in our case: entities), then aggregate (estimates) of their relevance scores to determine the collection's relevance [100, 104]. As pointed out earlier, we assume a "perfect" central broker; for lexicon-based methods it means complete term statistics from all collections; for document-surrogate methods it essentially amounts to a centralised index of all entities.

We formalise both strategies in a language modelling framework and rank collections (C) according to their probability of being relevant given a query (Q), $P(C|Q)$.

Collection-centric collection ranking (CC). Following Si et al. [106], the collection query-likelihood is estimated by taking a product of the collection prior, $P(C)$, and the individual term probabilities:

$$P(C|Q) \propto P(C) \cdot \prod_{t \in q} P(t|\theta_C). \tag{6.1}$$

We set priors proportional to the collection size: $P(C) \propto |C|$. A language model θ_C is built for each collection, by collapsing all entities of C into a single large document and then smoothing it with the global language model. Here, we use Dirichlet smoothing, as we found it to perform better empirically than Jelinek-Mercer smoothing used in [106]; we set the smoothing parameter to the average collection length.

Entity-centric collection ranking (EC). Under this approach, entities are ranked by the central broker, according to their probability of relevance, and the top relevant entities contribute to the collection's query-likelihood score:

$$P(C|Q) \propto \sum_{E \in C, r(E,Q) < \gamma} P(E|Q), \tag{6.2}$$

where $P(E|Q)$ is the query likelihood of the entity, computed using a standard language modelling approach and Dirichlet smoothing (with the average entity representation length used as the smoothing parameter). Further, $r(E, Q)$ denotes the rank position of entity E (the top ranked result has rank 0, the second in line has 1, and so on). Finally, γ is a rank threshold, set to 50 based on preliminary experiments; this value has also been commonly used in the literature, see, e.g. [100, 112]. It is worth mentioning that although collection priors are not explicitly included in Eq. (6.2), larger collections are implicitly favoured, as they are more likely to have more relevant results among the top γ.

6.3.2 Collection Selection

In the previous subsection we established mechanisms for ranking collections in order of relevance to a query. Next, we need to identify a set of collections that are likely to contain most relevant entities; this corresponds to Step 2 in Figure 5.1.

The problem is generally addressed by choosing a fixed cutoff ahead of time; for example, Si and Callan [104] use 5 to 20. We refer to this method as *top-K collection selection*. SUSHI [112] offers an alternative selection strategy; we discuss it in Section 2.4.1.

Formally, let $r(C, Q)$ be the rank of collection C for query Q according to the collection ranking component (where the highest ranked collection has rank value 0). The set of selected collections, $S_C(Q)$, is then defined as follows: $S_C(Q) = \{C|r(C, Q) < K\}$, where K is a fixed cutoff value.

Before proceeding further, it is important to point out that in this chapter we consider an idealised scenario with a "perfect" central broker. This means that the broker has full knowledge about the contents of each collection. We are aware that this is an unrealistic assumption in practice, but do this for a twofold reason. One, our main research interest is in comparing the effectiveness of collection ranking and selection methods; when doing so, we wish to rule out all other influencing factors, such as the quality of sampling (a technique, typically used for building collection summaries [101, 104]). Two, we want to compare our proposed solution, to be presented in Section 6.4, against this idealised setting; as we shall show later, our novel approach can deliver competitive performance without making such unrealistic assumptions.

6.4 AENN for Federated Entity Search

In this section we introduce a novel approach to collection ranking and collection selection for federated entity search. To this end, we build on the models introduced in Section 6.3. We focus on queries that target a particular entity, mentioned by its name. A significant portion of queries in web search are formulated that way [92]. Blanco et al. [20] explain this phenomenon as follows: *"users have learned that search engine relevance decreases with longer queries and have grown accustomed to reducing their query (at least initially) to the name of an entity"* [20]. Therefore, the problem we study is a significant one, with practical utility. While traditional collection ranking and selection techniques can immediately be applied, the question arises, whether we can do better by tailoring representations and models to entities.

The central idea of our approach is aptly captured in the acronym AENN: "All that an Entity Needs is a Name." Instead of building traditional collection summaries based on (full) textual representations of entities, as was done in the previous chapter, we only use their names and maintain a complete dictionary of entity names and identifiers at the central broker. This is a viable alternative as it requires only limited cooperation from the distributed collections (i.e., we need to be able to request a list of entities, with name and ID, they contain) and features minimal network traffic. Based on this lean, name-based representation, we can generate not only a ranking of collections but also an "expected" ranked list of entities (that is, a prediction of the final result list, that we would see after the merging step). This expected ranking serves as a basis of an entity-centric collection ranking, which, in turn, we utilise to aid us in the collection selection step by dynamically adjusting the number of collections selected. It is important to note that AENN is only used for collection ranking and selection; the next step in the processing pipeline (that we do not perform here) is to request the selected collections to generate a ranked list of entities given the input query. The collections may use the retrieval method of their choosing to perform this local ranking.

6.4.1 Collection Ranking

Our initial experiments with collection-centric (CC) and entity-centric (EC) collection ranking strategies suggest the two different approaches work best with different queries. Therefore, we expect to maximise performance, by taking a linear combination of the two methods:

$$AENN(C,Q) = (1-\lambda) \cdot CC(C,Q) + \lambda \cdot EC(C,Q), \tag{6.3}$$

where $CC(C,Q)$ and $EC(C,Q)$ are normalised collection scores generated by the corresponding models from Section 6.3.1. In the lack of training material, we combine the two with equal weights, i.e., set $\lambda = 0.5$. Note that under the AENN approach the central broker only contains the names of entities.

6.4.2 Collection Selection

A good collection selection method balances between effectiveness and efficiency. That is, select as few servers as possible, to minimise communication costs and latency. On the other hand, it avoids being too restrictive, since only the selected collections can contribute to the final result set. The central ranking of entities (ER) plays a vital role in our collection selection mechanism; it may be viewed as a prediction of the final ranked list of results that we expect to see at the end of the merging step. However, it is to be decided how much confidence we wish to assign to this prediction. Next, we define three different selection strategies; one favours precision, another prefers recall, and the final one attempts to balance between the two.

Precision-oriented selection (AENN(p)) We rely on the entity-centric (EC) collection ranking and believe that it would yield the highest precision. Subsequently, we only select collections that EC contains. Unlike the following two methods, this selection strategy may decide to "skip" certain collections that otherwise have a high AENN score, but did not contribute any results to the top γ of the ER ranking. This strategy shares similarities with SUSHI [112] in the sense that it only selects collections that are expected to contribute results to the final merged list.

Recall-oriented selection (AENN(r)) The most conservative strategy is to fall back to the collection-centric (CC) collection ranking, as that has a higher recall than EC. We start at the top of the CC ranking and include collections for selection until we have all from the EC ranking covered. Formally, this method selects the top ρ collections from the AENN ranking, where

$$\rho = \arg\min_x \forall C \in EC : r_{CC}(C,Q) < x. \tag{6.4}$$

Balanced selection (AENN(b)) This strategy aims at balancing between precision and recall, by selecting the top collections based on the AENN ranking until all collections from EC are covered. Formally, we select the top ρ from AENN such that

$$\rho = \arg\min_x \forall C \in EC : r_{AENN}(C,Q) < x. \tag{6.5}$$

Figure 6.1 illustrates the three strategies on a toy-sized example.

6.4.3 Entity Representation

We apply a rather straight-forward entity model: every unique subject found in the collection is an entity. From all subject-predicate-object triples with the entity as subject, we concatenate the object text values into a *content* field. An entity's *name* is given by the object values of a predefined list of predicates (e.g., *foaf:name* or *rdfs:label*). This is similar in spirit to the Unstructured Entity Model introduced in Section 4.4.2.

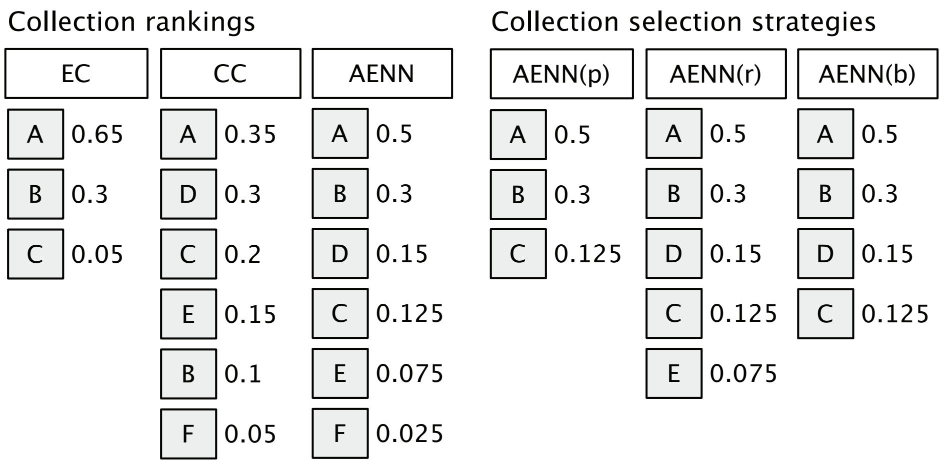


Figure 6.1: Illustration of collection selection strategies. The squared letters A,..,F represent collections, the numbers next to them are the collection ranking scores.

6.5 Experimental Evaluation

The main research question guiding us is as follows: How does the AENN method compare to traditional document-based methods on the collection ranking and collection selection tasks? We refer back to Section 5.5.2 for evaluation criteria we used, and present our results in the two subsequent sections.

6.5.1 Collection Ranking

Table 6.2 reports an overview of collection ranking results when disregarding prior information. The top two blocks show our baseline methods (CC and EC). All baseline methods are tested with two types of entity representations at the central broker (second column): name-only (N) and content (C). The last block presents our proposed method; recall that it always uses a name-only representation as the central broker in the AENN case maintains only the names of entities.

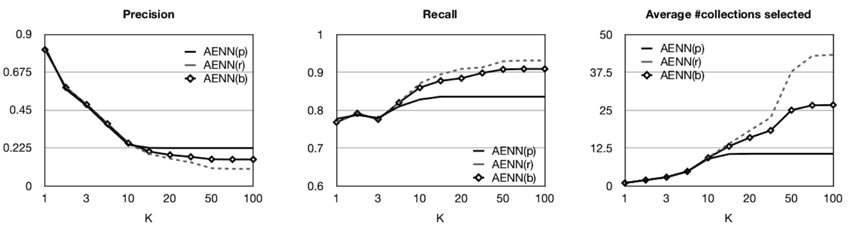
The first important observation from Table 6.2 is, that, as expected, the content-based representation provides better results than the name-only one. Apart from a few exceptions, this holds for all methods and collections, however, the difference is rather small for the *DBpedia* collection; this is because a name is available there for each entity. Third, our AENN method successfully combines the two collection ranking strategies. It outperforms all name-only representations for the BTC and BTC\DBpedia collections by 12% and 18% in terms of MAP, respectively. On the DBpedia collection the results are virtually the same as that of the EC run. The differences in MAP are significant for all collections. In sum, the overall performance of the AENN method makes it a viable alternative to other approaches

Table 6.1: Collection ranking results with collection priors. Significance is tested using a two-tailed paired t-test at the 0.01 level. $^{\dagger}/^{\ddagger}$denotes significant differences to the CC/EC rows, respectively.

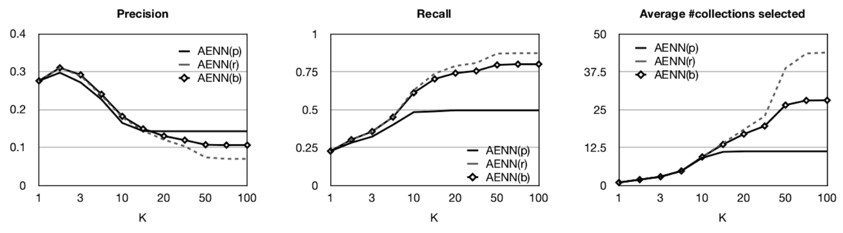
		BTC				**BTC\DBpedia**			
Meth.	**Rep.**	**MAP**	**MRR**	**P@10**	**NDCG**	**MAP**	**MRR**	**P@10**	**NDCG**
Lexicon-based methods									
CC	N	0.5276	0.8283	0.2213	0.7926	0.3046	0.4609	0.1521	0.4903
	C	0.5450	0.9001	0.1978	0.8475	0.2677	0.4227	0.1265	0.4643
Document-surrogate methods									
EC	N	0.6571	0.8997	0.2574	0.8555	0.4679	0.6008	0.1949	0.6224
	C	0.6339	0.9079	0.2360	0.8718	0.4189	0.6037	0.1735	0.5865
Our method									
AENN	N	$0.6223^{\dagger\ddagger}$	$0.8895^{\dagger}$	$0.2456^{\dagger\ddagger}$	$0.8457^{\dagger}$	$0.4409^{\dagger\ddagger}$	$0.5811^{\dagger}$	$0.1863^{\dagger}$	$0.6057^{\dagger\ddagger}$

Table 6.2: Collection ranking results without collection priors. Significance is tested using a two-tailed paired t-test at the 0.01 level. $^{\dagger}/^{\ddagger}$denotes significant differences to the CC/EC rows, respectively.

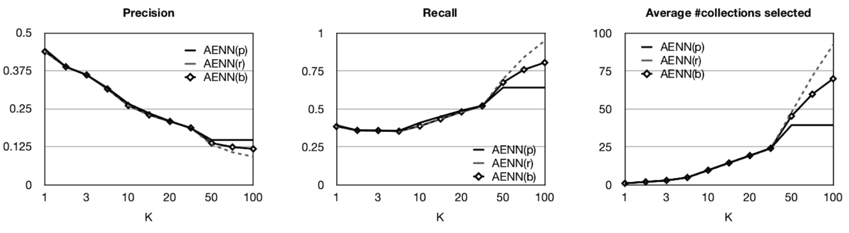
		BTC				**BTC\DBpedia**				**DBpedia**			
Meth.	**Rep.**	**MAP**	**MRR**	**P@10**	**NDCG**	**MAP**	**MRR**	**P@10**	**NDCG**	**MAP**	**MRR**	**P@10**	**NDCG**
Lexicon-based methods													
CC	N	0.2618	0.4496	0.1397	0.4447	0.2436	0.4001	0.1051	0.4283	0.1372	0.2234	0.1023	0.3982
	C	0.3018	0.5170	0.1699	0.5327	0.2490	0.4043	0.1128	0.4357	0.1391	0.2259	0.1069	0.4003
Document-surrogate methods													
EC	N	0.4753	0.6545	0.2272	0.5781	0.4720	0.6271	0.1863	0.6269	0.1323	0.2350	0.0908	0.3929
	C	0.5007	0.7013	0.2213	0.6370	0.4339	0.6143	0.1735	0.5947	0.1349	0.2071	0.1031	0.3929
Our method													
AENN	N	$0.4315^{\dagger\ddagger}$	$0.6048^{\dagger\ddagger}$	$0.2125^{\dagger\ddagger}$	$0.5572^{\dagger\ddagger}$	$0.4304^{\dagger\ddagger}$	$0.5743^{\dagger\ddagger}$	$0.1718^{\dagger\ddagger}$	$0.5918^{\dagger\ddagger}$	$0.1357^{\dagger\ddagger}$	0.2379	0.0915	0.3964



(a) BTC.



(b) BTC\DBpedia.



(c) DBpedia.

Figure 6.2: Comparison of AENN collection selection strategies.

that use a full content-based representation. Moreover, it has additional benefits for collection selection, as we shall see next.

We observer similar results when collection priors are used, as shown in Table 6.1 (results for the *DBpedia* setting are omitted in this case). The numbers for *BTC* are arguably much higher; again, this is caused by the strong impact of entities from *DBpedia*.

6.5.2 Collection Selection

First, we compare the three AENN collection selection strategies we devised against each other.

We show results for the BTC collection shown in Figure 6.2(a). We find that the three methods indeed work as they were originally intended: AENN(p) results in the highest precision and on average it never selects more than 11 collections. AENN(r), on the other hand, can select up to 44 collections; this leads to a high recall, especially for high K values. AENN(b) seems to be able to find the golden middle between the two; it performs well both in terms of precision and recall, while it keeps the number of selected collections reasonably low (28 at most). Both precision and recall are at high levels here, which can be attributed to the high fraction of DBpedia documents in the collection. This leads to DBpEDIA being both the is that our most important collection in terms of relevant entities it contains and the easiest to select.

The plots for BTC\DBpedia are depicted in Figure 6.2(b). Overall, the results are very similar, albeit at lower levels due to the fact that the DBpedia collection is missing now.

The DBpedia set, for which we show results in 6.2(c), selects a higher number of collections than we do on the other sets (up to 90 in the case of AENN(r). This is expected though because of the random distribution of document over collections. Precision and recall, on the other hand show similar behaviour as in BTC and BTC\DBpedia.

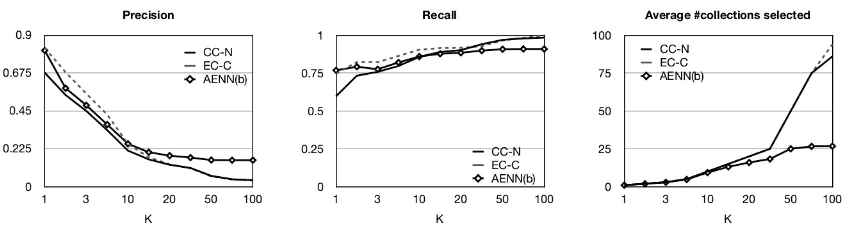
Next, we compare the AENN against two baselines, both using top-K selection with a fixed K value: (1) collection-centric using a name-only representation (CC-N), and (2) entity-centric using a content-based representation (EC-C). The latter serves as an upper limit that could be achieved if the broker had a local copy of the full contents of all collections. These results are shown in Figure 6.3.

Overall, we note that we observe similar behaviour for all three collections. The number of collections selected is very low on both BTC and BTC\DBpedia, as shown in 6.3(a) and 6.3(b). On BTC\DBpedia, where names are missing for many entities, AENN(b) is closer to CC-N than to EC-C, but it always outperforms the former, enonetheless.

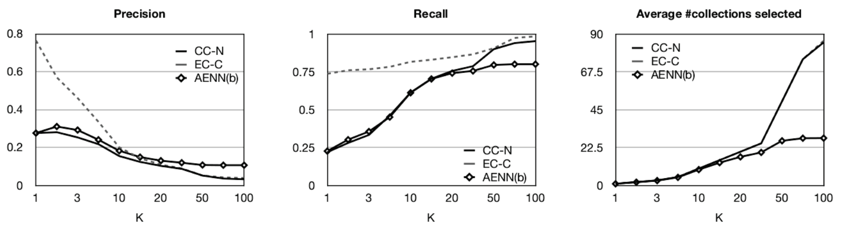
Figure 6.3(c) reports the results on the DBpedia collection. Again, this collection arguably provides the most difficult setting. We find that the balanced variant of the AENN method comes very close to the “oracle” run of EC-C, both on precision and on recall. It can also reduce the average number of selected collections, but only for high K values. This is due to the random distribution of relevant documents, a special characteristic of this setup.

6.6 Conclusions

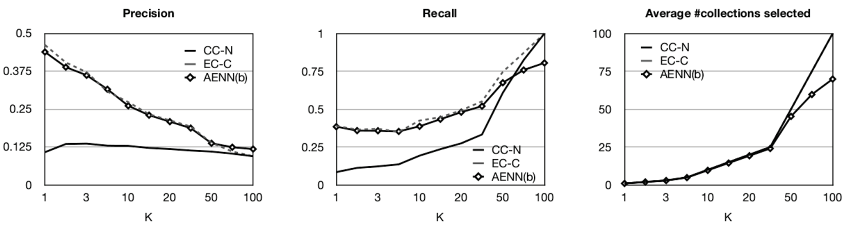
In this chapter, we investigated the feasibility of a federated search architecture for entity retrieval and studied two sub-problems in detail: collection ranking and collection selection. We made an argument that for queries that target a particular entity, which is very frequent in Web search, traditional document-based distributed



(a) BTC.



(b) BTC\DBpedia.



(c) DBpedia.

Figure 6.3: Comparison of baseline and the AENN(b) collection selection strategies.

retrieval techniques might not be the best choice. We proposed a novel method, AENN, that builds on the observation that for such queries the central broker could maintain a complete dictionary of entity names, instead of sampling full representations from each collection. This lean representation can then be utilised for collection selection and can also be used to gear results towards high precision or high recall. Further, we created three test collections based on WoD collections and performed an experimental evaluation using these. Our method has shown great promise as it performed just as good as the idealised setting for some collections, in terms of precision and recall, while selecting fewer collections.

As for future work, we believe there is further mileage to be gained by improving the name-based ranking of the central broker. We also wish to cover other aspects of distributed entity search such as efficiency aspects and various practical considerations (e.g., caching). Another open question is in how far sampling of entity names can help the collection ranking process, i.e., how applicable is sampling when only a very straightforward entity representation (its name) is used.

Chapter 7

Peer-to-peer (P2P) Networks

After having investigated broker-based architectures in Chapters 5 and 6, we now make the transition to a peer-to-peer (P2P) environment. This is motivated by the increasing growth of the WoD. Further, research in federated search considers typically around 100 collections. This is quite a low number when thinking of the constant growth both in size and number of the Web of Data. We therefore investigate the applicability of P2P technologies for search in the WoD, anticipating its continuing growth.

7.1 Introduction

Modern applications are increasingly deployed over widely distributed data sources and each of them stores vast amounts of data, a development partly driven by the growth of the web itself. Web information retrieval settings are a good example for such architectures, as they contain large document collections stored at disparate locations. Central assembly of the total information is neither feasible, as digital rights do not allow replication of documents, nor effective, since the cost of storing and maintaining this information is excessive.

It is crucial to aggregate information throughout the network in an efficient manner, i.e., compute global statistics based on local information shared by the individual peers. We provide a thorough investigation of this problem in this chapter. The techniques introduced are targeted at document frequency estimation in P2P networks and in the course of this chapter, we present an efficient hybrid approach for aggregation of document frequencies using a hierarchical overlay network for a carefully selected set of the most important terms, together with gossip-based aggregation for the remaining terms in the collections. We conduct experiments on three document collections, in order to evaluate the quality of the proposed hybrid aggregation.

One of the main problems in distributed retrieval lies in the difficulty of providing a qualitative ranking of documents with respect to user queries. In this context, the baseline or reference is the centralised case. At the same time, performance and scalability considerations play a vital role in the development and applicability of such a widely distributed system. Thus, the important problem in the context of unstructured P2P networks is to provide a comprehensive ranking of terms (and documents). Information about how many documents a term appears in (the so called document frequency of a given term) is vital for the task of searching documents and ranking the results of these searches according to popular information retrieval ranking models. As such, document frequency estimation is one of the main components of a search system and it is particularly difficult in the distributed context. Clearly, exchanging all terms and their respective document frequencies would be a solution, however the cost is prohibitive, even for modest network sizes and medium-sized document collections, and even more so for dynamically evolving collections. Therefore, we need a pre-selection of terms at the peer level to evaluate the usefulness of terms locally. The more flexible an approach is in handling held back terms, the more stable it is with respect to cheating or withholding of information by single peers. This aggregation process must work well without consuming excessive bandwidth, regardless of the size of the network topology.

In general, two alternatives exist for performing aggregation in unstructured P2P networks: 1) building a hierarchical overlay network that enables *hierarchical aggregation*, and 2) adopting a *gossip-based aggregation* protocol. Each approach has its own merits and shortcomings. Hierarchical aggregation is efficient, fast and results in accurate values. Gossip-based aggregation is simple, scalable and robust to peer failures. However, it only provides probabilistic guarantees for the accuracy of aggregation and induces higher communication cost. Motivated by this discussion, we propose a hybrid approach for aggregation of term frequencies that combines hierarchical and gossip-based aggregation, thus sharing their advantages.

The hierarchical overlay network is formed in a self-organising manner, which enables efficient aggregation of information. Then, carefully selected terms and their corresponding frequencies from each peer are pushed upwards in the hierarchy. A gossip-based aggregation protocol is employed to estimate the document frequency of less frequent terms by local periodic communication. In order to obtain the final results, the estimates resulting from the two approaches are merged and can be used for ranking documents or other information retrieval tasks.

For the hierarchical aggregation only a relatively small number of terms from a prohibitively large overall set of terms are selected. The remaining low-frequency terms are aggregated using gossiping. An interesting related issue is how to perform this term selection at individual peer level independently of other peers' contents, and we investigate the impact of term selection techniques in this context. Hence, the main contributions of this work are:

1. We present an approach for hierarchical aggregation that can be used to

estimate with high accuracy the document frequencies of carefully selected terms, without assembling all information at a central location.

2. We complement the hierarchical aggregation approach with gossip-based aggregation, in order to estimate the document frequency of the remaining terms.

3. We conduct an experimental evaluation on three document collections demonstrating the applicability and scalability of the approach, and we investigate the accuracy of the aggregated information.

4. We show how this aggregated information can improve results obtained by using language models.

The work presented in this chapter is based on techniques and experiments introduced in [76]. In this chapter we also present additional techniques to handle low-frequency terms as well as the hybrid method for frequency estimation. Further, we initially estimated information important for TFIDF weighting. However, this approach can easily be adopted to estimate the average document length and collection frequencies necessary for the language modelling approach. As such, this chapter is a possible extension of our entity search approach to P2P networks. The main differences between the P2P and federated search scenarios are outlined in Section 2.4.2. We further perform new experiments including a new quality measure and retrieval experiments with an additional large-scale collection.

The remainder of this chapter is structured as follows: in Section 7.2, we provide an overview of relevant work done in related areas. We then introduce preliminaries such as basic term selection approaches and aggregation in P2P networks in Section 7.3. The details of the hybrid estimation approach and the underlying hierarchical aggregation together with gossip-based elements are described in Section 7.4. We briefly talk about a cost model for assessing the communication cost in Section 7.5. The experimental setup as well as evaluation in terms of document retrieval and ranking are presented in Section 7.6. We show how our estimation techniques can be applied in the context of entity search in Sectin 7.7. Finally, in Section 7.8, we draw conclusions and give an outlook on future work.

7.2 Related Work

In this section, we provide an overview of research efforts that are deemed relevant to this chapter. First, we refer to the basics of federated search which are introduced in Sections 2.4.1 where we also provide a brief overview of related work in federated search. We then present the most prominent approaches on aggregation in large-scale distributed systems.

In this chapter, we implicitly study the effects of different term pre-selection methods on distributed document collections over an unstructured P2P network, even though its dynamic aspects are not the main concern in DIR research. Also, our experiments are specifically designed to show the effects of unequally distributed collections, which is a common case in DIR settings.

Aggregation of information in distributed systems is a challenging issue, thus it has attracted the attention of several research initiatives. One obvious method is to employ a hierarchy of nodes to perform hierarchical aggregation at intermediate levels. SDIMS [123] uses tree-based aggregation over a Distributed Hash Table (DHT) infrastructure, in order to provide a generic aggregation mechanism for large-scale systems. For other tree-based aggregation efforts we refer to Willow [115] and SOMO [127].

Gossip-based aggregation [49, 52] aims to provide a protocol for network-based aggregation in a completely decentralised manner. The actual process of aggregation is achieved through periodic interactions (also known as cycles) among nodes, which exchange aggregated values. Gossiping protocols for information aggregation also provide theoretical properties for convergence within a logarithmic number of steps with respect to the network size.

There also exist other systems that combine gossip-based aggregation with hierarchical aggregation. Probably the most well-known framework in this category is Astrolabe [116], which provides a continuously running aggregation mechanism. The purpose of aggregation is to compute aggregate values of a particular resource of interest (such as number of copies of a specific file) in a network-wide context. While Astrolabe focuses on maintaining aggregate values at any time, in [44], hierarchical gossiping is proposed to handle one-shot evaluation of aggregate queries.

To the best of our knowledge, none of the existing systems for aggregation has been applied in the context of information retrieval, where the underlying information that needs to be aggregated refers to terms and their respective frequency values in autonomous document repositories. As a result, an open issue that remains is to what extent a distributed and scalable aggregation mechanism for term frequency values can produce retrieval results of good quality. This is one of the topics covered in this chapter.

7.3 Preliminaries

Due to the resultant high number of terms found in text documents and the subsequent high dimensionality of the term vectors, the selection of a subset of terms to use for analysis and search is essential. Especially in the areas of machine learning and data mining it is a vital task to find a set of terms that both adequately represents the collection in question and filters out enough terms so that processing is computationally possible. We propose to use some of the existing techniques of filtering out less important terms on the peer level as a first filtering step before the estimation process. To this end, we refer back to Section 2.4.2 where we briefly provided the necessary background on the most prominent term selection techniques which can be integrated in our framework. Subsequently, we present an

Table 7.1: Overview of variables.

Variable	**Description**
df_t	Document frequency of term t
CF_t	Collection frequency of term t
h	Height of DESENT hierarchy
t_j	Term
t_{size}	Size of term/frequency tuple
$tf_{t,d}$	Frequency of term t in document d
N_P	Number of peers
$N_{l,i}$	Number of documents at peer P_i
P	Peer
S_Z	Number of peers in DESENT zone
T	Number of terms contributed from peer in aggregation process
TV_i	Term vector of document i

overview of aggregation in P2P networks, focusing mainly on a) hierarchical and b) gossip-based aggregation. An overview of variables and symbols used throughout the remainder of this chapter is given in Table 7.1.

Challenges and Objective

The high degree of distribution of documents in a P2P system with autonomous peers makes effective term selection particularly challenging. The reason is that when term selection is applied on a subset of the complete document collection, which resides on a peer, the terms that are deemed important may be less important when the entire collection is considered. Therefore, identifying globally important terms necessitates aggregation of local term frequency values, so that the aggregated result reflects the contents of the entire document collection. In our distributed context, it is not straightforward to choose an effective term selection method, as this is highly dependent on the degree of distribution and the representativeness of local document collections with respect to the global collection.

The main objective of this work is to identify appropriate term selection and aggregation techniques for application in large-scale P2P networks. We seek methods that produce aggregated results of comparable quality to the centralised case, where all documents would be available on a single peer.

7.3.1 Aggregation in P2P Networks

Aggregation is an important task for deploying useful applications in large-scale distributed systems. In the context of unstructured P2P networks, there exist two main approaches for aggregation of information: *hierarchical aggregation* and *gossip-based aggregation*.

Table 7.2: Comparison of aggregation methods.

Criterion	**Hierarchical**	**Gossip-based**
Correctness of aggregation	High	Probabilistic
Error recovery mechanism	Complex protocol	Simple protocol
Stability under stress	Unstable/prone to failure	Stable
Load balancing	Imposed on few peers	Fair load assignment
Computation cost	Reduced	Relatively high
Maintenance cost	Hierarchy maintenance	None
Communication costs	Low	High
Scalability	Aggregation at all levels	Local interactions only

In hierarchical aggregation, an often dynamic hierarchy is formed and information is aggregated at intermediate levels, before being propagated upwards. As a result, the aggregation process: a) is efficient due to the reduction of the communication cost, b) scalable, as the aggregation load is distributed to several peers, and c) is accurate, since the information that reaches the root accurately reflects the real aggregated value (in accordance with term ranking as shown in later experiments). On the other hand, the hierarchy induces additional maintenance cost, usually requires a complicated protocol that ensures fault-tolerance, and is unstable when the churn rate is high.

In gossip-based aggregation, peers constantly exchange information in cycles, by selecting a small random subset of other peers at each cycle. There exist theoretical guarantees that show that aggregated values converge exponentially fast to the true aggregates [49]. Gossiping is based on a simple and scalable protocol that is stable under stress because only local interactions between peers take place. Moreover, the assignment of load to all peers is fair, without any additional maintenance cost. On the downside, the guarantees for the aggregated values are probabilistic in nature and the cost for computing the aggregates is higher than in the case of hierarchical aggregation.

For comparative purposes, we provide a summary of the benefits and drawbacks of each approach in Table 7.2.

7.4 Hybrid Aggregation of Document Frequencies

In this section, we describe our approach for aggregating terms and their document frequencies without central assembly of all data. We employ an unstructured P2P architecture and the overall aim is to provide estimates of frequency values that are as similar as possible to the centralised case.

As described in the previous section, both hierarchical and gossip-based aggregation have their advantages and disadvantages. A natural question that arises is to what extent the former aggregation approaches can be combined, in order to overcome their limitations on an individual basis. This is the motivation to introduce a *hybrid* approach for aggregation of document frequency values in widely distributed unstructured P2P networks. For terms with high local frequency values, we employ hierarchical aggregation, thus computing more accurate aggregate values. Intuitively, terms with high frequency of occurrence are considered more important and have a larger impact on similarity ranking, therefore the aim is to compute their aggregate value with higher accuracy and in shorter time. The focus is then put on the remaining terms only after the dissemination of the hierarchical aggregation information. Subsequently, terms with low frequency value on local basis are aggregated using gossip-based aggregation. The aggregate frequencies of such terms will not be completely accurate, but they are approximated closely with probabilistic guarantees.

We first provide an overview of the DESENT architecture, which is used as the underlying hierarchical overlay network, and we describe how hierarchical aggregation is realised in this framework (Section 7.4.1). Then, we describe the second part of our hybrid aggregation, namely the gossip-based aggregation protocol employed (Section 7.4.2).

7.4.1 Hierarchical Aggregation

To the ends of creating a hierarchical overlay network over a purely unstructured (Gnutella-like) P2P network, no matter its network distance, we employ a variant of DESENT [38]. The reasons for this choice are the completely *distributed* and *decentralised* creation of the hierarchy, its low creation cost and robustness. The most important details of the basic algorithm are described in the following; for more in-depth explanations we refer to [38]. The DESENT hierarchy can be used for building overlays for searching, but also for other purposes like aggregation of data or statistics about contents from participating peers—which is the way in which DESENT is utilised in this chapter.

DESENT

For an illustrative example of the DESENT hierarchy, see Figure 7.1. The bottom level consists of the individual peers ($P_{A_1} \dots P_{A_n}$ and $P_{B_1} \dots P_{B_n}$). Then neighboring peers (network-wise) create *zones* of approximate size S_Z peers (i.e., groups of peers) around an *initiator* peer (P_A and P_B), which acts as a zone controller. Notice that the height (h) of the hierarchy equals to: $log_{S_Z} N_P$. These level 1 initiators (P_A and P_B) are mostly uniformly distributed over the network, and are selected independently of each other in a pseudo-random way. The initiators form the next level of the hierarchy, they are responsible for the peers in their zones, and they aggregate the information collected from their peers.

In the subsequent phases, super-zones are created, which consist of a number of neighboring zones from the previous level. Each super-zone is represented by a

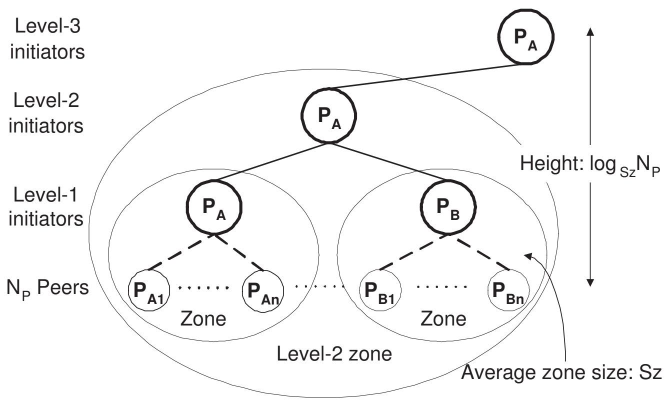


Figure 7.1: Example of a P2P hierarchy of height h=3 with peers and zones.

super-zone initiator that is responsible for the initiators in its zone and aggregates the information of these initiators. The zone initiators essentially form a P2P network similar to the original P2P network, and the aforementioned process is repeated recursively, using the zone initiators as peers. In the example of Figure 7.1, P_A is initiator both at level 2 and level 3. In this way, a hierarchy of initiators is created, with each initiator collecting aggregate information that refers to the contents of all peers in the tree rooted at that initiator. Finally, at the top-level initiator, aggregate information that spans the contents of the entire network is available.

Aggregation Process

The process of estimating the *frequency of selected terms* based on DESENT can be summarised as follows:

1. A tree-based P2P structure is created using the DESENT protocol [37, 38].
2. All peers select up to T terms from their local document collection using one of the techniques described in Section 2.4.2, and send these terms together with the total number of documents to the parent peer in the tree.
3. Each parent peer receives up to $S_Z T$ terms with respective document frequencies, where S_Z denotes the average number of peers in a zone. The parent peer selects up to T terms, these terms are propagated upwards together with the aggregated document frequencies and the total number of documents in the subtree rooted at the peer.
4. The process continues up to the level of the children of the root (i.e., peers at level $h-1$), where h denotes the height of the tree. Level 0 is the bottom level and level h is the level of the root peer. Instead of performing the last aggregation at the root peer, it is performed by the children of the root. This

is achieved by first distributing their aggregated values by hashing to the other root-children peers.

5. The estimated document frequency values and the total number of documents are disseminated to the participating peers.
6. The whole process is repeated at regular intervals, in order to capture changes in document contents, as well as improving the estimated values. An alternative to fixed-time intervals would be to employ heuristics to assess the fluctuation in the network, i.e., initiate the process once a given number of peers joins or leaves the network.

We will now describe in more detail the local term selection, document frequency calculation, aggregation, and the final dissemination of information to the peers.

Local Term Selection and Document Frequency Calculation

Each peer P_i selects up to T terms from the $N_{l,i}$ locally stored documents, using one of the unsupervised term selection techniques described in Section 2.4.2. Term selection at a peer is based on the peer's local knowledge only. Thus, the result of the term selection is a *term vector* TV_i, which is the number $N_{l,i}$ and vector of term tuples. Each term tuple in TV_i contains a term t_j and the local document frequency df_{t_j}: $TV_i = [N_{l,i}, [(t_1, df_{t_1}), ..., (t_T, df_{t_T})]]$.

Level-wise Aggregation

After the $S_Z T$ selected terms from the previous phase have been received, a new term vector is created of the received terms and their frequencies, this vector looks as follows, $TV_j = [N_s, [(t_1, df_{t_1}), ..., (t_{S_Z T}, df_{t_{S_Z T}})]]$. N_s is the sum of the received local frequencies, i.e., $N_s = \sum_{i=1}^{S_Z} N_{l,i}$. Furthermore, duplicate terms and their frequencies (i.e., the same term originating from several peers) are aggregated into one tuple. Subsequently the number of terms in the new term vector is less than $S_Z T$. Finally, the term vector is reduced to only contain T terms. Term selection is performed based on the frequency of appearance, therefore terms that have high frequency are favored. The intuition, which is also confirmed by related work in [120], is that it is important to identify terms that are globally frequent and forward such terms to the top of the hierarchy. The generated term vector after aggregation and term selection, again consisting of T terms, is sent to the next level in the tree and this process continues iteratively up to level $h - 1$, i.e., the children of the root.

Hash-based Distribution and Aggregation

Performing the final aggregation at the root peer is a straightforward process, however it makes the system vulnerable, as it induces a single point of failure.

Instead, the final aggregation is performed by the children of the root, at level $h-1$. Notice that in this phase, our approach trades efficiency for robustness. We employ a more costly way to aggregate information, however the overall system becomes fault-tolerant. The actual aggregation is achieved by having the level $h-1$ peers first distributing their aggregated values, by hashing, to the other level $h-1$ peers. A recipient peer becomes responsible for a different subset of terms and aggregates their frequencies, thus performing (part of) the task that the root peer would perform. After the aggregation of the received term vectors, the peers send all their aggregated results to the rest of the P2P network. In the end, all level $h-1$ peers have the complete aggregated values locally available.

The reason for hashing is two-fold. First, it is important that all statistics for one particular term end up at the same node, in order to provide aggregated values per term. Second, the workload of the final aggregation is distributed and shared among the level $h-1$ peers, thus achieving load-balancing.

Dissemination of Information

In the final phase, the aggregated term vectors are distributed to all participating peers. This is performed by using the hierarchy as a broadcast tree. The term vectors are sent downwards, until they reach the level-0 peers. The size of the disseminated information is equal to the number of term vectors ($S_Z T$) multiplied by the number of level $h-1$ peers. The aggregated terms and document frequencies are now available at all peers locally. As a consequence, any peer can use this information, in order to provide rankings of terms and documents taking into account the global document collection. In the experimental section, we study the accuracy of relevant ranking between pairs of terms to demonstrate the effectiveness of our approach.

7.4.2 Gossip-based Aggregation

After having elaborated in detail how hierarchical aggregation is performed, we proceed to describe the gossip-based aggregation mechanism. Local low-frequency terms are selected for gossip-based aggregation. In practice, this selection is done by selecting the terms for which no estimation is available from the hierarchical aggregation. The rationale is twofold. First, it suffices that such aggregates are computed with probabilistic guarantees only, without the strict requirement of accurate computation. Second, aggregates of low-frequency terms can be computed with some delay, therefore gossiping can be employed leading to eventual consistency of aggregate values. In contrast, high frequency terms need to be aggregated in a more timely fashion. Thus hierarchical aggregation is a more appropriate method because they have more significant impact on search results.

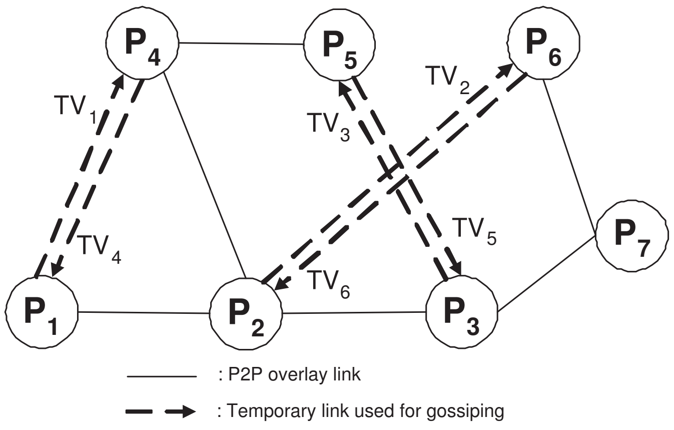


Figure 7.2: Gossip-based exchange of information between peers.

Algorithm 1 Gossip-based aggregation at peer P_i.

```
1: while (true) do
2:     $P_j \leftarrow$ GetRandomPeer()
3:     Send($TV_i$, $P_j$)
4:     $TV_j \leftarrow$ Receive($P_j$)
5:     aggregate($TV_i$,$TV_j$)
6: end while
```

Algorithm

The basic underlying gossiping protocol works in the following way. Any peer periodically exchanges information with another randomly selected peer. In principle, more than one peers can be selected for information exchange, however, for simplicity we assume that only one peer is selected. Each round of communication is also known as *cycle*. During a cycle, a peer exchanges its locally maintained state with that of another peer. The local state of a peer can be any value of interest that needs to be aggregated, and in the simplest case it is a plain number representing, e.g., the load of each peer.

In our setup, gossiping is used to aggregate term frequency values. Therefore, the basic intuition of gossip-based aggregation remains, only the amount of information that is exchanged between any two peers changes. Whenever two peers P_i and P_j engage in communication, their term vectors TV_i and TV_j are exchanged to perform aggregation of term frequency values. The term vectors contain information about the terms occurring on the respective peers and in how many documents they occur in following the definition given in Section 7.4. This is illustrated graphically in Figure 7.2, where the peer interactions at one random gossiping cycle are shown.

Algorithm 1 provides the pseudocode for gossip-based aggregation on peer P_i. In each gossiping cycle, P_i chooses a random peer P_j (line 2) and establishes a direct link through the P2P overlay network. Then, P_i sends its term vector TV_i to P_j (line 3), and receives from P_j its term vector TV_j respectively (line 4). Then, P_i aggregates the values of the term vectors (line 5). The aggregate function can be for example the average function, so for each pair of identical terms $t_i = t_j$, their average frequency value is computed ($\frac{d_i+d_j}{2}$) and it replaces the current state (frequency) of t_i in TV_i. In the case of terms that exist on only one of the peers (e.g., P_i), they are appended to the term vector of the other peer (e.g., TV_j) and the frequency value is divided by two (e.g., $\frac{d_i+0}{2}$). Notice that at any point during the gossiping protocol, the sum of frequency values can be easily computed by multiplying the average frequency values computed thus far with the number of peers N_P in the network. An interesting aspect of the gossip-based aggregation protocol is that it can be also used to estimate the value of N_P, in case this knowledge is not available to the peers. For details on computing this type of aggregates (counting the number of peers) we refer to [49].

7.4.3 Combination of Hierarchical Estimation and Gossiping

The results of the hierarchical estimation is used for document frequency estimations. Terms for which there exist no hierarchical estimates (low-frequency terms) are straight-forwardly assumed to have a document frequency of one. These will subsequently be combined with the results from the gossip-based aggregation in that the document frequencies of these terms are updated to the estimated values.

7.5 Cost Analysis

The cost for the creation of the DESENT hierarchy is described in [38]. Further, we employed a straightforward cost analysis model to assess the bandwith consumption of our aggregation approach. Details are not included in this thesis, but can—together with experimental results—be obtained from [77].

7.6 Experimental Evaluation

We conducted experiments using three different document collections. Since we put a special focus on large-scale P2P settings, we tried to use corpora of sufficient size. In fact, two out of these three corpora contain nearly 500,000 documents, and one is smaller with about 20,000 documents. Table 7.3 gives an overview of the sizes of the collections used. We list the number of documents, the disk space the collections use in uncompressed form, as well as the vocabulary size of the collection, i.e., the total number of unique terms in the collections, and the average number of terms per document, i.e., the average document length. All of this information is given for the preprocessed and indexed collection.

Table 7.3: Benchmark collections used in our experiments. We list the collections' names, the number of documents, the amount of disk space they use in uncompressed form, the vocabulary size of the collection, i.e., the number of distinct terms, and the average document length.

Name	**#Docs**	**Disk**	**Voc. size**	**Avg. doc length**
20 newsgroups	18.828	85M	94.753	97
DMOZ	484.113	2,9G	1.732.228	340
TREC8	528.155	2,8G	842.682	247

Data collections. We worked with the 20 newsgroups data set[1] which has become very popular for text experiments in the field of machine learning and has been used for example in [74]. The data set consists of newsgroup postings from 20 newsgroups. From each newsgroup, 1,000 articles posted in the year 1993 have been selected; after removing duplicate articles (mostly cross-postings to several newsgroups), 18.828 unique messages remain.

The DMOZ collection is a collection of 483,000 web pages, which are classified by the DMOZ taxonomy.[2] The collection has been created by retrieving the web pages that are linked from the leaf-classes of the DMOZ taxonomy. The fact that the collection has been automatically retrieved from the web leads to a high number of terms in this collection. Generally, the data used here is more noisy and less focused than in the other collections, which surely has disadvantages, but definitely shows the applicability of our techniques to a general real-life web setting. The taxonomy path to a page is considered to be the class/category of the page. It is the only test collection used in this chapter which is not publicly available.

Further, we used one of the collections included in the TREC information retrieval benchmarking initiative.[3] More specifically, we used the collections used in the TREC8 ad hoc evaluation. The ad hoc task is one of the traditional tasks in TREC evaluations and comprises both a collection of 500,000 to 700,000 text documents such as news messages and queries for that collection. The TREC8 collection consists of about 530,000 documents and 50 queries plus relevance judgments for them. The TREC8 collection comprises material from the Foreign Broadcast Information Service, the Los Angeles Times (randomly selected articles from 1989 & 1990), the Federal Register (1994), and Financial Times articles (1992-1994).[4]

All three test collections were preprocessed in terms of tokenisation, stop word removal and stemming for the English language.

[1] `http://people.csail.mit.edu/jrennie/20Newsgroups`

[2] `http://www.dmoz.org`

[3] `http://trec.nist.gov`

[4] The TREC8 ad hoc collection consists of documents from four different collections. In this context we used the information about which document belongs to which collection only in the distribution based on similarity. There we assume documents coming from the same sub-corpus to be similar.

Table 7.4: Varying distribution skew and similarity values used in experimental setups.

Id	Distribution Skew	Document Similarity within Peers
1	low	high
2	low	low
3	high	high
4	high	low

7.6.1 Experimental Setup

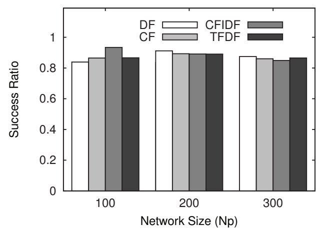
We identify the following basic parameters for our experiments and study their effect. First, the *number of partitions* or peers, as it affects the scalability of our approach. Then, the *distribution skew*, defined as the size distribution across the local partitions. A low distribution skew denotes equal amounts of documents per partition. Last, we consider the *document similarity*, defined as the degree to which documents in one partition are similar to each other. This simulates cases such as topically homogeneous collections (with a high degree of similarity) or cases of randomly distributed collections. To this end, we use class labels of documents and distribute documents to partitions already containing similar documents with a higher or lower probability according to the setting. In the case where no labels are available, document clustering is used instead to determine a measure of similarity.

In our experimental evaluation we use varying setups, in order to simulate different use cases. We vary the number of peers to study the scalability of our approach. For each given number of peers, we apply four settings: 1) low similarity, high distribution, 2) low similarity, low distribution, 3) high similarity, high distribution, and 4) high similarity, low distribution. To be able to show the impact of all extreme values of both parameters, we also included mixed setups and also the case of documents which are distributed in equal sized partitions and have no similarity relation to each other at the other end of the spectrum. We apply the aforementioned term selection methods at the local peer level to study their effect on the hybrid aggregation we propose. An overview of these four setups is given in Table 7.4.

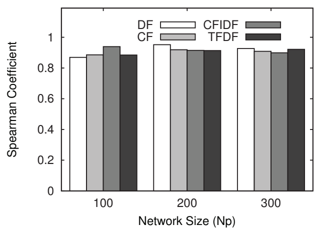
7.6.2 Evaluation of Term Ranking Quality

Our first aim is to study the quality of the aggregated document frequencies in terms of ranking. For this purpose, we use the 20 newsgroups and the DMOZ document collections.

Evaluation metrics. We define as *success ratio* the percentage of pairs of terms that have the same relative ranking in our approach and in the centralised case. In other words, for any two terms t_i and t_j the success ratio is the fraction of



(a) Success ratio.



(b) Spearman coefficient.

Figure 7.3: Scalability with network size for the 20 newsgroups collection.

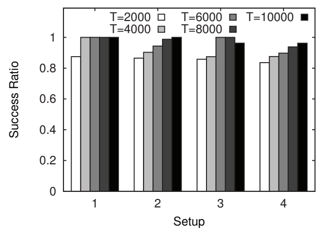
the number of such pairs with the same ranking with respect to the centralised ranking, over all possible combinations of pairs of terms. We chose this performance measure for existing standard approaches such as the Spearman or Kendall tau rank order correlation coefficients lack the support for rankings of different lengths, our approach, however, is closely related and basically extends these methods in its ability to handle different lengths of involved rankings.

We additionally provide results for measuring the Spearman coefficient between the two rankings. To this end, we first need to remove elements not available in both rankings. This means that all terms for which no estimated value is available are filtered out. The resultant equally sized rankings can then be compared using the *Spearman coefficient*, a measure for correlation between two rankings, operating on the rank on their elements rather than their numeric values.

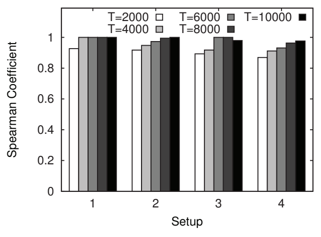
Results for the 20 newsgroups Collection

Figure 7.3 shows the result of our scalability study with respect to the network size. For this purpose, we increase the size of the network from 100 to 300 peers. The following values are used for the experimental parameters: number of terms for hierarchical aggregation T=2,000, number of gossiping cycles N_C=20, and zone size S_Z=10 (for N_P=100) and S_Z=20 (for N_P={200,300}). We use as basic setup the one with id=1, and we evaluate the following term selection methods: DF, CF, CFIDF, and TFDF.

In Figure 7.3(a), we depict the values of success ratio. Regardless of term selection method, our hybrid aggregation works effectively, achieving values higher than 80% and often close to 90%. This indicates that hybrid aggregation results in high quality term rankings that are similar to the centralised case. This is verified in Figure 7.3(b), when the Spearman coefficient is employed. In fact, the values of Spearman coefficient are higher than those of success ratio, again indicating the high quality aggregation of the proposed hybrid method. Another important observation is that the quality of term rankings does not deteriorate with increased network size, even when the number of peers is increased by a factor of three.



(a) Success ratio.



(b) Spearman coefficient.

Figure 7.4: Effect of increasing values of T for the 20 newsgroups collection.

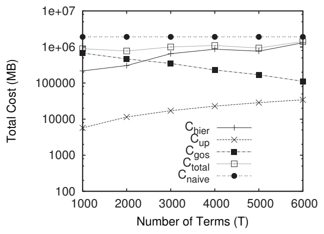


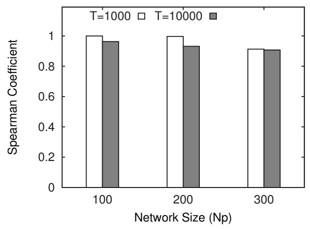
Figure 7.5: Experimentally derived cost for the 20 newsgroups collection and different numbers of aggregated terms.

We also study the effect of increasing values of terms T that are aggregated using hierarchical aggregation in Figure 7.4. Intuitively, as T increases, we expect that the quality of term ranking will improve, since more terms are aggregated hierarchically, thus resulting in more terms having accurate estimates of frequency values. Indeed, in Figure 7.4(a), we see that the values of success ratio metric increase with T irrespective of the setup. The same conclusions are drawn from Figure 7.4(b), where the values of the Spearman coefficient are shown.

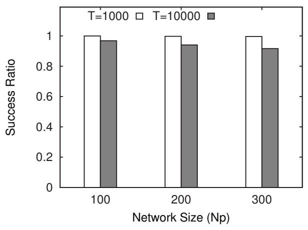
Results for the DMOZ Collection

We also performed a series of experiments using the DMOZ collection, in order to study the effect of larger corpora on our hybrid aggregation method. Document collections such as DMOZ are quite challenging, because of the increased vocabulary size and the noise present in the contents.

In Figure 7.6, we show the term ranking quality using the DF term selection technique. Our experimental parameters are S_Z=10, N_C=10, N_P=\{100,200,300\}, T=\{1,000;10,000\}. In order to limit the effect of noise, we additionally use a threshold on each peer that eliminates single-occurring terms on a peer from aggregation. We observe that high quality results (always higher than 90% and often close to 100%) are obtained for both our metrics: Spearman coefficient and success ratio.

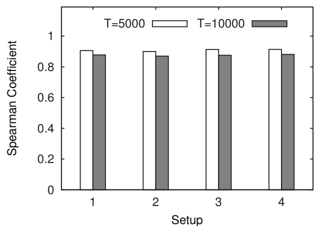


(a) Spearman coefficient.

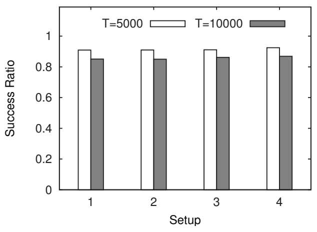


(b) Success ratio.

Figure 7.6: Term ranking quality for DF for the DMOZ collection.



(a) Spearman coefficient.



(b) Success ratio.

Figure 7.7: Term ranking quality for DF and N_P=1,000 for the DMOZ collection.

When the network size increases, we observe a small decrease in the values of our quality metrics. However, the absolute values are still higher than 90%.

Another interesting observation is that increasing values of T cause a small reduction in the values of Spearman coefficient and success ratio. We believe that this is because in the DMOZ collection the increased values of T lead to many low-frequency (noisy) terms being aggregated. Even a small value of T, such as 1,000, is sufficient to aggregate the important terms. In addition, the randomness of the hierarchical aggregation makes the selection of terms (that are finally aggregated hierarchically) random as well. This is the reason that increased values of T reduce the quality of ranking.

We also performed an experiment with a large network of N_P=1,000 peers to study the scalability of hybrid aggregation. The results are depicted in Figure 7.7. Again, the same conclusions are drawn verifying the performance of hybrid aggregation, even in the case of large-scale P2P networks.

Results for the TREC8 Collection

The fact that queries plus relevance judgments are available leads to several questions that can be answered in the context of P2P document frequency aggregation:

- Does the DESENT aggregation negatively influence retrieval results compared to the full information about document frequencies?
- What exactly is the tradeoff between the number of terms aggregated and efficiency in computation with regards to relevance evaluation?

We tested the retrieval performance achieved for the 50 provided queries together with the given relevance information used in the TREC evaluations. The basic question here is 'How many of the relevant documents will be retrieved?'. Each of the 50 queries is sent to the search system in order to answer this question. In the search system, we used three different settings for term weighting:

1. No document frequency information (i.e., $df = 1$ for all terms)
2. Real document frequency information obtained from the full index
3. Document frequency estimations computed by the P2P hybrid aggregation method

Evaluation Measures

Since there exist measures to incorporate both indicators in one value, we make use of such a measure called 'mean average precision (MAP)'. It is a single-figure measure for precision at all different recall levels, and has become increasingly popular within the TREC community. See Section 2.2.2 for details.

Experimental Settings

An illustrative overview of the retrieval experiments is given in Figure 7.8. We first partition the collection according to the different similarity and skew distribution setups. The input for the experiments is an unstructured P2P network topology, as shown on the top of the figure. Then, we employ DESENT and gossiping for estimating the global document frequencies. Both can be applied simultaneously, since gossiping is only used for the terms not covered and aggregated upwards by DESENT. The output of this phase is the estimated document frequency values for all terms, both high-frequency and low-frequency terms. Finally, both lists are merged to provide a comprehensive estimate for as many terms as possible. We further perform retrieval experiments based on these estimated frequencies and compare to both the case of centralised (real) document ones and missing document frequency information (i.e., all document frequencies are set to one).

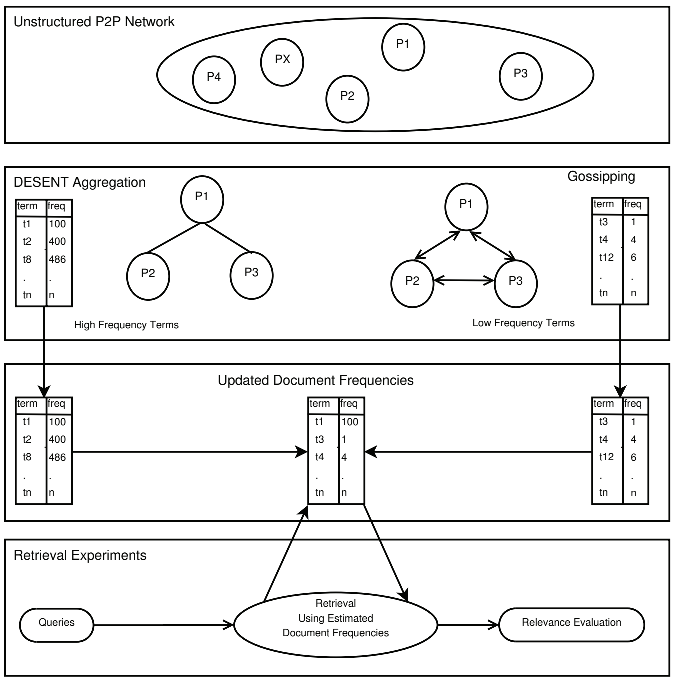


Figure 7.8: Overview of the architecture used in the experiments. The full system consists of DESENT, gossiping, frequency merging, and the retrieval component.

Experimental Results

The TREC8 collection has been used in many cases for relevance retrieval experiments. The reported results vary according to the additional techniques the authors used. A mean average precision of 0.3272 is, for example, reported in [55], where the authors use a combination of the probabilistic model and the language model approach. Theme-based document retrieval using an extensive lexical knowledge base is applied to the same problem in [61], and the best result reported there is a mean average precision of 0.413. Again, the authors use techniques additional to basic ranking and retrieval.

In our case, we employ a basic ranking model based on $tfidf$ weighting used in Java open source search engine Lucene.[5] All techniques we employ aim at improving document frequency estimation, which in turn could be used to improve retrieval results for different weightings. Hence, we do not compete with other groups working on this collection, basically all other ranking techniques could be used on top of our approach to improve the final results.

Tables 7.5, 7.6, and 7.7 show an overview of the obtained results; we list both MAP,

[5]`http://lucene.apache.org`

Table 7.5: Baseline retrieval results on the TREC8 ad hoc collection. We show the retrieval results in terms of MAP and Recall on a centralised index as well as results obtained without document frequency information available.

Setup	**MAP**	**R**
Full centralised index (baseline)	0.228	0.651
Full centralised index without df info	0.197	0.553

Table 7.6: Retrieval results on the TREC8 ad hoc collection. We show the retrieval results based document frequency estimation in terms of MAP and Recall using hierarchical aggregation. Results are averaged over 10 runs.

#Peers	**Setup**	**MAP**	**R**
100	1	0.214	0.628
100	2	0.213	0.626
100	3	0.217	0.635
100	4	0.216	0.633
200	1	0.214	0.629
200	2	0.215	0.629
200	3	0.217	0.634
200	4	0.215	0.633
300	1	0.215	0.628
300	2	0.215	0.630
300	3	0.217	0.633
300	4	0.216	0.634

and recall. We list the results on a centralised index as well as results obtained without document frequency information, available in Table 7.5. The experiment without *df* weighting builds the absolute baseline for further experiments. In a distributed setting it is always easily possible to assume 1 for all document frequencies.

Table 7.6 shows the results obtained when applying hierarchical aggregation only. It is interesting to note that the results are quite close to the centralised case. For instance, the average precision is around 0.215 and recall 0.631, when the corresponding values of the centralised case are 0.228 and 0.651. Moreover, the results are stable across experimental setups and the different numbers of peers apart from small fluctuations.

Furthermore, we show the results obtained when using the hybrid approach in Table 7.7. In this scenario, we rely on both the values obtained by hierarchical and the gossip-based aggregation. The results are always better than the hierarchical aggregation values alone consistently across all settings. For 100 peers and experimental setup id=1, the results exceed the results obtained from the centralised

Table 7.7: Retrieval results on the TREC8 ad hoc collection. We show the retrieval results based on document frequency estimation using hybrid aggregation. Results are averaged over 10 runs of hybrid aggregation.

#Peers	Setup	MAP	R
100	1	0.229	0.648
100	2	0.227	0.648
100	3	0.226	0.641
100	4	0.226	0.640
200	1	0.228	0.646
200	2	0.224	0.644
200	3	0.225	0.641
200	4	0.224	0.633
300	1	0.228	0.646
300	2	0.224	0.640
300	3	0.225	0.642
300	4	0.225	0.639

index. However, this is due to the limited number of queries. This approach yields better results only for 7 out of the 50 queries. Across all setups, we never get far below the results from the centralised index. In fact, for experimental setup id=1 (equal distribution and high similarity within peers), we are consistently equal to the centralised case.

We also provide an overview of estimation accuracy for document frequency values in Table 7.8. The results are drawn from 10 runs of retrieval evaluation and given for the four different experimental settings (see Table 7.4). We list the percentage of frequencies which are correctly estimated, i.e., equal to the real frequencies in the centralised case, as well as the average, minimum and maximum difference between estimated and real frequency. Further, we list the median, i.e., the difference that devides the deviations into two equal parts. A median of 11 for example means that half of the estimated frequencies differ by less than 11 from the real value. One finding is that some frequencies differ by large numbers—as seen in the maximum and mean columns. On the other hand, we see that in most of the cases the median is quite small, showing that a large part of the estimates are quite close to the real values.

Even though the estimation quality decreases with a higher number of peers, the respective retrieval experiments deliver high precision as outlined in the previous section. We therefore showed that in most cases the estimations are satisfactory, if not in the absolute value, in the form they are used in the ranking function.

Table 7.8: Estimation of document frequencies on TREC8 corpus. Results are averaged over 10 aggregation runs. We show the different numbers of peers, experimental setups as well as the percentage of correctly estimated document frequencies used in TREC8 queries. We also list the maximum, mean, and median differences to the correct frequencies.

#Peers	Setup	Equal DF	Max.	Mean	Median
100	1	0.14	16784.0	955.40	2.0
100	2	0.09	15516.0	1010.85	14.0
100	3	0.06	21789.0	3096.63	14.0
100	4	0.06	21882.0	2848.52	14.5
200	1	0.13	16086.0	1063.73	3.0
200	2	0.08	14977.0	1441.72	14.0
200	3	0.06	29587.0	4189.28	41.0
200	4	0.04	30887.0	4680.36	36.5
300	1	0.0	18320.0	1833.65	70.0
300	2	0.013	18806.0	1986.70	25.0
300	3	0.008	26704.0	3502.5	54.0
300	4	0.006	29450.0	4802.41	70.5

7.7 Entity Search in P2P Networks

Contrary to *tf-idf* based approaches, language models rely on the collection frequency of terms, i.e., how often a term in total occurs in the collection, as opposed to the number of documents it occurs in. The research results described in this chapter, however, can directly be applied to the language modelling use case. We described a general approach for information aggretation in P2P networks, which is not specific for document frequency estimation. In fact, switching to collection frequencies is straightforward. Each collection has information about the collection frequencies (i.e., the number of occurrences of a term in the collection), and can easily aggregate this value instead of the document frequencies we aggregated so far.

Once this information is aggregated throughout the P2P network it can be used for estimating the probabilities in the collection-wide background model and as such be integrated in the models described in Chapters 3 and 4. This happens in the smoothing process, where missing probabilities are substituted by a weighted mixture of the global probabilities. To evaluate our approaches we present three settings for background collection frequencies:

- Central Language Models: all collection frequencies are computed from the central collection.
- Baseline Lucene: we disregard collection-wide statistics to present an absolute

Table 7.9: Entity search baselines.

Retrieval Model	MAP	MRR	P@10	NDCG
LM centralised	0.1940	0.5046	0.2599	0.3675
Full centralised index without df info	0.0932	0.3038	0.1289	0.1994
LM aggregated 100 peers 50 terms	0.1488	0.5079	0.2539	0.2991

baseline. In this setting, all document frequencies are set to one (compare to the results for the TREC8 collection presented in Table 7.5).

- Estimates: we use the results of the P2P estimation for retrieval. They are used as a drop-in replacement for the global, collection-wide statistics.

Table 7.9 shows the baselines for our entity experiments. The scenario we present for evaluation now is taken from the setting introduced in Chapter 5. In the following we present how P2P estimation techniques can be applied to this setting. First, we show a central language model baseline. This will be the ideal result to compare to, e.g., the upper bounds for our estimations. This baseline incorporates Dirichlet smoothing with the smoothing parameter μ set to the collection-wide average document length. This result is presented in the first line of Table 7.9. It maybe noted that with a MAP of 0.1940, this is competitive. Next, we present results achieved with Lucene scoring without background information in the second line of the same table (in this case without document frequency information). This baseline is similar to the ones used on the non-entity collections and we achieve a rather low MAP of 0.0932, which points out the possible improvements that can be achieved with collection-wide estimates. In further experiments, we will try to improve over this deadline.

We show that a number of 50 terms per peer already is enough to achieve satisfactory performance, elevating the MAP to 0.1488, a substantial improvement from the 0.0932 that we consider as absolute baseline, and well in reach of the 0.1940 that we consider the optimal case. We want to note that further improvements are well possible with adjusting some of the parameters like the number of terms aggregated per peer and the amount of gossiping. Overall, these results show the competitiveness of our aggregation techniques for entity search, in fact, the results across the diverse data collections show that the estimation technique generalises and is not specific to a certain type of data.

7.8 Conclusions

In this chapter, an efficient hybrid method for aggregating document frequency values was presented, suitable for application in loosely-coupled unstructured P2P networks. The hybrid method combines hierarchical aggregation of carefully selected local terms with high frequency values, with gossip-based aggregation of the remaining low-frequency terms. We also provided an extensive experimental evaluation on three document collections, assessing both term ranking quality and retrieval results, having as reference point the results obtained by a centralised system that has the complete document collection available. The results show that our hybrid aggregation performs very well for variable setups. We further showed the direct applicability of our approaches to the distributed entity search use case by showing experimental results on an entity benchmark collection.

In our future work, we intend to deploy a widely distributed information retrieval system that uses hybrid aggregation as building block for estimating document frequency values.

Recap: Part III

In the **Distributed Aspects** part of this thesis, we described how to apply entity search methods in the distributed context—this was largely motivated by scalability issues which we deem realistic with the current growth of the WoD. First, we formalised the federated entity search problem in a language modelling setting in Chapter 5. To this end, we introduced all components of federated search: collection representation, collection selection, and results merging. We introduced a distributed testbed for entity search based on existing benchmark corpora. Further, we provided baseline results for all thee steps on that collection. We also provided some discussion of cooperative and uncooperative settings.

Having found the collection selection component to be particularly interesting and helpful in the retrieval pipeline, we focused on it in Chapter 6. Based on the observations made in the previous chapter, we proposed a new method for collection ranking and selection based on the characteristics of entity search requirements. We showed the applicability of our method on the testbed introduced in Chapter 5 and showed its competitiveness.

Driven by the ever-increasing size of the WoD, we applied P2P techniques to the distributed entity search problem in Chapter 7. We first introduced the specifics of the P2P setting, namely its increased level of dynamicity. We then described some of the basic components of P2P systems, specifically targeting the estimation of global statistics, in our case collection frequency information of terms. In an additional set of experiments, we showed the direct applicability to the entity search use case.

Part IV

Discussion and Outlook

In the last part of the thesis, we summarise our work and draw conclusions. To that end, we restate our initial research questions and answer them with respect to the research conducted in the course of the previous parts. Additionally, we provide an outlook on future work.

Chapter 8

Conclusions and Future Work

In this chapter, we will first summarise our main findings. We then draw conclusions in terms of putting the initial research questions in context with these findings. Finally, we give an outlook on future work pointing out the most interesting areas to continue the research ideas this thesis is evolving around and give a more general outlook on future research concerned with the Web of Data.

8.1 Contributions

We explained the main approaches to entity search and introduced extensions and modifications in Chapter 3. We provided a thorough overview of the area of entity search, putting a special focus on distributed aspects. We introduced the main use case of entity search, i.e., "answering arbitrary information needs related to particular aspects of objects [entities], expressed in unconstrained natural language and resolved using a collection of structured data" [92]. The structured data in our case consists of billions of *subject-predicate-object triples* which represent entities and their relations. We also described the list search task. We described our initial interest in entity search as motivated by our contribution to the Semantic Search Challenge. We showed the applicability of standard IR approaches to the problem and introduced a retrieval model taking into account both plain text and some (limited) structure of the data in terms of predicates. We found these models to perform well, but expected more mileage to be gained by incorporating more structure Based on these findings, we proposed an extended model, taking into account both plain text and entity structure based on structured retrieval models in Chapter 4.

We further introduced the use case of distributed entity search in the later chapters. To that end, we applied the models developed for the central case and applied them in a federated search environment. First, we formalised the entity search problem in a federated environment and provided baseline results in Chapter 5. There, we also found that the collection ranking component of federated search can specifically benefit from improved entity representations. We then developed new models based on this finding to improve collection ranking and, in our experiments, showed their effectiveness in Chapter 6. Further motivated by the ever-increasing size of the WoD, we investigated the underlying foundations needed for text search in the P2P context. We opened for entity search and other types of search across a high number of independent peers in Chapter 7. First we gave an introduction to the P2P setting, first and foremost by explaining its increased level of dynamicity. Further, we described some of the basic components of P2P systems, specifically targeting the estimation of global statistics, in our case collection frequency information of terms. Also, we showed the direct applicability to the entity search use case in an additional set of experiments.

The main contributions of this thesis are:

- The submission to the Semantic Search Challenge, showing the competitiveness of our approaches in an international forum.
- Analysis of the feasibility of current document retrieval models to the entity search task.
- The application of structured (semantic) retrieval models in the context of entity search.
- A hierarchical model for entity search in the Web of Data where we treat predicates differently based on their type.
- The formalisation of the federated entity search task in a generative language modelling framework.
- The development of benchmark data sets for federated entity search.
- Both a thorough analysis of entity search in federated search environments and the formulation of advanced techniques.
- A scalable model for term statistic aggregation in P2P networks and its application to the entity search scenario.

In the following, we reiterate the research questions laid out in the beginning of the thesis in Chapter 1 and present our answers that we found in the course of the thesis.

RQ1: How can traditional ad-hoc document retrieval techniques be applied in the context of the Web of Data?

In Chapter 3 we state the limited direct applicability, yet, we find that they are strong baselines when entities are represented as plain text.

RQ2: How can the structure of entities be exploited for the purpose of ad-hoc retrieval?

We showed the very competitive nature of structured retrieval models in Chapter 4. Our experiments clearly indicated that they can outperform the single-field case.

RQ3: How does field weighting affect search quality?

The structured retrieval models we applied work well in many cases even without field weighting in Chapter 4. Our experiments showed that the estimation of field weights is an open research area, particularly in the context of the WoD– due to its potentially large number of predicates or fields.

RQ4: Can existing, standard federated search techniques be applied to entity search in the Web of Data?

Federated search techniques are applicable in the WoD, however, entity modelling plays an important role here. All components of federated search strongly depend on collection representation, which is strongly connected to entity modelling.

RQ5: Can federated entity search benefit from improved entity modelling?

We presented research results showing that proper entity modelling makes federated search techniques a feasible option for entity search in Chapter 6.

RQ6: Is P2P search a viable alternative to broker-based (i.e., federated search) architectures for entity retrieval?

We provide a thorough investigation of P2P document retrieval in Chapter 7. We showed that broker-based architectures are not directly feasible for P2P document retrieval and its dynamic requirements. We also described how P2P can be a viable alternative if aggregation of global information is accounted for.

RQ7: How can the proposed frequency estimation technique be further improved?

In Chapter 7, we showed that improvements in frequency estimation benefit both document and entity retrieval. We described the combination of hierarchical aggregation and gossiping and how it improves the results. Our experiments are indicative of a general strong influence of estimation techniques and retrieval performance.

To sum up, we answered all research questions stated in Chapter 1, albeit some of them (e.g., RQ5 and RQ3) are subject to more research, as we will take up in our description of future work.

8.2 Future Work

Future work in entity search is going into multiple directions. First and foremost we see much work to be done in the area of query analysis or query understanding. This is also reflected by the analysis of the participants of the second strategic workshop on IR [1], where much focus is put on designing and realising holistic systems based on language understanding and question answering. Entity search can play a vital role in multiple components of such systems such as query understanding and results presentation.

The mapping of query terms and phrases to fields of the model and the estimation of weights to be assigned to these fields is a promising direction in which we already performed initial experiments (see RQ3). This work will seamlessly fit into the overall topic of combining structured retrieval and semantic search. This means also that a wider range of query types should be considered and the ability of future models to accommodate for these different queries will be of vital importance to further improvements (i.e., a long, free text query will be easier to map to fields than a simple, short keyword query where mapping might actually hurt performance instead of improving it). This requires a fundamental ability of the retrieval models to incorporate weights on a field- and query component basis (i.e., the mapping of query components to fields instead of mapping the query as a whole or only its individual terms).

A related area is query type detection; we introduced a new line of research in this direction in the area of query type identification in [7]. In that paper, a dynamic, semantic query analysis component incorporating both segmentation of queries, both entity and type detection, and partial query-to-property mapping is proposed.

Even though some of our models were used in a large-scale setup (collections of up to 80 million entities), a larger focus on scalability is desirable. This becomes more interesting in the context of federated search, as investigated earlier in the thesis. The amount of Linked Data is expected to grow with new actors publishing their data on the Web. The inclusion of these new data sources, i.e., the ability of systems to update their indices will be of crucial interest.

The combination of full text and structured data is another challenging area for future work. Many data sets combine both elements and therefore require adequate retrieval models. The Linked Data Track of the 2012 edition of INEX, e.g., featured such a collection [1], which we participated in. This benchmark aims at helping to close the gap between keyword search and Semantic Web techniques. The techniques introduced in this thesis are not directly applicable, but they are a good foundation for further research in this direction since we are able to handle predicates of different size. However, our solutions have not come far enough, we believe dynamic field weighting to play an important role in solving this problem.

The integration of query annotation in retrieval models is another large open question. Query annotation can comprise techniques such as query expansion or phrase detection. Both can undoubtedly influence retrieval effectiveness, however, it is not entirely clear how they can be incorporated in probabilistic retrieval models in

[1] `https://inex.mmci.uni-saarland.de/tracks/lod/`

the best way. One major challenge in this respect is the consideration of term or phrase priors as well as their estimation.

Last but not least, entity search should be seen in the context of users, their information needs, and eventually the queries users formulate in order to search for the information they want to find. The integration of entity search in existing systems is therefore another interesting field for future research. Parts of this research will be concerned with query analysis and intent discovery, whereas others will target query logs and the analysis of user behaviour on the Web, i.e., how does entity search fit into the more general use case of web search.

8.3 Outlook

The Semantic Web was introduced with confidence and high hopes. Some of these hopes were not quite fulfilled in the beginning. However, we have come to a point where many of its technologies have reached a certain level of maturity and are in widespread use; the same goes for its standards for publishing semantic data. This includes not only data repositories published as linked data but is also driven by the increasing use of RDFa and microdata, allowing users to "semantify" their own data with rather low technical requirements. The better availability of semantic data on the web is without a doubt a main contributor to its acceptance. Search in the Web of Data is one of them, combining both advanced query processing and search in RDF data—one of the foundations of the Semantic Web. At the same time commercial search engines have begun to pick up on these trends (see Google knowledge graph).

While these definitely are positive points, much work remains to be done. Semantic collections are still scattered across the Web and only in a few cases truly link to each other. The Linked Open Data initiative is a first step in this direction, but definitely needs support by intelligent tools in combining data from different sources. Once we are closer to that goal, ensuring quality levels that guarantee a satisfactory search experience for users will become vital. This will help facilitate both increased amounts of data available and their high quality.

References

[1] James Allan, Bruce Croft, Alistair Moffat, and Mark Sanderson. Frontiers, challenges, and opportunities for information retrieval: Report from SWIRL'12. *SIGIR Forum*, 46(1):2–32, June 2012.

[2] Leif Azzopardi, Mark Baillie, and Fabio Crestani. Adaptive query-based sampling for distributed IR. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*, pages 605–606, Seattle, WA, USA, August 6-11 2006. ACM.

[3] Mark Baillie, Leif Azzopardi, and Fabio Crestani. Adaptive query-based sampling of distributed collections. In *Proceedings of the 13th International Conference on String Processing and Information Retrieval (SPIRE'06)*, pages 316–328, Glasgow, Scotland, United Kingdom, October 11-13 2006. Springer.

[4] Wolf-Tilo Balke. Supporting information retrieval in peer-to-peer systems. In *Peer-to-peer Systems and Applications*, pages 337–352. Springer, 2005.

[5] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. DL meets P2P - distributed document retrieval based on classification and content. In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL'05)*, pages 379–390, Vienna, Austria, September 18-23 2005. Springer.

[6] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. Progressive distributed top k retrieval in peer-to-peer networks. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 174–185, Tokyo, Japan, April 5-8 2005. IEEE Computer Society.

[7] 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)*, Maui, HI, USA, October 29 - November 2 2012. Poster, accepted for publication.

[8] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR*, pages 43–50, Seattle, WA, USA, August 6-11 2006. ACM.

[9] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. A language modeling framework for expert finding. *Information Processing and Management*, 45: 1–19, 2009.

[10] Krisztian Balog, Ian Soboroff, Paul Thomas, Nick Craswell, Arjen P. de Vries, and Peter Bailey. Overview of the TREC 2008 enterprise track. In *Proceedings of the 17th Text Retrieval Conference (TREC'08)*. NIST, 2009.

[11] Krisztian Balog, Marc Bron, and Maarten de Rijke. Category-based query modeling for entity search. In *Proceedings of the 32nd European Conference on Information Retrieval (ECIR'09)*, pages 319–331, Milton Keynes, United Kingdom, March 28-31 2010. Springer.

[12] Krisztian Balog, Arjen P. de Vries, Pavel Serdyukov, Paul Thomas, and Thijs Westerveld. Overview of the TREC 2009 entity track. In *Proceedings of the 18th Text REtrieval Conference (TREC'09)*, 2010.

[13] Krisztian Balog, Marek Ciglan, Robert Neumayer, Wei Wei, and Kjetil Nørvåg. NTNU at SemSearch 2011. In *Proceedings of the 4th International Semantic Search Workshop of the 20th Int. World Wide Web Conference WWW2011)*, pages –, Hyderabad, India, April 19-21 2011.

[14] Krisztian Balog, Pavel Serdyukov, and Arjen P. de Vries. Overview of the TREC 2010 entity track. In *Proceedings of the 19th Text REtrieval Conference (TREC'10)*, Gaithersburg, MD, USA, November 15 - 18 2011. NIST.

[15] Krisztian Balog, Robert Neumayer, and Kjetil Nørvåg. Collection ranking and selection for federated entity search. In *Proceedings of 18th International Symposium of String Processing and Information Retrieval (SPIRE'12)*, Cartagena, Colombia, October 21-25 2012. Springer. Accepted for publication.

[16] Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haussmann. A case for semantic full-text search. In *Proceedings of the 1st Joint International Workshop on Entity-oriented and Semantic Search (JIWES'12)*, 2012.

[17] Florian Bäurle. A user interface for semantic full text search. Master's thesis, University of Freiburg, Faculty of Engineering, July 2011.

[18] Matthias Bender, Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. Global document frequency estimation in peer-to-peer web search. In *Proceedings of the 9th International Workshop on the Web and Databases (WebDB'06)*, Chicago, IL, USA, June 30 2006.

[19] Tim Berners-Lee and Mark Fischetti. *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor.* Harper San Francisco, September 1999.

[20] Roi Blanco, Harry Halpin, Daniel M. Herzig, Peter Mika, Jeffrey Pound, Henry S. Thompson, and Thanh Tran Duc. Entity search evaluation over structured web data. In *Proceedings of the 1st International Workshop on Entity-Oriented Search (EOS)*, Beijing, China, July 28 2011.

[21] Roi Blanco, Peter Mika, and Sebastiano Vigna. Effective and efficient entity search in RDF data. In *Proceedings of the 10th International Semantic Web Conference (ISWC'11)*, pages 83–97, Bonn, Germany, October 23-27 2011. Springer.

[22] 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, New York, NY, USA, 2010. ACM.

[23] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 21–28, Seattle, WA, USA, July 9-13 1995. ACM.

[24] Jamie Callan. *Advances in Information Retrieval*, volume 7, chapter Distributed Information Retrieval. Kluwer Academic Publishers, 2000.

[25] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, April 2001.

[26] Stéphane Campinas, Renaud Delbru, Nur Aini Rakhmawati, Diego Ceccarelli, and Giovanni Tummarello. Sindice BM25F at SemSearch 2011. In *Proceedings of the 4th International Semantic Search Workshop (SEMSEARCH'11)*, 2011.

[27] Marek Ciglan, Kjetil Nørvåg, and Ladislav Hluchý. The semsets model for ad-hoc semantic list search. In *Proceedings of the 21st World Wide Web Conference 2012 (WWW'12)*, pages 131–140, Lyon, France, April 16-20 2012. ACM.

[28] Philipp Cimiano, Peter Haase, and Jörg Heizmann. Porting natural language interfaces between domains: an experimental user study with the orakel system. In *Proceedings of the 12th international Conference on Intelligent User interfaces (IUI'07)*, pages 180–189, Honolulu, Hawaii, USA, 2007. ACM.

[29] Francisco Matias Cuenca-Acuna, Christopher Peery, Richard P. Martin, and Thu D. Nguyen. PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of the 12th International Symposium on High-Performance Distributed Computing (HPDC'03)*, pages 236–249, Seattle, WA, USA, June 22-24 2003. IEEE Computer Society.

[30] Jeff Dalton and Sam Huston. Semantic entity retrieval using web queries over structured RDF data. In *Proceedings of the 3rd International Semantic Search Workshop (SEMSEARCH'10)*, 2010.

[31] Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of ESWC'2010*, Heraklion, Crete, Greece, May 30 - June 3 2010. Springer.

[32] 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*, pages 245–251, 2008.

[33] Renaud Delbru, Nur Aini Rakhmawati, and Giovanni Tummarello. Sindice at SemSearch 2010. In *Proceedings of the 3rd International Semantic Search Workshop (SEMSEARCH'10)*, 2010.

[34] Renaud Delbru, Nickolai Toupikov, Michele Catasta, and Giovanni Tummarello. A node indexing scheme for web entity retrieval. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC'10)*, pages 240–256, Heraklion, Crete, Greece, May 30 - June 3 2010. Springer.

[35] Gianluca Demartini, Tereza Iofciu, and Arjen P. De Vries. Overview of the INEX 2009 entity ranking track. In *Proceedings of the 8th International Workshop of the Initiative for the evaluation Of XML retrieval (INEX'10)*, pages 254–264. Springer, 2010.

[36] Gianluca Demartini, Philipp Kärger, George Papadakis, and Peter Fankhauser. L3S Research Center at the Semsearch 2010 Evaluation for Entity Search Track. In *Proceedings of the 3rd International Semantic Search Workshop (SEMSEARCH'10)*, 2010.

[37] Christos Doulkeridis, Kjetil Nørvåg, and Michalis Vazirgiannis. Scalable semantic overlay generation for P2P-based digital libraries. In *Proceedings of the 10th European Conference on Research and Advanced Technology for Digital Libraries (ECDL'06)*, pages 26–38, Alicante, Spain, September 17-22 2006. Springer.

[38] Christos Doulkeridis, Kjetil Nørvåg, and Michalis Vazirgiannis. DESENT: decentralized and distributed semantic overlay generation in P2P networks. *IEEE Journal on Selected Areas in Communications*, 25(1):25–34, January 2007.

[39] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. Language-model-based ranking for queries on RDF-graphs. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09)*, pages 977–986, Hong Kong, China, November 2-6 2009. ACM.

[40] Jonathan L. Elsas, Jaime Arguello, Jamie Callan, and Jaime G. Carbonell. Retrieval and feedback models for blog feed search. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*, pages 347–354, Napa Valley, California, USA, October 26-30 2008. ACM.

[41] Óscar Ferrández, Rubén Izquierdo, Sergio Ferrández, and José Luis Vicedo González. Addressing ontology-based question answering with collections of user queries. *Information Processing and Management*, 45(2):175–188, 2009.

[42] Luis Gravano and Hector Garcia-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB'95)*, pages 78–89, San Francisco, CA, USA, 1995.

[43] Benjamin M. Gross. Navigation, organization, and retrieval in personal digital libraries of email. In *Proceedings of the 5th International Conference on Asian Digital Libraries (ICADL'02)*, pages 258–259, Singapore, December 11-14 2002. Springer.

[44] Indranil Gupta, Robbert van Renesse, and Kenneth P. Birman. Scalable fault-tolerant aggregation in large process groups. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'01) (formerly: FTCS)*, pages 433–442, Göteborg, Sweden, July 1-4 2001. IEEE Computer Society.

[45] 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, Hyderabad, India, March 28 - April 1 2011. ACM.

[46] 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'10)*, Shanghai, China, November 8 2010.

[47] Daniel M. Herzig and Thanh Duc Tran. Semsearch 2010 entity search track. scoring model for entity search on RDF graphs. In *Proceedings of the 3rd International Semantic Search Workshop (SEMSEARCH'10)*, 2010.

[48] Djoerd Hiemstra and Arjen P. de Vries. Relating the new language models of information retrieval to the traditional retrieval models. Technical report, University of Twente CTIT, May 2000. Technical Report TR-CTIT-00-09.

[49] Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, 2005.

[50] Jaap Kamps, Shlomo Geva, Andrew Trotman, Alan Woodley, and Marijn Koolen. Overview of the INEX 2008 ad hoc track. In *Advances in Focused*

Retrieval: 7th Intl. Workshop of the Initiative for the Evaluation of XML Retrieval, pages 1–28. Springer, 2010.

[51] Esther Kaufmann, Abraham Bernstein, and Lorenz Fischer. NLP-Reduce: a 'naïve' but domain-independent natural language interface for querying ontologies. In *Proceedings of ESWC 2007 (Demo Session)*, Innsbruck, Austria, June 3-7 2007. Springer.

[52] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS'03)*, pages 482–491, Cambridge, MA, USA, October 11-14 2003. IEEE Computer Society.

[53] Jinyoung Kim and W. Bruce Croft. A field relevance model for structured document retrieval. In *Proceedings of the 34rd European Conference on Information Retrieval (ECIR'12)*, pages 97–108, Barcelona, Spain, April 1 - 5 2012.

[54] Jinyoung Kim, Xiaobing Xue, and W. Bruce Croft. A probabilistic retrieval model for semistructured data. In *Proceedings of the 31st European Conference on Information Retrieval (ECIR'09)*, pages 228–239, Toulouse, France, April 6-9 2009. Springer.

[55] K. L. Kwok, Laszlo Grunfeld, and M. Chan. Trec-8 ad-hoc, query and filtering track experiments using PIRCS. In *Proceedings of TREC-8*, 1999.

[56] Xitong Liu and Hui Fang. A study of entity search in semantic search workshop. In *Proceedings of the 3rd International Semantic Search Workshop (SEMSEARCH'10)*, 2010.

[57] Vanessa Lopez, Victoria S. Uren, Enrico Motta, and Michele Pasin. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *Journal of Web Semantics*, 5(2):72–105, 2007.

[58] Vanessa Lopez, Andriy Nikolov, Miriam Fernández, Marta Sabou, Victoria S. Uren, and Enrico Motta. Merging and ranking answers in the semantic web: The wisdom of crowds. In *Proceedings of the 4th Asian Semantic Web Conference (ASWC'09)*, pages 135–152, Shanghai, China, December 6-9 2009. Springer.

[59] David E. Losada and Leif Azzopardi. An analysis on document length retrieval trends in language modeling smoothing. *Information Retrieval*, 11(2): 109–138, April 2008.

[60] Jie Lu and Jamie Callan. Full-text federated search of text-based digital libraries in peer-to-peer networks. *Information Retrieval*, 9(4):477–498, 2006.

[61] Kavi Mahesh, Jacquelynn Kud, and Paul Dixon. Oracle at TREC8: A lexical approach. In *Proceedings of TREC-8*, 1999.

[62] Saadia Malik, Gabriella Kazai, Mounia Lalmas, and Norbert Fuhr. Overview of the inex 2005. In *Proceedings of the 4th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX'05)*, pages 1–15, Dagstuhl Castle, Germany, November 28-29 2005.

[63] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, United Kingdom, 2008.

[64] Rudolf Mayer and Robert Neumayer. Multi-modal analysis of music: A large-scale evaluation. In *Proceedings of the Workshop on Exploring Musical Information Spaces, (WEMIS'09)*, pages 30–35, Corfu, Greece, October 1-2 2009.

[65] Rudolf Mayer, Robert Neumayer, Doris Baum, and Andreas Rauber. Analytic comparison of Self-organising Maps. In *Proceedings of the 7th International Workshop on Self-Organizing Maps (WSOM'09)*, pages 182–190, St. Augustine, FL, USA, 2009. Springer.

[66] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Interacting with (semi-) automatically extracted context of digital objects. In *Proceedings of the Workshop on Context, Information And Ontologies (CIAO'09)*, pages 1–9, Heraklion, Greece, June 1 2009. ACM.

[67] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Data recovery from distributed personal repositories. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries (ECDL'09)*, Corfu, Greece, September 27 - October 2 2009. Springer.

[68] Edgar Meij, Marc Bron, Laura Hollink, Bouke Huurnink, and Maarten Rijke. Learning semantic query suggestions. In *Proceedings of the 8th International Semantic Web Conference (ISWC'09)*, pages 424–440, Chantilly, VA, USA, October 25-29 2009. Springer.

[69] Sergey Melnik, Sriram Raghavan, Beverly Yang, and Hector Garcia-Molina. Building a distributed full-text index for the web. *ACM Transactions on Information Systems*, 19(3):217–241, 2001.

[70] Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. Minerva∞: A scalable efficient peer-to-peer search engine. In *Proceedings of the 6th International ACM/IFIP/USENIX Middleware Conference (MIDDLEWARE'05)*, pages 60–81, Grenoble, France, November 28 - December 2 2005. Springer.

[71] Peter Mika, Edgar Meij, and Hugo Zaragoza. Investigating the semantic gap through query log analysis. In *Proceedings of the 8th International Semantic Web Conference (ISWC'09)*, pages 441–455, Chantilly, VA, USA, October 25-29 2009. Springer.

[72] David N. Milne and Ian H. Witten. Learning to link with Wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*, pages 509–518, Napa Valley, California, USA, October 26-30 2008. ACM.

[73] Michael Minock. C-Phrase: a system for building robust natural language interfaces to databases. *Data and Knowledge Engineering*, 69(3), 2010.

[74] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[75] Robert Neumayer and Kjetil Nørvåg. Evaluation of feature combination approaches for text categorisation. In *Proceedings of the 19th International Symposium on Methodologies for Intelligent Systems (ISMIS'11)*, pages 438 – 448, Warsaw, Poland, June 28-30 2011.

[76] Robert Neumayer, Christos Doulkeridis, and Kjetil Nørvåg. Aggregation of document frequencies in unstructured P2P networks. In *Proceedings of 10th International Conference on Web Information Systems Engineering (WISE'09)*, pages 29–42, Poznan, Poland, October 5-7 2009. Springer.

[77] Robert Neumayer, Christos Doulkeridis, and Kjetil Nørvåg. A hybrid approach for estimating document frequencies in unstructured P2P networks. *Information Systems*, 36(3):579–595, May 2011.

[78] Robert Neumayer, Rudolf Mayer, and Kjetil Nørvåg. Combination of feature selection methods for text categorisation. In *Proceedings of the 33rd European Conference on Information Retrieval (ECIR'11)*, pages 763–766, Dublin, Ireland, April 19-21 2011.

[79] Robert Neumayer, George Tsatsaronis, and Kjetil Nørvåg. TRUMIT: A tool to support large-scale mining of text association rules. In *Proceedings of the 10th European Conference on Machine Learning and Knowledge Discovery in Databases - European Conference, (ECML PKDD 2011)*, pages 646–649, Athens, Greece, September 5-9 2011. Springer.

[80] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. On the modeling of entities for ad-hoc entity search in the web of data. In *Proceedings of the 34rd European Conference on Information Retrieval (ECIR'12)*, pages 133–145, Barcelona, Spain, April 1 - 5 2012.

[81] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. When simple is (more than) good enough: Effective semantic search with (almost) no semantics. In *Proceedings of the 34rd European Conference on Information Retrieval (ECIR'12)*, pages 540–543, Barcelona, Spain, April 1 - 5 2012.

[82] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. Ranking distributed knowledge repositories. In *Proceedings of the International Conference on Theory and Practice of Digital Libraries Research and Advanced Technology for Digital Libraries (TPDL'12)*, pages 486–491, Paphos, Cyprus, September 23 - 27 2012. Springer.

[83] Henrik Nottelmann and Norbert Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03)*, pages 290–297, Toronto, Canada, July 28 - August 1 2003. ACM.

[84] Henrik Nottelmann and Norbert Fuhr. Combining cori and the decision-theoretic approach for advanced resource selection. In *Proceedings of the 26th European Conference on Information Retrieval (ECIR'04)*, pages 138–153, Sunderland, United Kingdom, April 5-7 2004. Springer.

[85] Henrik Nottelmann and Norbert Fuhr. Comparing different architectures for query routing in peer-to-peer networks. In *Proceedings of the 28th European Conference on Information Retrieval (ECIR'06)*, pages 253–264, London, United Kingdom, April 10-12 2006. Springer.

[86] 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, Toronto, Canada, July 28 - August 1 2003. ACM.

[87] Paul Ogilvie and Jamie Callan. Hierarchical language models for XML component retrieval. In *Proceedings of the 3rd International Workshop of the Initiative for the evaluation Of XML retrieval (INEX'10)*, pages 269–285, 2005.

[88] Odysseas Papapetrou, Sebastian Michel, Matthias Bender, and Gerhard Weikum. On the usage of global document occurrences in peer-to-peer information systems. In *In Proceedings of the On the Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences CoopIS, DOA, and ODBASE*, pages 310–328, Agia Napa, Cyprus, October 31 - November 4 2005. Springer.

[89] Jaehui Park and Sang-goo Lee. Keyword search in relational databases. *Knowledge and Information Systems*, 26:175–193, February 2011.

[90] José R. Pérez-Agüera, Javier Arroyo, Jane Greenberg, Joaquin Perez Iglesias, and Victor Fresno. Using BM25F for semantic search. In *Proceedings of the 3rd International Semantic Search Workshop*, pages 2:1–2:8, 2010.

[91] Ivana Podnar, Toan Luu, Martin Rajman, Fabius Klemm, and Karl Aberer. A peer-to-peer architecture for information retrieval across digital library collections. In *Proceedings of the 10th European Conference on Research and Advanced Technology for Digital Libraries (ECDL'06)*, pages 14–25, Alicante, Spain, September 17-22 2006. Springer.

[92] 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, Raleigh, North Carolina, USA, 2010. ACM.

[93] Paraskevi Raftopoulou, Euripides G. Petrakis, Christos Tryfonopoulos, and Gerhard Weikum. Information retrieval and filtering over self-organising digital libraries. In *Proceedings of the 12th European conference on Research and Advanced Technology for Digital Libraries (ECDL'08)*, pages 320–333, Aarhus, Denmark, September 14 - 19 2008. Springer.

[94] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, 2009.

[95] Stephen E. Robertson, Hugo Zaragoza, and Michael J. Taylor. Simple BM25 extension to multiple weighted fields. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management (CIKM'04)*, pages 42–49, Washington, DC, USA, November 8-13 2004. ACM.

[96] Avi Rosenfeld, Claudia V. Goldman, Gal A. Kaminka, and Sarit Kraus. PHIRST: a distributed architecture for P2P information retrieval. *Information Systems*, 34(2):290–303, 2009.

[97] Ozgur D. Sahin, Fatih Emekçi, Divyakant Agrawal, and Amr El Abbadi. Content-based similarity search over peer-to-peer systems. In *Proceedings of the 2nd International Workshop on Databases, Information Systems, and Peer-to-peer Computing (DBISP2P'04)*, pages 61–78, Toronto, Canada, August 29-30 2005. Springer.

[98] Gerard Salton. *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968.

[99] Jangwon Seo and W. Bruce Croft. Blog site search using resource selection. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*, pages 1053–1062, Napa Valley, California, USA, October 26-30 2008. ACM.

[100] Milad Shokouhi. Central-rank-based collection selection in uncooperative distributed information retrieval. In *Proceedings of the 29th European Conference on Information Retrieval (ECIR'07)*, pages 160–172. Springer-Verlag, April 2-5 2007.

[101] Milad Shokouhi and Luo Si. Federated search. *Foundations and Trends in Information Retrieval*, 5(1):1–102, 2011.

[102] Milad Shokouhi and Justin Zobel. Robust result merging using sample-based score estimates. *ACM Transactions on Information Systems*, 27(3):14:1–14:29, May 2009.

[103] Milad Shokouhi, Falk Scholer, and Justin Zobel. Sample sizes for query probing in uncooperative distributed information retrieval. In *Proceedings of the 8th Asia-Pacific Web Conference (APWeb'06)*, pages 63–75, Harbin, China, January 16-18 2006. Springer.

[104] Luo Si and Jamie Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03)*, pages 298–305, Toronto, Canada, July 28 - August 1 2003. ACM.

[105] Luo Si and Jamie Callan. A semisupervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 21(4):457–491, 2003.

[106] Luo Si, Rong Jin, Jamie Callan, and Paul Ogilvie. A language modeling framework for resource selection and results merging. In *Proceedings of the 11th ACM International Conference on Information and Knowledge Management (CIKM'02)*, pages 391–397, McLean, VA, USA, November 4-9 2002. ACM.

[107] Gleb Skobeltsyn, Toan Luu, Ivana Podnar Zarko, Martin Rajman, and Karl Aberer. Query-driven indexing for scalable peer-to-peer text retrieval. In *Proceedings of the 2nd International Conference on Scalable Information Systems (Infoscale'07)*, page 14, Suzhou, China, June 6-8 2007. ACM.

[108] Amanda Spink, Dietmar Wolfram, Major B. J. Jansen, and Tefko Saracevic. Searching the web: the public and their queries. *Journal of the American Society for Information Science and Technology*, 52:226–234, February 2001.

[109] Torsten Suel, Chandan Mathur, Jo wen Wu, Jiangong Zhang, Alex Delis, Mehdi Kharrazi, Xiaohui Long, and Kulesh Shanmugasundaram. ODISSEA: A peer-to-peer architecture for scalable web search and information retrieval. In *Proceedings of the 6th International Workshop on Web and Databases (WebDB'03)*, pages 67–72, San Diego, CA, USA, June 12-13 2003.

[110] Valentin Tablan, Danica Damljanovic, and Kalina Bontcheva. A natural language query interface to structured information. In *Proceedings of the 5th European Semantic Web Conference (ESWC'08)*, pages 361–375, Tenerife, Canary Islands, Spain, June 1-5 2008. Springer.

[111] Chunqiang Tang and Sandhya Dwarkadas. Hybrid global-local indexing for efficient peer-to-peer information retrieval. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI'04)*, pages 211–224, San Francisco, CA, USA, March 29-31 2004. USENIX.

[112] Paul Thomas and Milad Shokouhi. SUSHI: scoring scaled samples for server selection. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*, pages 419–426, Boston, MA, USA, July 19-23 2009. ACM.

[113] Dolf Trieschnigg. Proof of concept concept-based biomedical information retrieval. Master's thesis, University of Twente, Centre of Telematics and Information Technology, 2010.

[114] Christina Unger and Philipp Cimiano. Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In *Proceedings of the 16th International Conference on Applications of Natural Language to Information Systems (NLDB'11)*, pages 153–160, Alicante, Spain, June 28-30 2011. Springer.

[115] Robbert van Renesse and Adrian Bozdog. Willow: DHT, aggregation, and publish/subscribe in one protocol. In *Proceedings of the 3rd International Workshop on Peer-to-peer Systems (IPTPS'04), Revised Selected Papers*, pages 173–183, La Jolla, CA, USA, February 26-27 2005. Springer.

[116] Robbert van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2): 164–206, 2003.

[117] Charles Lowell Viles and James Cornelius French. On the update of term weights in dynamic information retrieval systems. In *Proceedings of the 4th ACM International Conference on Information and Knowledge Management (CIKM'95)*, pages 167–174, Baltimore, MD, USA, November 28 - December 2 1995. ACM.

[118] Charles Lowell Viles and James Cornelius French. Dissemination of collection wide information in a distributed information retrieval system. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 12–20, Seattle, WA, USA, July 9-13 1995. ACM.

[119] Ellen M. Voorhees. *TREC: Experiment and Evaluation in Information Retrieval.* Digital Libraries and Electronic Publishing. MIT Press, September 2005.

[120] Hans Friedrich Witschel. Global term weights in distributed environments. *Information Processing and Management*, 44(3):1049–1061, 2008.

[121] Jinxi Xu and W. Bruce Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'99)*, pages 254–261, Berkeley, CA, USA, 1999. ACM.

[122] Yan Xu, Bin Wang, Jintao Li, and Hongfang Jing. An extended document frequency metric for feature selection in text categorization. In *Revised Selected Papers of the 4th Asia Infomation Retrieval Symposium (AIRS'08)*, pages 71–82, Harbin, China, January 15-18 2008. Springer.

[123] Praveen Yalagandula and Michael Dahlin. A scalable distributed information management system. In *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 379–390, Portland, OR, USA, August 30 - September 3 2004. ACM.

[124] ChengXiang Zhai. Statistical language models for information retrieval a critical review. *Foundations and Trends in Information Retrieval*, 2:137–213, 2008.

[125] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *Transactions on Information Systems*, 22:179–214, 2004.

[126] Jiangong Zhang and Torsten Suel. Efficient query evaluation on large textual collections in a peer-to-peer environment. In *Proceedings of the 5th IEEE International Conference on Peer-to-peer Computing (P2P'05)*, pages 225–233, Konstanz, Germany, August 31 - September 2 2005. IEEE Computer Society.

[127] Zheng Zhang, Shuming Shi, and Jing Zhu. SOMO: Self-organized metadata overlay for resource management in P2P DHT. In *Proceedings of the 2nd International Workshop on Peer-to-peer Systems (IPTPS'03)*, pages 170–182, Berkeley, CA, USA, February 21-22 2003. Springer.

[128] Christian Zimmer, Christos Tryfonopoulos, Klaus Berberich, Manolis Koubarakis, and Gerhard Weikum. Approximate information filtering in peer-to-peer networks. In *Proceedings of the 9th International Conference on Web Information Systems Engineering (WISE'08)*, pages 6–19, Auckland, New Zealand, September 1-3 2008. Springer.