ANOTAÇÕES SEMÂNTICAS EM CONSULTAS BASEADA NA INTENÇÃO DO USUÁRIO

RAFAEL GLATER DA CRUZ MACHADO

ANOTAÇÕES SEMÂNTICAS EM CONSULTAS BASEADA NA INTENÇÃO DO USUÁRIO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Rodrygo Luis Teodoro Santos
Coorientador: Nivio Ziviani

Belo Horizonte

Abril de 2017

RAFAEL GLATER DA CRUZ MACHADO

INTENT-AWARE SEMANTIC QUERY ANNOTATION

Dissertation presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais — Departamento de Ciência da Computação in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Rodrygo Luis Teodoro Santos
Co-Advisor: Nivio Ziviani

Belo Horizonte

April 2017

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Machado, Rafael Glater da Cruz.

M149i Inter-aware semantic query annotation / Rafael Glater da Cruz Machado. — Belo Horizonte, 2017.
xxii, 61 p. il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Rodrygo Luis Teodoro Santos
Coorientador: Nivio Ziviani

1. Computação – Teses. 2. Aprendizado de ranqueamento 3. Recuperação da informação. 4. Aprendizado de representações. 5. Busca semântica. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*73(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Intent-aware semantic query annotation

RAFAEL GLATER DA CRUZ MACHADO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

[illegible]

PROF. RODRYGO LUIS TEODORO SANTOS - Orientador
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. NIVIO ZIVIANI - Coorientador
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. ALTIGRAN SOARES DA SILVA
Departamento de Ciência da Computação - UFAM

[illegible]

PROF. MARCOS ANDRÉ GONÇALVES
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 07 de abril de 2017.

Acknowledgments

First of all, I would like to thank God for the opportunity to do a master's degree at a major university and learn from world-renowned people. During this work, I had the chance to meet amazing people for which I am very grateful.

In particular, I would like to thank my advisor, Rodrygo Santos, and my co-advisor Nivio Ziviani, who believed in me and agreed to guide me throughout this work. It is impossible to measure how much I learned from them.

I must express my very profound gratitude to my parents, whose love and guidance are with me in whatever I pursue. Also, to my brothers for friendship and sympathetic ear. You are always there for me.

Finally, I would like to express my gratitude to my colleagues from the LATIN laboratory, your friendship was essential during this journey.

“A journey of a thousand miles begins with a single step.”
(Lao-Tzu)

Resumo

O entendimento de uma consulta é uma tarefa desafiadora, principalmente devido à ambigüidade inerente da linguagem natural. Uma estratégia comum para melhorar a compreensão das consultas em linguagem natural é anotá-las com informações semânticas extraídas de uma base de conhecimento. No entanto, consultas com diferentes intenções podem se beneficiar de diferentes estratégias de anotação. Por exemplo, algumas consultas podem ser efetivamente anotadas com uma única entidade ou um atributo de entidade, outras podem ser melhor representadas por uma lista de entidades de um único tipo ou por entidades de vários tipos distintos, e outras podem ser simplesmente ambíguas. Nesta dissertação, propomos um framework para aprendizagem de anotações semânticas em consultas de acordo com a intenção existente em cada uma. Experimentos minuciosos em um benchmark publicamente disponível mostram que a abordagem proposta pode melhorar significativamente quando comparadas às abordagens agnósticas baseadas em campos aleatórios de Markov e de aprendizado de ranqueamento. Nossos resultados demonstram ainda, de forma consistente, a eficácia de nossa abordagem para consultas de várias intenções, comprimentos e níveis de dificuldade, bem como sua robustez ao ruído na detecção de intenção.

Palavras-chave: Aprendizado de ranqueamento, Recuperação da informação, Aprendizado de representações, Busca semântica, Anotação semântica em consultas.

Abstract

Query understanding is a challenging task primarily due to the inherent ambiguity of natural language. A common strategy for improving the understanding of natural language queries is to annotate them with semantic information mined from a knowledge base. Nevertheless, queries with different intents may arguably benefit from specialized annotation strategies. For instance, some queries could be effectively annotated with a single entity or an entity attribute, others could be better represented by a list of entities of a single type or by entities of multiple distinct types, and others may be simply ambiguous. In this dissertation, we propose a framework for learning semantic query annotations suitable to the target intent of each individual query. Thorough experiments on a publicly available benchmark show that our proposed approach can significantly improve state-of-the-art intent-agnostic approaches based on Markov random fields and learning to rank. Our results further demonstrate the consistent effectiveness of our approach for queries of various target intents, lengths, and difficulty levels, as well as its robustness to noise in intent detection.

Palavras-chave: Semantic Query Annotation, Learning to Rank, Intent-aware.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

A user's search query has traditionally been treated a short, underspecified representation of his or her information need [Jansen et al., 2000]. Despite the trend towards verbosity brought by the popularization of voice queries in modern mobile search and personal assistant interfaces [Guy, 2016], query understanding remains a challenging yet crucial task for the success of search systems. One particularly effective strategy for improving the understanding of a query is to annotate it with semantic information mined from a knowledge base, such as DBPedia.[1] In particular, previous analysis has shown that over 70% of all queries contain a semantic resource (a named entity, an entity type, relation, or attribute), whereas almost 60% have a semantic resource as their primary target [Pound et al., 2010].

1.1 Semantic Search

Semantic search is not limited to search on text resources, but also includes searching on images, audio, video and other types of resources. Since this thesis presents an approach for semantic search on text resources, we will limit the background discussion on it. Semantics is primarily the linguistic, and also philosophical, study of meaning. It deals with the relations between things — like words, phrases and symbols — and what they denote. From this definition, we can say that semantic search is the act of searching with meaning. For a better understanding of this concept, in the following, we make a comparison between lexical search (which does not consider the meaning of the query or of the retrieved information items) and semantic search.

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

1.1.1 Lexical Search vs. Semantic Search

Just over a decade ago, most search engines were still performing lexical searching. This means that, given a user query, they try to literally match the exact query words or a variant of them in the documents collection without worrying about what the whole query actually means, nor what the candidate documents say. To illustrate, consider the query "*stanford university*" inputted to a search engine. Arguably the homepage of the Stanford University[2] is a good match for this query. Its homepage have exactly these two terms in the title and probably in other sections, making the search engine just need to find the occurrence of these terms without understand what the words "*stanford*" and "*university*" means, nor what they mean together.

The lexical approach works well for the previous example and for other similar cases where the search engine only needs to literally match query terms, but now consider the query "*Tom Hanks movies where he plays a leading role*". This is an example of a longer and more complex query, where the lexical approach would not bring a satisfactory result. In this case, search engines need to "understand" what the user is expecting as an answer to show the proper results. That is why we call this type of query as a semantic search, because of the meaning inherent from it.

For humans, the intent of this query is clear: the user (probably) wants as a result, a list of movies starred by the actor Tom Hanks, more specifically, movies where his role was a leading one. Although, in this case, the query intent is (probably) clear, the relevance of the results is relative. The terms *leading role* can have different interpretations for different users, making the semantic search a challenging task. A search engine following the lexical approach, when trying to match terms like "Tom Hanks", probably would bring results directly related to the actor instead of the movies, even more difficult is to match the terms "*leading role*" in the page of the movies which he starred, because they would not be present.

For the correct comprehension of the query, modern search engines combine different techniques aiming to understand what the user is looking for. Among these techniques, we can highlight the query annotation task, which is the focus of this thesis, aiming to enrich a query with semantic resources, in order to help a search engine understand the user needs.

[2]`https://www.stanford.edu/`

1.2 Semantic Query Annotation

As previously discussed, the task of query annotation consists in generating a ranking of semantic resources associated to the query. Formally, given a query q and a knowledge base k, we use a function Ψ to extract candidate annotations, such that $\Psi(q, k) = A = \{a_1, a_2, ..., a_n\}$, where A is a set of ranked semantic annotations which were extracted from the knowledge base k, according to the query q.

State-of-the-art semantic query annotation approaches leverage features extracted from the descriptive content of candidate semantic resources (e.g., the various textual fields in the description of an entity [Zhiltsov et al., 2015; Nikolaev et al., 2016]) or their structural properties (e.g., related semantic resources [Tonon et al., 2012]) in a knowledge base. In common, these approaches treat every query uniformly, regardless of its target intent. By "intent", in this thesis, we mean the type (or class) of a query, like the traditional query classification, such as informational or navigational [Broder, 2002], instead of the synonym of "information need" in some contexts.

In contrast, we hypothesize that queries with different intents may benefit from specialized annotation strategies. For instance, some queries could be effectively annotated with a single entity (e.g., "*us president*") or an entity attribute (e.g., "*us president salary*"). Other queries could be better represented by a list of entities of a single type (e.g., "*us presidents*") or of mixed types (e.g., "*us foreign affairs*"). Finally, some queries may be simply ambiguous and demand annotations suitable for disambiguation (e.g., "*us*").

In this thesis, we propose a framework for learning semantic annotations suitable to the target intent of each individual query. Our framework comprises three main components: (i) intent-specific learning to rank, aimed to produce ranking models optimized for different intents; (ii) query intent classification, aimed to estimate the probability of each query conveying each possible intent; and (iii) intent-aware ranking adaptation, aimed to promote the most relevant annotations given the detected intents. To demonstrate the applicability of our framework, we experiment with a state-of-the-art learning to rank algorithm for intent-specific learning, multiple classification approaches for intent classification, and two adaptive strategies for annotation ranking. Thorough experiments using a publicly available semantic annotation test collection comprising queries with different intents show that our proposed framework is effective and significantly improves state-of-the-art intent-agnostic approaches from the literature. Moreover, a breakdown analysis further reveals the consistency of the observed gains for queries of various target intents, lengths, and difficulty levels, as well as the robustness of the framework to noise in intent detection.

1.3 Thesis Statement

The statement of this thesis is that the relevance of a semantic resource given a query depends on the intent underlying this query and, consequently, queries with different intents may benefit from different ranking models optimized for different intents. In particular, by optimizing a ranking model for a specific intent, each model will behave differently, promoting different annotations according to the intent which it was trained for. For instance, queries seeking a specific entity (e.g., "donald trump") should consider annotations related to the name of the entity, while queries seeking for entities of a single type (e.g., "us presidents") should consider annotations related to a common category in the knowledge base. This statement raises the following research questions, which will be answered in the upcoming chapters:

Q1. Do different intents benefit from different ranking models?

Q2. How accurately can we predict the intent of each query?

Q3. How effective is our semantic query annotation approach?

Q4. What queries are improved the most and the least?

1.4 Thesis Contributions

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

1. An intent-aware framework for learning semantic query annotations from structured knowledge bases.

   In this thesis we propose a framework for semantic query annotation that is sensible to the user's search intent. We detail each component of the framework and the tasks involved in instantiating them. Despite the particular instantiations chosen in this thesis, the framework is general and suitable for different methods.

2. An analysis of the specificity of several content and structural features for different query intents.

   Our approach uses multiple ranking models optimized specifically for different intent, using features based on textual content and also semantic features derived from the structure of the knowledge base. We analyze each model, investigating the most relevant features in each intent, correlating them and discussing their specificities.

3. A thorough validation of the proposed framework in terms of annotation effectiveness and robustness.

   We present experimental results validating the proposed approach when compared to state-of-the-art intent agnostic approaches for queries of various characteristics, including different query intents, lengths and difficulty levels, discussing success and failure cases.

1.5 Thesis Overview

The remainder of this thesis is organized as follows:

- Chapter 2 describes background material on semantic search and discusses related work on semantic query annotation, query intent classification and exploitation.
- Chapter 3 presents our proposed framework for learning semantic query annotations, describes its components and their instantiation. We present the algorithms used for intent detection, how the ranking models are trained and the strategies used to choose them according to the detected intent. We also propose content and structure based features to represent queries and semantic resources.
- Chapter 4 describes the experimental setup that supports the evaluation of the proposed approach, detailing the knowledge base, queries, relevance judgments and intent taxonomy used in our experiments. Baselines used for comparison and the procedure undertaken to train and test them as well as our own models are also described in this chapter.
- Chapter 5 presents the evaluation results of the proposed approach, evaluating the effectiveness of having a specific ranking model for each intent, the accuracy of intent classification and further evaluation for queries of different intents, sizes and difficulty levels.
- Chapter 6 concludes this thesis, summarizing the contributions and conclusions made along the chapters. Future work on the applicability of the proposed framework with different ranking algorithms and query intents is also presented.

Chapter 2

Background and Related Work

This chapter is divided in two parts. First, in Section 2.1 we provide a background discussion of basic tasks for semantic search used in this thesis, followed by an overview of related work on Section 2.2. In particular, in Section 2.2.1 we present related work on semantic query annotation using knowledge bases, and in Section 2.2.2 we discuss related attempts to exploit query intents in different search tasks.

2.1 Basic Tasks for Semantic Search

When searching on text resources, users express their intention through natural language text. In this section we discuss some basic techniques that are commonly (though not always) used by state-of-art approaches to deal with semantic search on text. Beyond theses techniques, we focus on Natural Language Processing (NLP), an research area and application that explores how computers can be used to understand and manipulate natural language text. Bellow, we list common NLP tasks:

Part-Of-Speech (POS) tagging. Consists in identify, from a pre-defined set, the grammatical role of each word of a sentence. Each pre-defined grammatical role have a tag to identify itself. Some typical POS tags are: NN (noun), VB (verb), adjective (JJ). POS-tagging is important to help in word sense disambiguation, which may vary according to the context. The word *table* could be a place where you sit, a 2-D layout of numbers and we can also have the expression "table a discussion". To illustrate, let us consider the sentence: *The sailor dogs the hatch.* After using POS-tagging we have:

The/DT sailor/NN dogs/VBZ the/DT hatch/NN

We can observe that the word "*dogs*" were tagged as a verb (VBZ), not as the more common plural noun. Semantically analyzing the sentence, is possible to infer that "*sailor*" and "*hatch*" implicate "*dogs*" as an action to the object "*hatch*" in the nautical context.

Shallow Parsing (or Chunking). The task of chunking is to identify and tag the basic constituents of a sentence, based on the POS-tagged words, and then linking them to higher order units that have discrete grammatical meanings. Some typical chunking tags are: NP (noun phrase), VB (verb phrase), ADJP (adjective phrase). A possible chunking of the example query from Section 1.1.1 is:

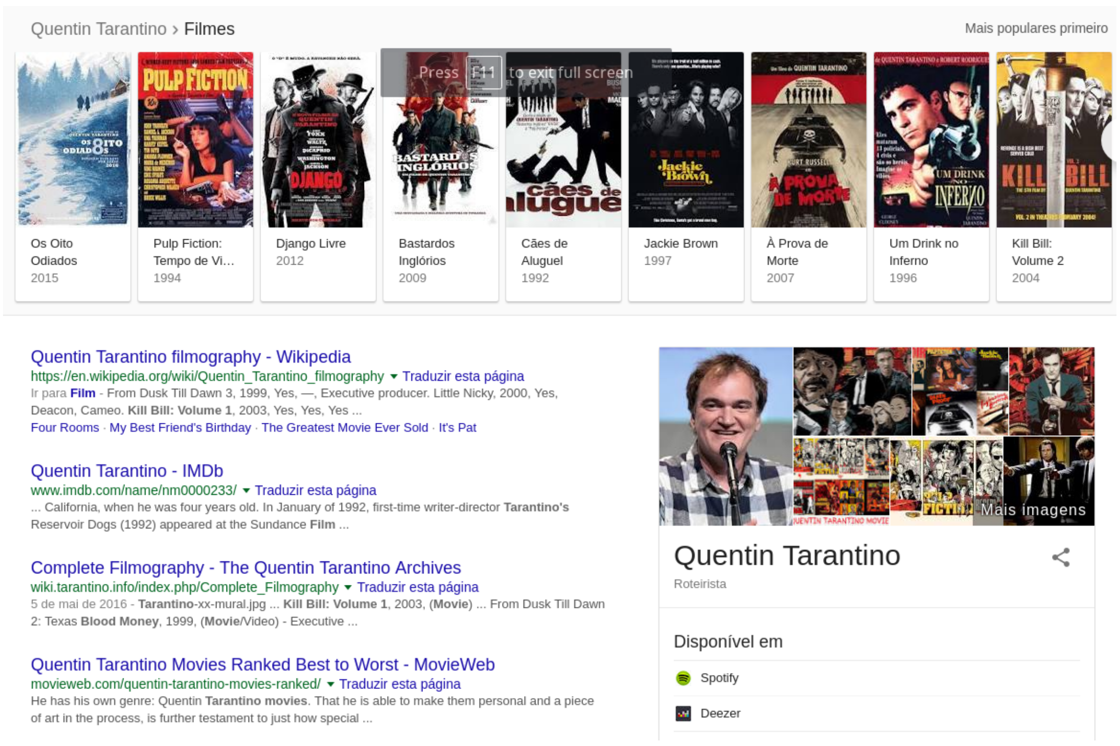
(**S**
  (**NP** Tom/NNP Hanks/NNP)
  (**NP** movies/NNS)
  where/WRB
  he/PRP
  (**VP** plays/VBZ)
  (**NP** a/DT leading/JJ role/NN)
)

The sentence (S) above, was chunked in noun phrases (NP) and verb phrases (VP). The words "*where*" and "*he*" did not fit this rule and were not tagged. The chunking task can be used as a first step for entity recognition and sentence parsing. For instance, the first noun phrase, formed by proper nouns, can indicate the presence of an entity. The verb phrase can indicate a relation between the entity, in the beginning of the sentence, and noun phrase after the verb.

Named-Entity Recognition (NER). Is the task of recognizing which word sequences from the text might refer to an entity and classify it into a pre-defined category, such as the name of persons, organizations, locations, etc. A named entity is real world object that can be described. It can be abstract (i.e., "Theory of relativity") or have a physical existence (i.e., "Albert Einstein"). In the example above (the query "*Tom Hanks movies where he plays a leading role*"), with NER, the words "*Tom Hanks*" can be recognized as a person, given signs that the query is related to him.

2.1.1 Raw Text and Structured Text

When searching on the web, the available documents are basically textual content written in natural language. To perform a lexical search, this type of data can be sufficient. But, to perform a semantic search, we need more structured data. In this section we discuss the importance of structured data for semantic search.


Quentin Tarantino › Filmes
Mais populares primeiro

Press F11 to exit full screen

Os Oito Odiados 2015
Pulp Fiction: Tempo de Vi… 1994
Django Livre 2012
Bastardos Inglórios 2009
Cães de Aluguel 1992
Jackie Brown 1997
À Prova de Morte 2007
Um Drink no Inferno 1996
Kill Bill: Volume 2 2004

Quentin Tarantino filmography - Wikipedia
https://en.wikipedia.org/wiki/Quentin_Tarantino_filmography ▾ Traduzir esta página
Ir para **Film** - From Dusk Till Dawn 3, 1999, Yes, —, Executive producer. Little Nicky, 2000, Yes, Deacon, Cameo. **Kill Bill: Volume 1**, 2003, Yes, Yes, Yes ...
Four Rooms · My Best Friend's Birthday · The Greatest Movie Ever Sold · It's Pat

Quentin Tarantino - IMDb
www.imdb.com/name/nm0000233/ ▾ Traduzir esta página
... California, when he was four years old. In January of 1992, first-time writer-director **Tarantino's** Reservoir Dogs (1992) appeared at the Sundance **Film** ...

Complete Filmography - The Quentin Tarantino Archives
wiki.tarantino.info/index.php/Complete_Filmography ▾ Traduzir esta página
5 de mai de 2016 - **Tarantino**-xx-mural.jpg ... **Kill Bill: Volume 1**, 2003, (**Movie**) ... From Dusk Till Dawn 2: Texas **Blood Money**, 1999, (**Movie**/Video) - Executive ...

Quentin Tarantino Movies Ranked Best to Worst - MovieWeb
movieweb.com/quentin-tarantino-movies-ranked/ ▾ Traduzir esta página
He has his own genre: Quentin **Tarantino movies**. That he is able to make them personal and a piece of art in the process, is further testament to just how special ...

Mais imagens

Quentin Tarantino
Roteirista

Disponível em
Spotify
Deezer

Figure 2.1: Hybrid results for the query "tarantino movies". Combining a traditional list of documents with a possible direct answer for this query.

Generally, semantic queries expect a more direct response, rather than a list of documents that possibly contain what the user is looking for. Modern search engines combine both types of results, expecting to directly show the answer to the user, without the need of clicking in the documents of the resulting list. Figure 2.1 illustrates an example of results combination from a search engine for the query "*tarantino movies*". From Figure 2.1 we can see a traditional list of documents, possible containing the answer for the query. We can also note semantic boxes, on the top and on the right. The box of the top have a list of movies directed or written by Quentin Tarantino, which seems to be the most appropriated answer to this query. The box in the right shows information about the person Quentin Tarantino, which is the central entity of the query. It is quite probable that these boxes already have what the user is looking for.

In Figure 2.1, the result for the query "*tarantino movies*" was assertive when returning a list of movies directed or written by Quentin Tarantino. However, there are other public figures whose last name is "Tarantino" (Giuseppe Tarantino, Javier Tarantino, Ray Tarantino, etc), how the search engine was able to realize that we were referring to Quentin Tarantino, writer and filme producer? This is possible with structured data, which allows to notice a strong relation between the entity "Quentin

Tarantino“ with several other entities of the type ”film“, which is a synonym of ”movie“, a term present in the query.

Structured data are intended to organize the information in a format that can be read by computers. A type of structured data are the knowledge bases, which represent a set of records in a database, which typically refer to some kind of ”knowledge“ about the world. By convention, records are stored in triples in the format *subject predicate object*. In The following, we present 6 records[1] extracted from DBPedia[2]:

```
<Quentin_Tarantino>    <type>           <Person>
<Quentin_Tarantino>    <placeOfBirth>   <Knoxville,_Tennessee>
<Reservoir_Dogs>       <writer>         <Quentin_Tarantino>
<Kill_Bill_Volume_2>   <writer>         <Quentin_Tarantino>
<Reservoir_Dogs>       <type>           <Film>
<Kill_Bill_Volume_2>   <type>           <Film>
```

A well structured knowledge base, always use the same identifiers to refer to the same type of information in different records. For instance, all records about the place of birth of a person, will use the predicate ”placeOfBirth“. That is the consistency between records that makes possible to find, for example, the films written by Quentin Tarantino, as illustrated by Figure 2.1. On the other hand, in a collection of raw text, written in natural language, text may not be orthographically or grammatically correct and the same information can be expressed in different forms by different authors, making it difficult to extract information. For example, the sentences ”Quentin Tarantino was born in Knoxville, Tennessee“, ”Tarantino is from Knoxville, TN“ and ”Knoxville is the place of birth of Quentin Tarantino“, all of them express the same information, which for humans is easy to realize, but not so easy for a machine.

Although the advantages provided by a knowledge base, to query on this type of data is need a specialized language, which is not friendly for common users. A common approach is to combine both types of data, allowing a traditional searching on text collections, but enriched by knowledge bases.

2.2 Related Work

In this section we present related work on semantic query annotation using knowledge bases and also related attempts to exploit query intents in different search tasks.

[1]For better understanding, we omitted the URL part from the triples, leaving just the names of the objects.

[2]Detailed information about DBPedia is given in Section 4.1

2.2.1 Semantic Query Annotation

Semantic search approaches [Bast et al., 2016] have been extensively researched in recent years, motivated by a series of related workshops and evaluation campaigns [de Vries et al., 2007; Balog et al., 2009; Alonso and Zaragoza, 2008]. While some research has been devoted to semantic search on the open Web [Bron et al., 2010; Santos et al., 2010b; Balog et al., 2009], particularly relevant to this paper are approaches focused on ranking semantic resources (e.g., named entities) mined from a structured domain, such as a knowledge base. The top ranked resources can be used directly to enrich a search engine's results page with structured semantic information [Bi et al., 2015] or indirectly to annotate the user's query for further processing for improved search quality.

Search in knowledge bases is typically performed using structured query languages such as SPARQL.[3] However, producing structured queries requires some expertise from the user, which limits the applicability of this approach in a broader scenario. To support unstructured querying, most previous semantic search approaches adapt traditional IR techniques to find, in the knowledge base, resources that match the user's query. For instance, some related works have used standard bag-of-words models, like BM25 [Balog and Neumayer, 2013; Tonon et al., 2012; Pérez-Agüera et al., 2010] and language models (LM) [Elbassuoni et al., 2009; Elbassuoni and Blanco, 2011; Neumayer et al., 2012; Zhiltsov and Agichtein, 2013; Herzig et al., 2013]. Extending traditional bag-of-words models, multi-fielded approaches have been proposed to appropriately weight information present in different fields describing a semantic resource. For instance, approaches based on BM25F [Blanco et al., 2011; Fetahu et al., 2015; Tonon et al., 2012; Pérez-Agüera et al., 2010; Blanco et al., 2010; Campinas et al., 2011] permit the combination of the BM25 scores of different fields into the final retrieval score. Multi-fielded approaches based on a mixture of language models have also been proposed [Ogilvie and Callan, 2003; Bron et al., 2013], which linearly combine query likelihood estimates obtained from multiple fields.

Also contrasting with bag-of-words models, recent approaches have exploited dependencies among query term occurrences in the descriptive content of a semantic resource. Building upon the framework of Markov random fields (MRF) [Metzler and Croft, 2005], these approaches construct a graph of dependencies among the query terms, which is used to estimate the relevance of each retrieved semantic resource. In particular, Zhiltsov et al. [2015] introduced a multi-fielded extension of MRF, called FSDM, which estimates the weight of each field with respect to three types of query

[3]https://www.w3.org/TR/rdf-sparql-query/

concept: unigram, ordered bigram, and unordered bigram. FSDM was later extended by Nikolaev et al. [2016], who proposed to estimate field weights with respect to individual query concepts. To cope with the explosive number of concepts (i.e., every possible unigram, ordered, and unordered bigram), they instead learn field weights with respect to a fixed set of concept features (e.g., the probability of occurrence of the concept in a field). In contrast to both of these approaches, we propose to learn the appropriateness of intent-specific feature-based ranking models for each individual query, by automatically predicting the target intent of this query. In Chapter 5, we compare our approach to FSDM as a representative of the current state-of-the-art.

In addition to exploiting the descriptive content of semantic resources, other researchers have adopted a hybrid approach [Bron et al., 2013; Tonon et al., 2012; Rocha et al., 2004; Elbassuoni et al., 2009; Herzig et al., 2013], leveraging structural properties of the knowledge base. In these approaches, an initial ranking of semantic resources is either re-ranked or expanded using the knowledge base structure to find related resources, which can be done through structured graph traversals [Tonon et al., 2012] or random walks [Rocha et al., 2004]. For instance, Tonon et al. [2012] exploited entities initially retrieved using BM25 as seeds in the graph from which related entities could be reached. Bron et al. [2013] proposed a method that makes a linear combination of the scores of a content-based approach using language models and a structure-based approach, which captures statistics from candidate entities represented according to their relations with other entities, expressed in RDF triples. Relatedly, Elbassuoni et al. [2009] proposed a language modeling approach to rank the results of exact, relaxed, and keyword-augmented graph-pattern queries over RDF triples into multiple subgraphs. The Kullback-Leibler divergence between the query language model and the language models induced by the resulting subgraphs was then used to produce the final ranking. While our main focus is on learning strategies rather than on specific features, to demonstrate the flexibility of our proposed framework, we exploit multiple structural properties of each semantic resource as additional features. In particular, these features are used for both detecting the intent of a query as well as for ranking semantic resources in response to this query.

2.2.2 Exploiting Query Intents

The intent underlying a user's search query has been subject of intense research in the context of web search. Broder [2002] proposed a well-known intent taxonomy, classifying web search queries into informational, navigational and transactional. Rose and Levinson [2004] later extended this taxonomy to consider more fine-grained classes.

In the context of semantic search, Pound et al. [2010] categorized queries into four major intents: entity queries, which target a single entity; type queries, which target multiple entities of a single type; attribute queries, which target values of a particular entity attribute; and relation queries, which aim to find how two or more entities or types are related. Entity queries and type queries accounted for more than 50% of a query log sampled in their study, whereas attribute and relation queries accounted for just over 5%. Other works focused on more specific intents, such as a question intent [Tsur et al., 2016], which targets answers to the question expressed in the query. In our experiments, we use an intent taxonomy comprising the three major classes described in these studies, namely, entity, type, and question queries, as well as an additional class including less represented intents, such as attribute and relation queries.

In addition to detecting query intents, several approaches have attempted to adapt the ranking produced for a query in light of some identified query property, such as its intent. For instance, Yom-Tov et al. [2005] proposed to adaptively expand a query depending on its predicted difficulty. Kang and Kim [2003] proposed to apply different hand-crafted ranking models for queries with a predicted informational, navigational, or transactional intent. However, such a hard intent classification may eventually harm the effectiveness of an adaptive approach, when queries of different intents benefit from a single ranking model [Craswell and Hawking, 2004]. To mitigate this effect, instance-based classification approaches have been used to identify similar queries (as opposed to queries with the same predicted intent) for training a ranking model. For example, Geng et al. [2008] resorted to nearest neighbor classification for building training sets for a given test query. Relatedly, Peng et al. [2010] proposed to estimate the benefit of multiple candidate ranking models for a given query by examining training queries that are affected by these models in a similar manner. In the context of search result diversification, Santos et al. proposed adaptive approaches for estimating the coverage of different query aspects given their predicted intent [Santos et al., 2011] as well as for estimating when to diversify given the predicted ambiguity of the query [Santos et al., 2010a]. Our proposed approach resembles these adaptive ranking approaches as we also resort to query intent classification as a trigger for ranking adaptation. Nonetheless, to the best of our knowledge, our approach is the first attempt to produce adaptive learning to rank models for a semantic search task.

2.3 Summary

In this chapter we presented background discussion about basic tasks for semantic search and a related work on semantic query annotations and query intent exploitation. In particular, in Section 2.1 we presented basic natural language tasks that generally supports semantic search on texts, including POS-tagging, named-entity recognition and shallow parsing. While, in Section 2.1.1, we presented how structured data can help on semantic search in contrast to raw text written in natural language.

In the second part of this chapter, on Section 2.2 we presented the related work. More specifically, in Section 2.2.1 we discussed the related work on semantic query annotation, presenting approaches that adapts traditional IR techniques to search on knowledge bases, we also present approaches based on Markov random fields, which considers the dependency between query terms and some variants using a multi-field representation of the resources, and finally, approaches adopting a hybrid solution which generates an initial ranking and then reorders it using information from knowledge bases. In Section 2.2.2, we discuss about related works on query intent exploitation, presenting approaches that propose different types of intents and approaches that adapts the resulting according to the detected intent.

In the next chapter we will present our proposed intent-aware framework for learning to rank semantic query annotations.

Chapter 3

Intent-Aware Ranking Adaptation for Semantic Query Annotation

Annotating queries with semantic information is an important step towards an improved query understanding [Alonso and Zaragoza, 2008]. Given a query, our goal is to automatically annotate it with semantic resources mined from a knowledge base, including named entities, attributes, relations, etc. For instance, the query "*us president*" could be annotated with arguably relevant semantic resources including "Donald Trump", "Federal Government", "White House." In this paper, we hypothesize that the relevance of a semantic resource given a query depends on the intent underlying this query. For the previous example, knowing that the query "*us president*" targets information around a single entity could promote alternative semantic resources including "Inauguration", "First 100 days", and "Controversies".

In this chapter, we propose an intent-aware framework for learning to rank semantic query annotations. In particular, we posit that the probability $P(r|q)$ that a given semantic resource r satisfies the user's query q should be estimated in light of the possible intents $i \in \mathcal{I}$ underlying this query. Formally, we define:

$$P(r|q) = \sum_{i \in \mathcal{I}} P(i|q)\, P(r|q,i), \tag{3.1}$$

where $P(i|q)$ is the probability that query q conveys an intent i, with $\sum_{i \in \mathcal{I}} P(i|q) = 1$, and $P(r|q,i)$ is the probability of observing semantic resource r given the query and this particular intent.

In Figure 3.1, we describe the three core components of our framework. In particular, the *query intent classification* and the *intent-specific learning to rank* components rely on supervised learning approaches to estimate $P(i|q)$ and $P(r|q,i)$, respectively, for each intent $i \in \mathcal{I}$. In turn, the *intent-aware ranking adaptation* component implements two alternative policies to suit the final ranking to the detected intents of each individual query.

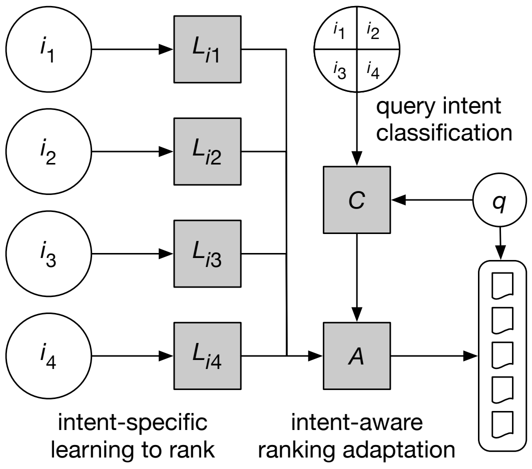

i_1
i_2
i_3
i_4
L_{i1}
L_{i2}
L_{i3}
L_{i4}
i_1 i_2 i_3 i_4
query intent classification
C
q
A
intent-specific learning to rank
intent-aware ranking adaptation

Figure 3.1: Intent-aware semantic query annotation. Each intent-specific ranking model L_i is learned on a query set comprising only queries with intent i. The query intent classification model C is learned on a set comprising queries of various intents. The intent-aware ranking adaptation strategy A uses the query intent classification outcome to decide on how to leverage the intent-specific ranking models.

3.1 Query Intent Classification

The first component of our framework is responsible for predicting the possible intents underlying a query [Brenes et al., 2009]. For this task, we adopt a standard multi-label classification approach. In particular, we aim to learn a query classification model $C : \mathcal{X} \rightarrow \mathcal{Y}$ mapping the input space $\mathcal{X}$ into the output space $\mathcal{Y}$. Our input space $\mathcal{X}$ comprises m learning instances $\{\vec{x}_j\}_{j=1}^m$, where $\vec{x}_j = \Phi(q_j)$ is a feature vector representation of query q_j as produced by a feature extractor Φ. In turn, our output space $\mathcal{Y}$ comprises m labels $\{y_j\}_{j=1}^m$, where y_j corresponds to one of the target intents $i \in \mathcal{I}$ assigned to query q_j by a human annotator. To learn an effective classifier C, we experiment with several classification algorithms in Section 5.2.

Table 3.1 presents the features we use to represent a query for intent classification. We use a total of 31 simple features, including both lexical as well as semantic ones. Lexical features like number of query terms and mean query term size can help detect, for example, natural language queries, which are usually longer than others. In addition, part-of-speech tags can help identify question queries, indicating the presence of wh-pronouns (e.g., what, where, why, when). Lastly, semantic features include the number of categories and number of ontology classes returned when using the query to search a knowledge base. Our intuition is that queries seeking for a specific entity will probably return fewer categories or ontology classes than queries seeking for a list of entities. For instance, the query "*eiffel*" returns only 5 categories, while the query "*list of films from the surrealist category*" returns more than 103,000.

Table 3.1: Query features for intent classification.

#	**Feature**	**Qty**
1	No. of query terms	1
2	Avg. query term size (in characters)	1
3	No. of matched categories in DBPedia	1
4	No. of matched ontology classes in DBPedia	1
5	No. of POS tags of different types	27
	TOTAL	31

3.2 Intent-Specific Learning to Rank

The second component of our framework aims to produce multiple ranking models, each one optimized for a specific query intent $i \in \mathcal{I}$. To this end, we resort to learning to rank [Liu et al., 2009]. Analogously to our query intent classification models in Section 3.1, our goal is to learn an intent-specific ranking model $L_i : \mathcal{V} \to \mathcal{W}$ mapping the input space $\mathcal{V}$ into the output space $\mathcal{W}$. Our input space includes n learning instances $\{\vec{V}_j\}_{j=1}^n$, where $\vec{V}_j = \Omega(q_j, \mathcal{R}_j)$ is a feature matrix representation (produced by some feature extractor Ω) of a sample of semantic resources $r \in \mathcal{R}_j$ retrieved for query q_j annotated with intent i. In our experiments, $\mathcal{R}_j$ is produced using BM25 [Robertson et al., 1995], although any unsupervised ranking technique could have been used for this purpose. Our output space $\mathcal{W}$ comprises n label vectors $\{\vec{W}_j\}_{j=1}^n$, where $\vec{W}_j$ provides relevance labels for each semantic resource $r \in \mathcal{R}_j$. To learn an effective ranking model L_i for each intent $i \in \mathcal{I}$, we use LambdaMART [Wu et al., 2008], which represents the current state-of-the-art in learning to rank [Chapelle and Chang, 2011].

Table 3.2 lists all 216 features used to represent each semantic resource $r \in \mathcal{R}_j$. Features #1-#6 are content-based features commonly used in the learning to rank literature [Liu et al., 2009], such as number of tokens, BM25, coordination level matching (CLM), TF, and IDF scores. These are computed in a total of 8 descriptive fields of r, such as name, attributes, categories (see Section 4.1 for a full description). Since TF and IDF are defined on a term-level, query-level scores are computed using multiple summary statistics (sum, min, max, avg, var). Finally, CLM, TF, IDF, and TF-IDF are computed for both unigrams and bigrams. Next, features #7-#14 are semantic features derived from a knowledge base. For instance, feature #7 indicates whether r is an entity directly mentioned in the query, while feature #8 considers the number of direct connections between r and all entities mentioned in the query. As an example of the latter feature, in the query "*songs composed by michael jackson*", the candidate resource "Thriller" will be directly related to the entity "Michael Jackson" (present in the query). For both features, we use DBPedia Spotlight[1] for entity recognition in queries. Features #9-#14 are query-independent features quantifying the connectivity of each candidate resource r with respect to other resources in the knowledge base (e.g., entities, categories, ontology classes).

Table 3.2: Semantic resource features for learning to rank. Features marked as 'Bi' are computed also for bigrams.

#	**Feature**	**Bi**	**Qty**
1	No. of tokens (per-field)		8
2	BM25 (per-field)		8
3	CLM (per-field)	✔	16
4	TF (per-field sum, min, max, avg, var)	✔	80
5	IDF (per-field sum)	✔	16
6	TF-IDF (per-field sum, min, max, avg, var)	✔	80
7	Matching entity		1
8	No. of direct relations with query entities		1
9	No. of matched relations with query terms		1
10	No. of inlinks		1
11	No. of outlinks		1
12	No. of linked ontology classes		1
13	No. of linked categories		1
14	No. of linked entities		1
	TOTAL		216

To keep our approach general, instead of handpicking features more likely to be useful for a particular intent, we use the same 216 available features when learning every intent-specific model L_i. To ensure that the learned model L_i is indeed optimized to its target intent i, intent-specific learning is achieved by using one training query set per intent, as illustrated in Figure 3.1.

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

3.3 Intent-Aware Ranking Adaptation

Sections 3.1 and 3.2 described supervised approaches for learning a query intent classification model C as well as multiple intent-specific ranking models L_i for all $i \in \mathcal{I}$. Importantly, all of these models are learned offline. When an unseen query q is submitted online, we must be able to return a ranking of semantic resources well suited to the target intent of q. Because we tackle query intent classification as a multi-label problem, we can actually estimate the probability $P(i|q)$ of different intents $i \in \mathcal{I}$ given the query q.

To exploit this possibility, we devise two strategies to adapt the ranking produced for a query q to the target intent(s) of this query. Our first strategy, called *intent-aware switching*, assigns each query a single intent, namely, the most likely one as predicted by the intent classification model C. For instance, for a target set of intents $\mathcal{I} = \{i_1, i_2, i_3\}$ of which i_1 is predicted as the most likely for q, we could instantiate Equation (3.1) with $P(i_1|q) = 1$, $P(i_2|q) = 0$, and $P(i_3|q) = 0$. As a result, only $P(r|q, i_1)$ (estimated via ranking model L_1) would have an impact on the final ranking, such that:

$$P(r|q) = P(r|q, i_1).$$

Some queries may have no clear winning intent. Other queries may prove simply difficult to classify correctly. To cope with uncertainty in intent classification, we propose a second ranking adaptation strategy, called *intent-aware mixing*. In this strategy, we use the full probability distribution over intents predicted by the classification model C to produce the final ranking for q. In the aforementioned example, suppose the predicted intent distribution is $P(i_1|q) = 0.7$, $P(i_2|q) = 0.2$, and $P(i_3|q) = 0.1$. Leveraging this distribution directly in Equation (3.1), we have a mixture of intent-specific ranking models contributing to the final ranking:

$$\begin{aligned} P(r|q) = & \ 0.7 \times P(r|q, i_1) \\ & + 0.2 \times P(r|q, i_2) \\ & + 0.1 \times P(r|q, i_3). \end{aligned}$$

To assess the effectiveness of our proposed intent-aware ranking adaptation strategies for semantic query annotation, in the next section, we compare these strategies to each other as well as to state-of-the-art intent-agnostic approaches from the literature.

3.4 Summary

In this chapter, we presented our intent-aware framework for learning to rank semantic query annotations, which posits that detecting the intent behind a given query and annotating it with semantic information extract from a knowledge base can lead better results compared to agnostic approaches. The proposed framework is composed by three core components, which were described in the sections of this chapter. In particular, in Section 3.1, we discussed the first component of the framework, presenting our approach for intent classification and the features used to represent a query. In Section 3.2, we discussed the second component, presenting our approach to produce multiple ranking models, each one optimized for a specific query intent and the list of features used to represent the semantic resources. Finally, in Section 3.3, we presented two strategies for ranking adaptation, *intent-aware switching*, which assigns each query a single intent and *intent-aware mixing*, which uses the full probability distribution over intents predicted by the classification model.

In the next chapter, we will present the experimental setup, used to conduct our evaluation experiments.

Chapter 4

Experimental Setup

In this chapter, we detail the experimental setup that supports the evaluation of our proposed intent-aware semantic query annotation approach introduced in Chapter 3. In particular, our experiments aim to answer the following research questions:

Q1. Do different intents benefit from different ranking models?

Q2. How accurately can we predict the intent of each query?

Q3. How effective is our semantic query annotation approach?

Q4. What queries are improved the most and the least?

In the following, we describe the knowledge base, queries, relevance judgments, and intent taxonomy used in our experiments. We also describe the baselines used for comparison and the procedure undertaken to train and test them as well as our own models.

4.1 Knowledge Base

The knowledge base used in our experiments is the English portion of DBPedia 3.7,[1] which comprises information extracted from Wikipedia dumps generated in late July 2011. DBPedia is available in the form of Resource Description Framework (RDF)[2] triples. RDF is a framework recommended by the World Wide Web Consortium (W3C)[3] for representing semantic information on the Web in the form of subject-predicate-object triples, where it is possible to describe additional information of a

[1]`http://wiki.dbpedia.org/data-set-37`
[2]https://www.w3.org/RDF/
[3]https://www.w3.org/

web resource (e.g. <Albert Einstein, *deathDate*, 1955-04-18>) or its relation with other resources (e.g. <Albert Einstein, *spouse*, Mileva Maric>).

Each resource on DBPedia belongs to one of three types of concepts: entities, categories and ontologies. Entities include all kinds of semantic resources, from real entities (e.g. Albert Einstein) to abstract concepts (e.g. Theory of Relativity), while categories represent a group of entities with something in common (e.g. Harry Potter Characters) and ontologies are classes with the aim of classifying entities in a formal and generic way (e.g. Person, Animal, Place). A resource is identified by a URL, which is prefixed according to the three aforementioned concepts: "http://dbpedia.org/resource/" for entities, "http://dbpedia.org/resource/Category:" for categories and "http://dbpedia.org/ontology/" for ontologies. This version of DBPedia contains information on more than 3.6 million entities organized in over 170,000 categories and 320 ontology classes in a 6-level deep hierarchy.

The information provided by the DBPedia dataset is available in different text files, where each line contains a RDF triple describing a piece of information, except the ontology file, which is a XML[4] file, listing and relating the hierarchy between each class (details are given bellow). The files are separated according to the information they contains, for instance, one file contains RDF triples relating an entity with a category it belongs, while other contains RDF triples relating an entity with its attribute information, and so on. In the following we list the files used in this thesis, describing and giving examples of each one:

- DBPedia Ontology: This file is the only one that is in a format other than RDF triples. It is formatted as a Web Ontology Language (OWL)[5], a pattern proposed by W3C, which is a XML listing and relating the hierarchy between ontology classes. In the example below, we present the definition of two classes: "British Royalty" (on line 1) and "Scientist" (on line 6).

```
1 <owl:Class rdf:about="http://dbpedia.org/ontology/BritishRoyalty">
2   <rdfs:subClassOf rdf:resource="http://dbpedia.org/ontology/Person"></rdfs:subClassOf>
3   <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"></rdfs:subClassOf>
4 </owl:Class>
5
6 <owl:Class rdf:about="http://dbpedia.org/ontology/Scientist">
7   <rdfs:subClassOf rdf:resource="http://dbpedia.org/ontology/Person"></rdfs:subClassOf>
8   <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl\#Thing"></rdfs:subClassOf>
9 </owl:Class>
```

[4]https://www.w3.org/XML/
[5]https://www.w3.org/OWL/

For each class definition, it is also listed from what classes it is derived (through the *subClassOf* element). In the example, “British Royalty” is a subclass of “Person”, which is a subclass of “Thing”, and “Scientist” is a subclass of “Person”, which is a subclass of “Thing”. This structure forms a hierarchy tree of ontologies classes, where the root note is the class “Thing”.

- Categories Label: A file containing RDF triples, listing all categories existing in the DBPedia collection, relating their URLs with their names. In the example below, we present RDF triples for four categories: “Princesses of Wales”, “British humanitarians”, “Daughters of British earls” and “Road accident deaths in France”. The first part of the triple is the category URL, followed by the predicate *label* (through the URL `http://www.w3.org/2000/01/rdf-schema#label` and the third part is the label string:

```
<http://dbpedia.org/resource/Category:Princesses_of_Wales>
<http://www.w3.org/2000/01/rdf-schema#label>
"Princesses of Wales"@en

<http://dbpedia.org/resource/Category:British_humanitarians>
<http://www.w3.org/2000/01/rdf-schema#label>
"British humanitarians"@en

<http://dbpedia.org/resource/Category:Daughters_of_British_earls>
<http://www.w3.org/2000/01/rdf-schema#label>
"Daughters of British earls"@en

<http://dbpedia.org/resource/Category:Road_accident_deaths_in_France>
<http://www.w3.org/2000/01/rdf-schema#label>
"Road accident deaths in France"@en
```

- Categories Skos: A file containing RDF triples using the Simple Knowledge Organization System (SKOS)[6], a W3C recommendation to represent structured vocabulary. In this file, categories are related to each other according to their subject specificity, indicating if a certain category is a broader or a narrow subject of other. In the example below, we present three RDF triples, indicating that the category “Princesses of Wales” has “British royal titles” and “British princesses by marriage” categories as a broader subject, while the category “British humanitarians” has the “Humanitarians” category as a broader subject. The first and third parts of the RDF triple are related categories URLs and the second is the predicate indicating the type of the relation, which can be *broader* or *narrow*:

```
<http://dbpedia.org/resource/Category:Princesses_of_Wales>
<http://www.w3.org/2004/02/skos/core#broader>
```

[6]`https://www.w3.org/2004/02/skos/`

```
<http://dbpedia.org/resource/Category:British_royal_titles>

<http://dbpedia.org/resource/Category:Princesses_of_Wales>
<http://www.w3.org/2004/02/skos/core#broader>
<http://dbpedia.org/resource/Category:British_princesses_by_marriage>

<http://dbpedia.org/resource/Category:British_humanitarians>
<http://www.w3.org/2004/02/skos/core#broader>
<http://dbpedia.org/resource/Category:Humanitarians>
```

- Entities Label: A file containing RDF triples, listing all entities existing in the DBPedia collection, relating their URLs with their names. In the example below, we present RDF triples for three entities: “Diana, Princess of Wales”, “Frances Shand Kydd” and “John Spencer, 8th Earl Spencer”. The first part of the triple is the entity URL, followed by the predicate *label*, indicating that it is being defined its label (through the URL `http://www.w3.org/2000/01/rdf-schema#label`, and the third part is the label string:

```
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>
<http://www.w3.org/2000/01/rdf-schema#label>
"Diana, Princess of Wales"@en

<http://dbpedia.org/resource/Frances_Shand_Kydd>
<http://www.w3.org/2000/01/rdf-schema#label>
"Frances Shand Kydd"@en

<http://dbpedia.org/resource/John_Spencer,_8th_Earl_Spencer>
<http://www.w3.org/2000/01/rdf-schema#label>
"John Spencer, 8th Earl Spencer"@en
```

- Extended Abstracts: A file containing RDF triples, defining the extended abstract of all entities. In the example below, we present a RDF triple of the entity “Diana, Princess of Wales”. The first part of the triple is the entity URL, followed by the predicate *abstract* (through the URL `http://dbpedia.org/ontology/abstract`) and the third part is a text of the extended abstract:

```
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>
<http://dbpedia.org/ontology/abstract>
"Diana, Princess of Wales (Diana Frances;nee Spencer; 1 July 1961 - 31 August 1997) was
    an international personality of the late 20th century as the first wife of Charles,
     Prince of Wales, whom she married on 29 July 1981. (...) From 1989, she was the
    president of Great Ormond Street Hospital for Children."@en
```

- Ontology Infobox Properties: A file containing RDF triples, defining attribute values for entities. This attributes are those which appears in the infoboxes of Wikipedia, and can be a raw value (like a date or population number) or can be another entity.

The example bellow sets four attributes for the entity “Diana, Princess of Wales”, her birth place, date and her parents. The first part of the triple is the entity for which the attribute is being set, the second part is the predicate indicating which attribute is and the third part is the attribute value, which can be a string or a URL to another entity:

```
<http://dbpedia.org/resource/Diana,_Princess_of_Wales> <http://dbpedia.org/ontology/birthPlace> <http://dbpedia.org/resource/Sandringham,_Norfolk>

<http://dbpedia.org/resource/Diana,_Princess_of_Wales> <http://dbpedia.org/ontology/birthDate> "1961-07-01"

<http://dbpedia.org/resource/Diana,_Princess_of_Wales> <http://dbpedia.org/ontology/parent> <http://dbpedia.org/resource/Frances_Shand_Kydd>

<http://dbpedia.org/resource/Diana,_Princess_of_Wales> <http://dbpedia.org/ontology/parent> <http://dbpedia.org/resource/John_Spencer,_8th_Earl_Spencer>
```

- Ontology Infobox Types: A file containing RDF triples relating a entity to a ontology class. The example bellow presents three RDF triples, classifying the entity “Diana, Princess of Wales” in three classes: “British Royalty”, “Person” and “Thing”. The first part of the triple is the entity URL that is being classified, the second part is the predicate *type* (through the URL `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`) and the third part is a URL of a ontology class:

```
<http://dbpedia.org/resource/Diana,_Princess_of_Wales> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/BritishRoyalty>

<http://dbpedia.org/resource/Diana,_Princess_of_Wales> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/Person>

<http://dbpedia.org/resource/Diana,_Princess_of_Wales> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Thing>
```

- Entities Categories: A file containing RDF triples relating a entity to a category. The example below presents four RDF triples, indicating that the entity “Diana, Princess of Wales” belongs to four categories: “Princess of Wales”, “British humanitarians”, “Daughters of British earls” and “Road accident deaths in France”. The first part of the triple is the entity URL that is being setting, the second part is the predicate *subject* (through the URL `http://purl.org/dc/terms/subject`) and the third part is a URL of a category:

```
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>
<http://purl.org/dc/terms/subject>
<http://dbpedia.org/resource/Category:Princesses_of_Wales>

<http://dbpedia.org/resource/Diana,_Princess_of_Wales>
<http://purl.org/dc/terms/subject>
<http://dbpedia.org/resource/Category:British_humanitarians>

<http://dbpedia.org/resource/Diana,_Princess_of_Wales>
<http://purl.org/dc/terms/subject>
<http://dbpedia.org/resource/Category:Daughters_of_British_earls>

<http://dbpedia.org/resource/Diana,_Princess_of_Wales>
<http://purl.org/dc/terms/subject>
<http://dbpedia.org/resource/Category:Road_accident_deaths_in_France>
```

- Redirects: A file containing RDF triples defining alternative URLs for the same entity. The example below presents four RDF triples, setting four different URLs for the entity "Diana, Princess of Wales". In this example, the entity "Diana, Princess of Wales" can also be referred as "Lady Di", "Lady Diana", "Diana Spencer" or "Princess Di", all the four URLs points to the same final URL. The first part of the triple is the alternative URL for the entity, the second part is the predicate *wikiPageRedirects* (through the URL `http://dbpedia.org/ontology/wikiPageRedirects`) and the third part is the entity URL to which it will be redirected:

```
<http://dbpedia.org/resource/Lady_Di>
<http://dbpedia.org/ontology/wikiPageRedirects>
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>

<http://dbpedia.org/resource/Lady_Diana>
<http://dbpedia.org/ontology/wikiPageRedirects>
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>

<http://dbpedia.org/resource/Diana_Spencer>
<http://dbpedia.org/ontology/wikiPageRedirects>
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>

<http://dbpedia.org/resource/Princess_Di>
<http://dbpedia.org/ontology/wikiPageRedirects>
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>
```

- Disambiguations: A file containing RDF triples disambiguating entities. Similar to the *Redirects* file, presents alternative URLs to the same entity, but the origin URL is ambiguous. The example below, presents three RDF triples indicating that the ambiguous names "Di", "Diana" and "Diana Spencer (disambiguation)" refer to the same entity “Diana, Princess of Wales”. The first part of the triple is the ambiguous entity URL, the second part is the predicate *wikiPageDisambiguates* (through the URL `http://dbpedia.org/ontology/wikiPageDisambiguates`) and the third part is the entity URL to which it disambiguates:

```
<http://dbpedia.org/resource/Di>↘
<http://dbpedia.org/ontology/wikiPageDisambiguates>↘
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>

<http://dbpedia.org/resource/Diana>↘
<http://dbpedia.org/ontology/wikiPageDisambiguates>↘
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>

<http://dbpedia.org/resource/Diana_Spencer_(disambiguation)> ↘
<http://dbpedia.org/ontology/wikiPageDisambiguates>↘
<http://dbpedia.org/resource/Diana,_Princess_of_Wales>
```

Each RDF triple on DBPedia can be viewed as a part of a massive graph, where entities, categories and ontologies are nodes and their semantic relation are edges. Building on the aforementioned examples of RDF triples around the entity *Diana, Princess of Wales*, Figure 4.1 illustrates how this triples, together, can be viewed as a graph. Entity nodes (represented as gray nodes) can be linked to other entities, categories (white) or ontologies (black) nodes. Each link between nodes, (represented as edges in Figure 4.1), have a label indicating the type of this relation. For instance, in Figure 4.1, the entity node *Diana, Princess of Wales* are linked to other entities, like *Sandringham, Norfolk*, which is, according to the edge label, her birth place. Her node is also linked to the ontology node *Person*, indicating this node is a person, and to some category nodes, like *Princess of Wales* and *Road accident deaths in France.*

To create the knowledge base for our experiments, we parsed all this files and created two linked structures: a graph (like illustrated in Figure 4.1) to explore the semantic relation between resources and a textual content index to help while searching for query terms. The graph were built upon the Titan 0.5.4[7] and the textual content were indexed using Elasticsearch 1.7.5.[8]

To build the graph, first we parsed the files *Entities Label*, *Categories Label* and *DBPedia Ontology* to create the nodes for all entities, categories and ontologies existing in the DBPedia collection, respectively. The other files were parsed to build the edges. The file *Categories Skos* were parsed to link the category nodes to each other, while the *Ontology Infobox Properties* were parsed to link entity nodes to each other. The

[7]`http://titan.thinkaurelius.com`

[8]`https://www.elastic.co/products/elasticsearch`

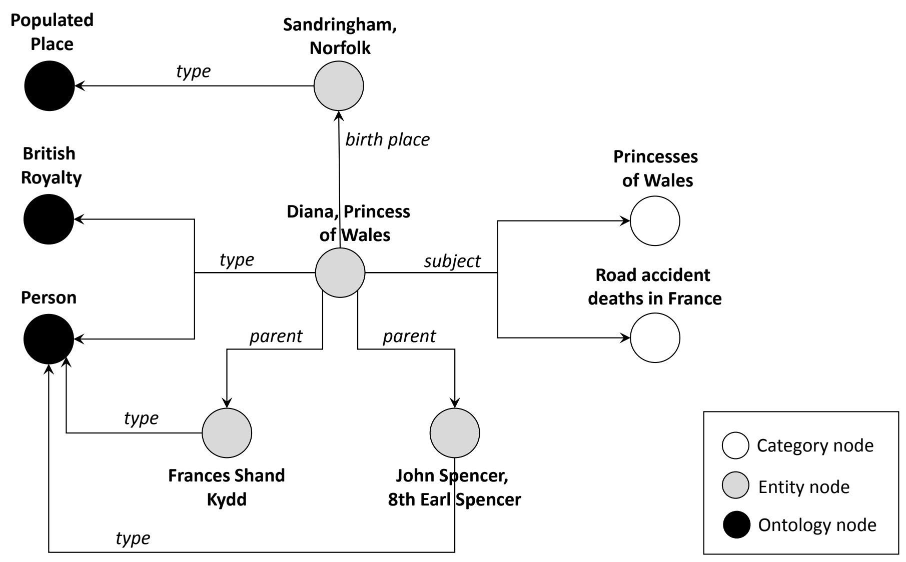


Figure 4.1: Part of a knowledge base graph built from RDF triples.

file *Ontology Infobox Types* were parsed to link entity nodes with ontology nodes and the file *Entities Categories* were parsed to link entity nodes with category nodes.

For the textual content, we created a fielded content representation as proposed by Zhiltsov et al. [2015], dividing the entity content in different fields: *Names*, *Attributes*, *Categories*, *Similar entity names* and *Related entity names* fields. In addition, we included three other fields: *Ontology classes*, *URL* and *All*. Each field has a different meaning and we built them parsing different files, as described below:

- Names: All names of an entity parsed from the *Entities Label* file.

- Attributes: Information describing the entity, parsed from the *Extended Abstract* and *Infobox Properties* file, extracting all textual properties that is not another entity, e.g. birth dates from persons, population from places.

- Categories: Name of categories from the *Category Labels* file, from categories to which the entity belongs to, listed in the *Categories Entities* file.

- Similar entity names: Alternative names to the same entity (e.g. "Lady Di", "Diana Spencer", "Princess Di"), parsed from the *Redirects* and *Disambiguations* files.

- Related entity names: Entities names that are directly related to the given entity with the predicate of this relation, e.g. "birth place Sandringham, Norfolk", built from the *Infobox Properties* file, considering only the triples where the attribute value is a valid entity.

- Ontology classes: Class names of ontologies to which the entity belongs, parsed from *Ontology Infobox Types* file.
- URL: The entity URL parsed from the *Entities Label* file.
- All: Special field concatenating the available content from all fields above.

Table 4.1 illustrates an example of a fielded representation of the entity "Diana, Princess of Wales". Indexed terms were lower-cased, stemmed using Krovetz [1993] stemmer and stopwords were removed. From Table 4.1, it follows that, each field is a concatenation of terms parsed from the files discussed above. For instance, we can observe that the field *Similar entity names* is composed by the names found in the *Redirects* and *Disambiguations* files and the *Ontology Classes* field is a concatenation of the three classes, "British Royalty", "Person" and "Thing". The same goes for the other fields.

Table 4.1: Multi-fielded entity representation for Princess Diana

Field	Content
Names	Diana, Princess of Wales
Attributes	Diana, Princess of Wales (Diana Frances; nÃ©e Spencer; 1 July 1961 - 31 August 1997) was an international personality of the late 20th century (...). death Date 1997-08-31 birth Date 1961-07-01 title Princess
Categories	Princesses of Wales British humanitarians Daughters of British earls English Anglicans Mine action Mountbatten-Windsor family Road accident deaths in France Spencer-Churchill family 1961 births 1997 deaths (...)
Similar entity names	Diana Spencer, Princess of Wales; Lady Diana Spencer; Diana Princess of Wales; Princess Di; Lady Di; (...)
Related entity names	resting Place Althorp resting Place Northamptonshire death Place Pitie-Salpetriere Hospital birth Place Sandringham, Norfolk parent Frances Shand Kydd parent John Spencer, 8th Earl Spencer
Ontology Classes	British Royalty Person Thing
URL	http://dbpedia.org/resource/Diana,_Princess_of_Wales
All	Omitted. (All content above concatenated)

4.2 Queries, Relevance Judgments, and Intents

We use a publicly available benchmark[9] built on top of DBPedia 3.7, which comprises a total of 485 queries from past semantic search evaluation campaigns [Balog and

[9]`http://bit.ly/dbpedia-entity`

Neumayer, 2013]. In total, there are 13,090 positive relevance judgments available. While some of these include graded labels, for a homogeneous treatment of all queries, we consider relevance as binary. Each evaluation campaign has distinct characteristics, comprising queries from different types, as described bellow:

- INEX-XER: The INitiative for the Evaluation of XML retrieval (INEX) organizes a XML Entity Ranking track (INEX-XER) to provide a forum where researchers may compare and evaluate techniques for engines that return lists of entities. The INEX-XER 2009 track seeks a list of entities, where entities are represented by their Wikipedia/DBPedia page.

- TREC Entity: The Text REtrieval Conference (TREC), from 2009, organizes a related entity finding task from the Entity Track, which provides queries focused on specific relationship between entities.

- SemSearch ES: Semantic Search Challenge (SemSearch) in the World Wide Web Conference, from 2010 and 2011, establishes an academic competition for the best systems that can answer a number of queries that are focused on the task of Entity Search (ES). This task provides short keyword queries seeking for a particular entity.

- SemSearch LS: List Search (LS) is the second task of the Semantic Search Challenge (SemSearch) which provides more complex queries expecting multiple entities as answer.

- QALD-2: The Question Answering over Linked Data (QALD) challenge, 2012 edition, aims at providing an benchmark for assessing and comparing state-of-the-art-systems that mediate between a user, expressing his or her information need in natural language, and linked data sources.

- INEX-LD: The INEX 2012 evaluation campaign is consisted of a five tracks: Linked Data, Relevance Feedback, Snippet Retrieval, Social Book Search, and Tweet Contextualization. Query collection were selected from the Ad-hoc Search Task of the Linked Data (LD) track and consists of keyword-style queries with different intents.

Queries from these different evaluation campaigns forms a benchmark with a wide variety of query intents, including entity, type, relation and attribute queries, as well as queries with a question intent. Following past research [Balog and Neumayer, 2013; Zhiltsov et al., 2015; Nikolaev et al., 2016], we organize these queries into four intent-specific query sets, the salient statistics of which are described in Table 4.2:

- *E*: Short keyword queries taken from SemSearch ES collection referring to a particular entity from different classes, like places (e.g., "*orlando florida*"), real persons (e.g., "*jack johnson*"), fictional characters (e.g., "*harry potter*"), companies (e.g., "*pizza populous detroit mi*") and many others. Some queries are ambiguous (e.g. "*ben frankling*" which can be a person or a ship);
- *T*: Keyword queries from three different collections (INEX-XER, TREC Entity, SemSearch LS) expecting a list of entities of a certain type, some of them are generic (e.g., "*continents in the world*"), others more restrictive (e.g. "*movies with eight or more academy awards*");
- *Q*: Natural language question queries from QALD-2 collection. This collection includes queries seeking for a specific entity (e.g. "*who is the husband of Amanda Palmer?*"), while others expects a list of entities with a simple request (e.g. "*Give me a list of all American inventions*") and others are more complex (e.g. "*Which daughters of British earls died in the same place they were born in?*");
- *O*: Queries with different intents from INEX LD collection, including queries searching a particular entity (e.g. "*invented telescope*"), entities from certain type (e.g. "*tango music composers*"), queries relating two or more entities (e.g. "*Nelson Mandela John Dube*") and queries searching for attribute information of a particular entity (e.g. "*Sweden Iceland currency*").

We decided to consider these four sets of queries as intents due to the fact that they were provided by different benchmark campaigns, focusing in different search tasks and with distinct revelance judgment strategies. To check that this division makes sense, Figure 4.2 illustrates the distribution of query types throughout the four query groups we are using (E, T, Q, O). To this end, each query was manually labelled by three different people into one of the query types proposed by Pound et al. [2010] (Entity, Type, Attribute, Relation, Other).

From Figure 4.2, it follows that group E, which has queries seeking for a specific entity, has 84% of Entity queries, 9% of Type queries, 1% of Attribute queries, 3% of Relation queries and 3% of Other queries. Group T, which has queries seeking for entities of a specific type, has 86% of Type queries, 3% of Entity queries, 6% of Attribute queries, 4% of Relation queries and 1% of other queries. Group Q, which has natural language questions, mostly has Entity and Type queries, with 35% and 54%

respectively, followed by 7% of Attribute queries, 4% of Relation queries and 0% of Other queries. Group O, comprising queries from different aspects, is the most diverse group, with 28% of Entity queries, 43% of Type queries, 1% of Attribute queries, 14% of Relation queries and 14% of Other queries.

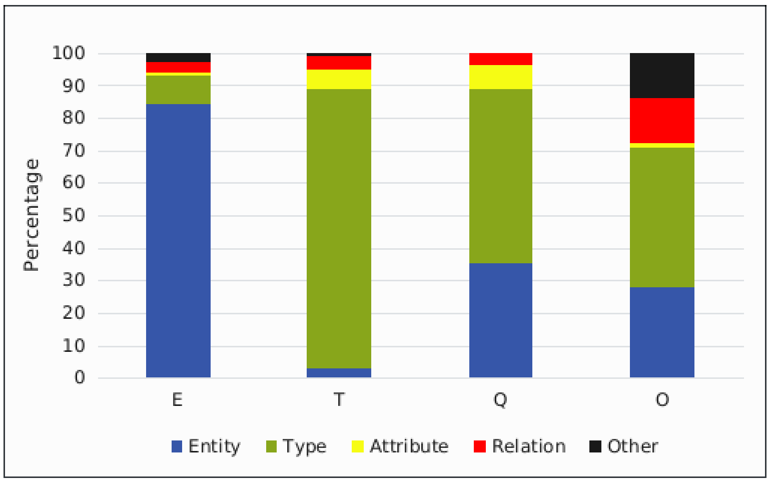


Figure 4.2: Query types distribution over each intent group

Table 4.2: Statistics of the intent-specific query sets used in our evaluation. Length and qrels denote per-query averages of query length and positive judgments in each set.

Set	**Campaign [Balog and Neumayer, 2013]**	**Queries**	**Length**	**Qrels**
E	SemSearch ES	130	2.7	8.7
T	INEX-XER, SemSearch LS, TREC Entity	115	5.8	18.4
Q	QALD-2	140	7.9	41.5
O	INEX-LD	100	4.8	37.6
TOTAL		485	5.3	26.55

Continuing the analysis of each query group, from Table 4.2, it follows that, besides the semantic diversity of each set, this collection of queries is also diverse in terms of length and number of relevant results. The Q set has the largest queries, with an average of almost 8 terms per query, and the largest number of relevant results, with an average of 41 relevant entities per query. On the other hand, the E set, has the shortest queries (2.7 terms on average) and the smallest number of relevant (8.7 entities). The set with most queries is the Q set, with 140 queries, while the O set has the least number, with 100 queries.

4.3 Retrieval Baselines

We compare our approach to multiple intent-agnostic baselines from the literature. As a vanilla ad-hoc search baseline, we consider BM25 with standard parameter settings ($k_1 = 1.2$, $b = 0.8$). To assess the effectiveness of our intent-aware ranking adaptation strategies introduced in Section 3.3, we further contrast them to two intent-agnostic strategies, which consistently apply a single ranking model for all queries, regardless of their target intent. As illustrated in Table 4.3, the *fixed* strategy applies a model L_i learned on one intent-specific query set, whereas the *oblivious* strategy applies a model L_R learned on a set of random queries. For a fair comparison, both of these baseline strategies as well as our own intent-aware switching and mixing strategies use the same learning algorithm (LambdaMART) and ranking features (all 216 features in Table 3.2). Lastly, we further contrast our approach to FSDM [Zhiltsov et al., 2015] (see Section 2.2.1) as a representative of the current state-of-the-art.

Table 4.3: Example application of intent-agnostic (baseline) and intent-aware ranking adaptation strategies.

	intent-agnostic					intent-aware	
i	fixed-E	fixed-T	fixed-Q	fixed-O	oblivious	switching	mixing
E	L_E	L_T	L_Q	L_O	L_R	L_E	$\sum_i w_i L_i$
T	L_E	L_T	L_Q	L_O	L_R	L_T	$\sum_i w_i L_i$
Q	L_E	L_T	L_Q	L_O	L_R	L_Q	$\sum_i w_i L_i$
O	L_E	L_T	L_Q	L_O	L_R	L_O	$\sum_i w_i L_i$

4.4 Training and Test Procedure

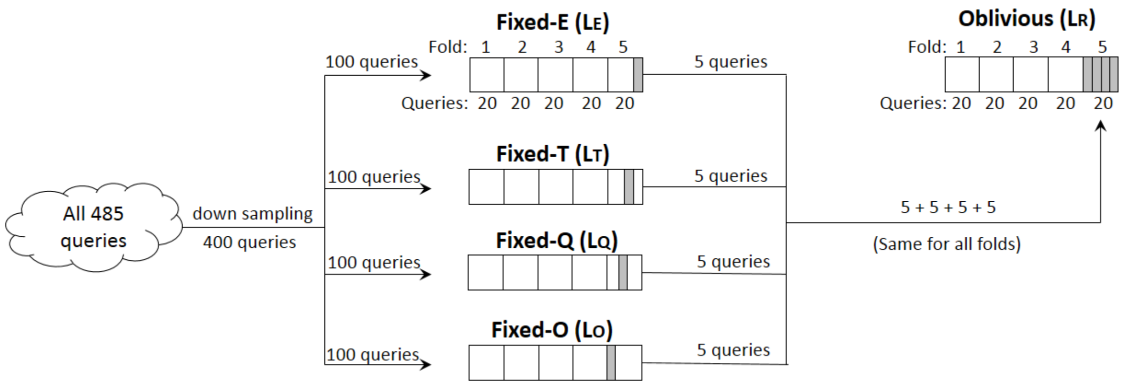


Figure 4.3: Division of the data for a 5-fold cross-validation of each model.

Figure 4.3 describes the division of the training and test folds for each model. For a fair comparison between our intent-aware semantic query annotation approach and the intent-agnostic baselines described in Section 4.3, we randomly downsample all query sets in Table 4.2 until they reach 100 queries each (i.e., the number of queries in the smallest query set). This ensures the learning process is not biased towards any particular intent. To learn an intent-specific model L_i for each intent $i \in \mathcal{I} = \{E, T, Q, O\}$, we perform a 5-fold cross validation in the corresponding query set from Table 4.2. For the oblivious strategy, the intent-agnostic model L_R is also learned via 5-fold cross validation on a set of 100 queries sampled uniformly at random from the four intent-specific query sets after downsampling. This multi-intent query set is also used to tune the parameters of FSDM [Zhiltsov et al., 2015] for different concepts (unigrams, ordered and unordered bigrams) and each of the fields listed in Section 4.1. The weights for unigrams, ordered bigrams and unordered bigrams are equal to 0.68, 0.18 and 0.14 respectively. Weights for each field for the three concepts are shown in Table 4.4. In each cross-validation round, we use three partitions (60 queries) for training, one for validation (20 queries), and one (20 queries) for testing.

Learning to rank is performed using the LambdaMART implementation in RankLib 2.7,[10] optimizing for normalized discounted cumulative gain at the top 100 results (nDCG@100). LambdaMART is deployed with default hyperparameter settings,[11] with 1,000 trees with 10 leaves each, minimum leaf support 1, unlimited threshold candidates for tree splitting, learning rate 0.1, and early stopping after 100 non-improving iterations. All results are reported as averages of all test queries across the five cross-validation rounds. In particular, we report nDCG@10, precision at 10 (P@10), and mean average precision (MAP). All evaluation metrics are calculated on the top 100 results returned by each approach. To check for statistically significant differences among them, we use a two-tailed paired t-test and write △ (▽) and ▲ (▼) to denote significant increases (decreases) at the 0.05 and 0.01 levels, respectively. A further symbol ∘ is used to denote no significant difference.

4.5 Summary

This chapter introduced the research questions inherent to the proposed method and gave details of the experimental setup aimed to answer them. In Section 4.1, we presented the knowledge base indexed from DBPedia and how we parsed its RDF

[10]`https://sourceforge.net/p/lemur/wiki/RankLib%20How%20to%20use/`

[11]Hyperparameter tuning on validation data showed no significant improvements in our preliminary tests.

Table 4.4: Tunned fields weight for each query concept for the FSDM approach.

	Fields				
Query Concept	Names	Attributes	Categories	Similar Entities	Related Entities
Unigrams	0.11	0.37	0.20	0.17	0.15
Ordered Bigrams	0.18	0.28	0.18	0.24	0.12
Unordered Bigrams	0.24	0.35	0.09	0.14	0.18

triples to create a fielded representation of entities and the relation between them. We also presented, in Section 4.2, the test collection composed by a set of queries from different evaluation benchmarkings. Each set comprises queries with different characteristics that we considered as search intents, totaling 4 types: entity queries, type queries, question queries and other queries (like attribute queries and relation queries). In Section 4.3, we presented the intent-agnostic baseline methods used to make a comparison evaluation. We used the BM25 as a vanilla baseline and FSDM as a state-of-the-art one. We also used an agnostic learning-to-rank model in contrast to our intent-aware ranking adaptation strategies also based on learning-to-rank models. Finally, in Section 4.4, we detailed the training and test procedure using 5-fold cross-validation and how we separated data for each fold for each query type to guarantee a fair comparison between all models. Chapter 5 will present the experimental results answering the research questions stated in this chapter.

Chapter 5

Experimental Evaluation

In this chapter, we empirically evaluate our approach in order to answer the four research questions stated in Chapter 4. First, in Section 5.1 we investigate the effectiveness of using different ranking models for different query intents. Then, in Section 5.2, we test the accuracy in predicting the intent behind queries and the robustness to noise in prediction. In Section 5.3, we present the effectiveness of our approach in contrast to different baselines and, in Section 5.4, we present a further analysis for queries with different intents, lengths, and difficulty. Finally, in Section 5.5, we discuss some successful and failure cases where the intent-awareness model can contribute to a better result when compared to the agnostic model.

5.1 Intent Specificity

The core hypothesis of our proposal is that different queries may benefit from a ranking model optimized to their intent. To verify this hypothesis, we address *Q1*, by assessing the specificity of ranking models optimized to the four intents described in Table 4.2. To this end, Figure 5.1 correlates the importance assigned to all 216 features by each intent-specific ranking model L_i, for $i \in \mathcal{I} = \{E, T, Q, O\}$. Feature importance is quantified using the least square improvement criterion proposed by Lucchese et al. [2015] for gradient boosted regression tree learners, such as LambdaMART. This method is based on the original work on gradient boosted regression trees from Friedman [2001]. For each tree of the model, for each feature a measure similar to the least square improvement measure proposed by Friedman [2001] is computed. Since each tree split node improves the objective function, the total gain for a feature can be estimated by summing up the gains across all the split nodes for all trees where the feature is used. From Figure 5.1, we observe a generally low correlation ($\rho < 0.5$) between models,

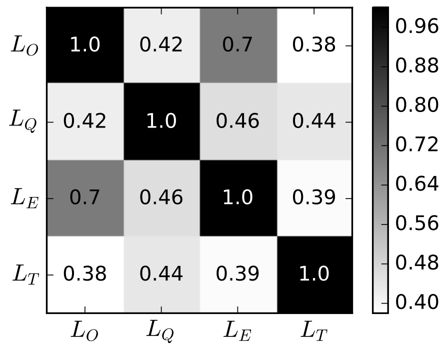


Figure 5.1: Spearman's correlation coefficient for feature importance across pairs of intent-specific ranking models.

except for the L_E and L_O models, with $\rho \approx 0.7$.

Table 5.1 lists the five most important features for each intent-specific model. The entity-oriented L_E model gives importance to features related to the occurrence of bigrams in the name and similar entities fields. For instance, the query "*martin luther king*" expects semantic resources named "Martin Luther King III" and "Martin Luther King High School." The type-oriented L_T model considers a variety of distinct features, two features related to the categories field are present in the top 5, which are useful for queries like "*state capitals of the united states of america.*" The question-oriented L_Q model gives importance to features describing the relation between entities and ontology classes, derived from both content fields as well as the graph structure underlying the knowledge base. These can help to identify relevant resources linked to an entity in the query through qualified relations, as in the query "*who was the successor of john f. kennedy?*" Lastly, the L_O model, which is optimized on a set comprising queries of various intents, strongly favors content-based features, which are arguably effective for broad queries such as "*einstein relativity theory.*" Recalling question *Q1*, these results provide a strong indication of the specificity of different models to queries of different intents.

Since each ranking model gives importance to different features, the next experiment aims to check whether, given a query intent, the best ranking model is in fact the one which it was trained for. Table 5.2 compares the performance of queries from a specific intent when applied to each existing ranking model. From Table 5.2, it follows that the ranking model that best performs for each query intent is the one corresponding to the same intent. We can also note that the L_T model is the most balanced one, performing reasonably well for all intents. This can be explained because, as we already discussed above, the L_T model gives importance to a variety of distinct features, and probably generalizes better to different types of intent.

Table 5.1: Top 5 features per ranking model.

	#	**Feature**
	1	TF-IDF sum of bigrams in *similar entities*
	2	Matching entity
L_E	3	TF sum of bigrams in *similar entities*
	4	TF avg of bigrams in *similar entities*
	5	TF-IDF max of bigrams in *similar entities*
	1	CLM in *categories*
	2	CLM in *all content*
L_T	3	No. of inlinks
	4	No. of tokens in *similar entities*
	5	TF-IDF sum of bigrams in *categories*
	1	BM25 in *ontology classes*
	2	No. of matched relations with query terms
L_Q	3	No. of direct relations with query entities
	4	No. of inlinks
	5	TF-IDF max of unigrams in *ontology classes*
	1	TF sum of bigrams in *name*
	2	BM25 in *name*
L_O	3	TF-IDF max of unigrams in *categories*
	4	TF-IDF max of bigrams in *name*
	5	TF-IDF var of bigrams in *all content*

Intent	**Model**	**P@10**	**nDCG@10**	**MAP**
	L_O	**0.259**	**0.303**	**0.113**
O	L_T	0.237°	0.274°	0.102°
	L_Q	$0.140^{\blacktriangledown}$	$0.157^{\blacktriangledown}$	$0.056^{\blacktriangledown}$
	L_E	$0.236^{\triangledown}$	$0.268^{\blacktriangledown}$	0.100°
	L_O	$0.202^{\blacktriangledown}$	$0.211^{\blacktriangledown}$	$0.148^{\blacktriangledown}$
T	L_T	**0.289**	**0.327**	**0.218**
	L_Q	$0.195^{\blacktriangledown}$	$0.215^{\blacktriangledown}$	$0.137^{\blacktriangledown}$
	L_E	$0.146^{\blacktriangledown}$	$0.139^{\blacktriangledown}$	$0.111^{\blacktriangledown}$
	L_O	$0.045^{\blacktriangledown}$	$0.079^{\blacktriangledown}$	$0.061^{\blacktriangledown}$
Q	L_T	$0.104^{\blacktriangledown}$	$0.198^{\blacktriangledown}$	$0.144^{\triangledown}$
	L_Q	**0.143**	**0.273**	**0.202**
	L_E	$0.038^{\blacktriangledown}$	$0.070^{\blacktriangledown}$	$0.049^{\blacktriangledown}$
	L_O	$0.245^{\blacktriangledown}$	$0.445^{\blacktriangledown}$	$0.329^{\blacktriangledown}$
E	L_T	$0.177^{\blacktriangledown}$	$0.298^{\blacktriangledown}$	$0.224^{\blacktriangledown}$
	L_Q	$0.131^{\blacktriangledown}$	$0.213^{\blacktriangledown}$	$0.158^{\blacktriangledown}$
	L_E	**0.293**	**0.498**	**0.386**

Table 5.2: Performance of each query intent when applied to specific ranking models.

5.2 Intent Classification Accuracy

The results in the previous experiment suggest that exploiting the specificity of different query intents may result in more effective ranking models. Before investigating whether this is indeed the case, in this section, we address *Q2*, with the aim of establishing what level of query intent detection accuracy can be attained in practice. To this end, we experiment with a range of traditional classification algorithms implemented in Scikit-learn 0.17.1,[1] optimized via 5-fold cross validation using the same partitions leveraged for learning to rank, as detailed in Section 4.4. Table 5.3 reports intent classification accuracy averaged across test queries in all cross-validation rounds. As shown in the table, according to a two-tailed paired t-test over the accuracy of each fold, all algorithms (except AdaBoost) are statistically equivalent. We choose to use Stochastic Gradient Descent (SGD) with a log loss, performing an incremental logistic regression as the intent classifier in the remainder of our experiments.

Table 5.3: Query intent classification accuracy.

Algorithm	**Accuracy**
AdaBoost	0.670▼
Support Vector Machines	0.740°
Gradient Boosting	0.757°
Bagging	0.760°
Random Forest	0.765°
Logistic Regression	**0.770**

Table 5.4: Confusion matrix for the intent classification.

		Predicted Intent			
		O	**Q**	**E**	**T**
Original Intent	**O**	**71**	0	25	4
	Q	0	**94**	0	6
	E	20	0	**77**	3
	T	15	14	5	**66**

According to Table 5.3, the chosen classifier has an accuracy of 77%, which raises the question of which intent is mistakenly identified in the 33% of the remaining queries. Table 5.4 illustrates the confusion matrix relating the original intent of the queries with the intent identified by the classifier. From Table 5.4 it follows that, from the 100 queries in group O, 71 are correctly classified, 25 are mistakenly classified as E and 4 as T. Queries from group Q are the most correct, with 94 correctly classified and missing only 6 as T. Group E, 77 are correctly classified, 20 are mistakenly classified as O and 3 as T. Queries from group T are the most misclassified, correctly classifying 66 queries, while 15 are mistakenly classified as O, 14 as Q and 5 as E. Structurally, queries from the Q group are the most different – they are longer and have common part-of-speech classes from natural language sentences – which explains the high accuracy in this group, while the others (O, E and T) are structurally similar and can confuse the classifier.

The top performing classifier in Table 5.3 still leaves room for further improvement in intent classification accuracy. An interesting question here is whether this

[1]`http://scikit-learn.org/`

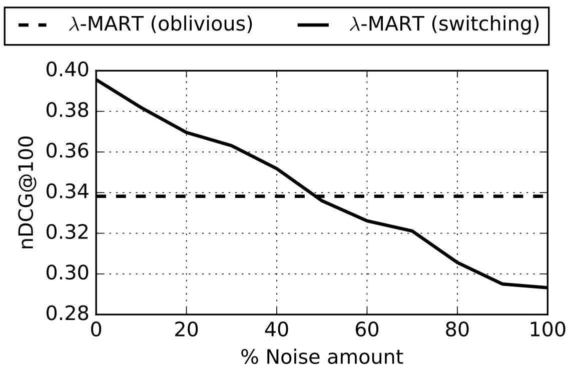


Figure 5.2: Semantic query annotation robustness for simulated intent classifiers of a range of accuracy levels.

level of accuracy is enough for an effective deployment of our proposed intent-aware semantic query annotation approach. To further investigate the role of the intent classification component in our approach, we measure the impact of a range of simulated intent classifiers on the effectiveness of the produced ranking of semantic annotations. In particular, starting from a perfect intent classifier (i.e., an oracle), we gradually introduce noise in the classification outcome by replacing the correct intent with a random one, up to the point where the classification itself becomes a random guess of the four available intents (i.e., E, T, Q, and O). As shown in Figure 5.2, our intent-aware switching strategy can outperform the intent-agnostic oblivious strategy with up to 50% of random noise in intent classification, which is a remarkable result. Recalling *Q2*, the experiments in this section demonstrate that accurate intent classification is feasible, and that the overall ranking annotation performance is robust to a considerable amount of noise in the predicted intents.

5.3 Annotation Effectiveness

Section 5.1 showed the promise of leveraging intent-specific ranking models, while Section 5.2 demonstrated that achieving this promise is feasible with reasonably accurate query intent classifiers. In this section, we address *Q3*, by assessing the effectiveness of our intent-aware semantic query annotation approach in contrast to the various baselines described in Section 4.3. These include BM25 as a vanilla ad-hoc search baseline, FSDM as a representative of the current state-of-the-art, and multiple deployments of LambaMART using baseline intent-agnostic ranking adaptation strategies (fixed and oblivious) as well as our proposed intent-aware strategies (switching and mixing). Table 5.5 summarizes the results of this investigation in terms of P@10, nDCG@10, and

MAP averaged across all 400 test queries from the four query sets in Table 4.2.[2] In each row describing baseline results (the top half of the table), a first of the symbols introduced in Section 4.4 denotes a statistically significant difference (or lack thereof) with respect to LambdaMART (switching), whereas a second symbol denotes potential differences with respect to LambdaMART (mixing). A further symbol is shown alongside LambdaMART (switching) to denote a significant difference (or lack thereof) with respect to LambdaMART (mixing). For each evaluation metric, we also report the number of queries negatively affected (−), positively affected (+), and unaffected (=) when comparing each approach to LambdaMART (mixing).

Table 5.5: Comparison of intent-agnostic (BM25, FSDM, LambdaMART fixed and oblivious) and intent-aware (LambdaMART switching and mixing) semantic query annotation.

	P				**nDCG**				**MAP**			
	@10	−	=	+	**@10**	−	=	+	**@100**	−	=	+
BM25	$0.181^{\blacktriangledown\blacktriangledown}$	52	188	160	$0.250^{\blacktriangledown\blacktriangledown}$	99	99	202	$0.163^{\blacktriangledown\blacktriangledown}$	99	58	243
FSDM	$0.204^{\blacktriangledown\blacktriangledown}$	72	182	146	$0.289^{\blacktriangledown\blacktriangledown}$	115	105	180	$0.195^{\blacktriangledown\blacktriangledown}$	126	63	211
LambdaMART												
(fixed-E)	$0.178^{\blacktriangledown\blacktriangledown}$	35	213	152	$0.244^{\blacktriangledown\blacktriangledown}$	69	119	212	$0.162^{\blacktriangledown\blacktriangledown}$	76	60	264
(fixed-T)	$0.202^{\blacktriangledown\blacktriangledown}$	50	212	138	$0.275^{\blacktriangledown\blacktriangledown}$	95	113	192	$0.172^{\blacktriangledown\blacktriangledown}$	101	58	241
(fixed-Q)	$0.152^{\blacktriangledown\blacktriangledown}$	36	182	182	$0.215^{\blacktriangledown\blacktriangledown}$	57	122	221	$0.139^{\blacktriangledown\blacktriangledown}$	56	69	275
(fixed-O)	$0.188^{\blacktriangledown\blacktriangledown}$	48	209	143	$0.260^{\blacktriangledown\blacktriangledown}$	93	110	197	$0.163^{\blacktriangledown\blacktriangledown}$	96	64	240
(oblivious)	$0.192^{\blacktriangledown\blacktriangledown}$	40	214	146	$0.276^{\blacktriangledown\blacktriangledown}$	87	113	200	$0.178^{\blacktriangledown\blacktriangledown}$	98	58	244
(switching)	$0.227^{\blacktriangledown}$	28	302	70	$0.329^{\blacktriangledown}$	90	169	141	$0.219^{\blacktriangledown}$	101	89	141
(mixing)	**0.243**				**0.346**				**0.229**			

From Table 5.5, we first observe that FSDM performs strongly, outperforming all intent-agnostic variants deployed with LambdaMART, which confirms its effectiveness as a representative of the state-of-the-art. Also of note is the fact that a single model trained on a set of multiple intents using the oblivious strategy cannot consistently improve upon the best performing intent-specific model, produced by the fixed-T strategy. In contrast, both of our intent-aware ranking adaptation strategies are able to consistently leverage the best characteristics of each individual intent, significantly outperforming all intent-agnostic baselines in all settings. In particular, compared to FSDM, our switching strategy improves by up to 11% in P@10, 14% in nDCG@10, and 12% in MAP. Compared to the best performing intent-agnostic strategy under LambdaMART (fixed-T), gains are as high as 12% in P@10, 20% in nDCG@10, and 27% in MAP. Lastly, we also note that our mixing strategy further significantly improves upon the switching strategy. This result suggests that merging multiple intent-specific

[2]Effectiveness breakdown analyses per query intent and various other query characteristics are presented in Section 5.4.

models (the mixing strategy) can be safer than applying a single model associated with the most likely query intent (the switching strategy). Recalling *Q3*, these results attest the effectiveness of our intent-aware ranking adaptation for semantic query annotation.

5.4 Breakdown Analyses

The previous analysis demonstrated the effectiveness of our approach on the entire set of 400 queries. To further shed light on the reasons behind such an effective performance, we address question *Q4*, by analyzing the improvements brought by our approach for queries with different intents, lengths, and difficulty.

5.4.1 Analysis by Query Intent

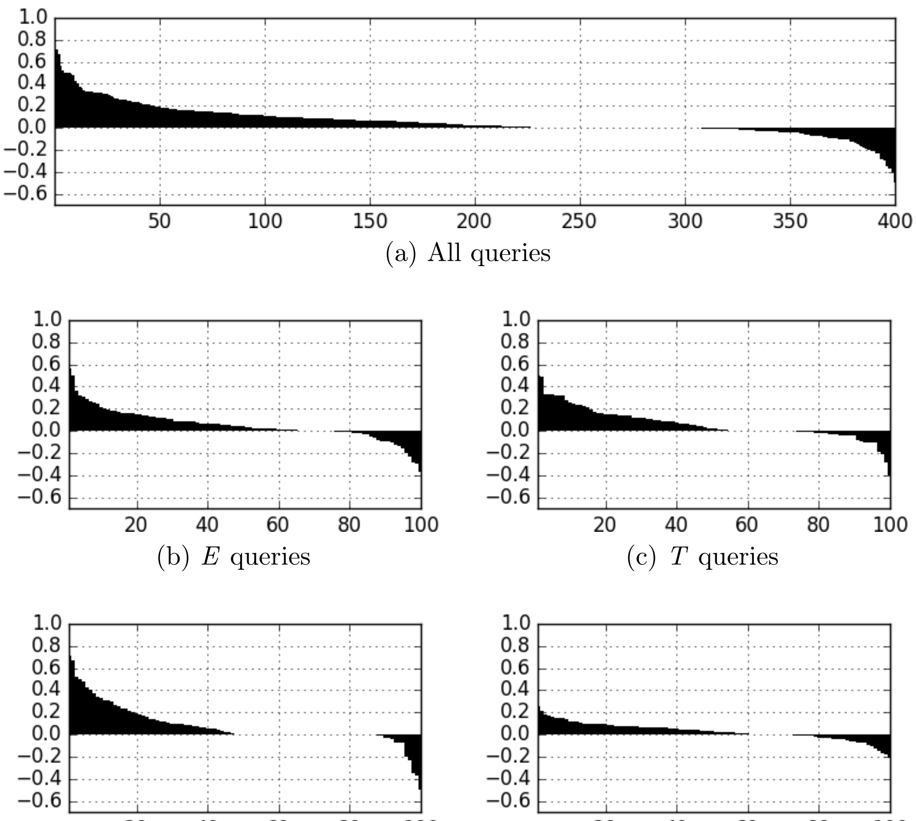
Table 5.6 breaks down the results in Table 5.5 according to the target intent of each query. For brevity, only the best among the fixed strategy variants is shown. Note that while our approach aims to predict the correct intent of each query, there is no guarantee that a perfect intent classification will be achieved, as discussed in Section 5.2. Hence, it is important to understand how well our approach performs on queries of each target intent. From Table 5.5, as expected, the best fixed strategy for each group of queries is that optimized for the group itself (e.g., fixed-E is the best fixed strategy for entity queries—the E group). Nonetheless, our intent-aware mixing strategy is the most consistent across all groups, with effectiveness on a par with the best fixed strategy for each group. Compared to our switching strategy, the mixing strategy is particularly effective for type queries (the T group), with statistical ties for all other groups. Regarding performance differences across the target intents, we note that all approaches achieve their best absolute performance on E queries followed by queries with other intents (the O group), which also includes entity queries. The effective results attained even by the simple BM25 baseline suggest that queries with these intents are well handled by content-based approaches.

Compared to the intent-agnostic FSDM baseline, our largest improvements are observed for type queries (the T group) and question queries (the Q group). For T queries, the structure-based features exploited by our learning to rank approach bring only small improvements, as observed by contrasting LambdaMART (oblivious) with FSDM. However, with our proposed intent-aware ranking adaptation strategies, further marked improvements are observed, with the mixing strategy significantly improving upon the oblivious strategy by up to 25% in P@10, 35% in nDCG@10, and 44% in MAP. For Q queries, both the extra features exploited via learning to rank as well

Table 5.6: Effectiveness breakdown by query intent.

	P				**nDCG**				**MAP**			
	@10	−	=	+	**@10**	−	=	+	**@100**	−	=	+
***E* queries (100 queries)**												
BM25	$0.240^{\blacktriangledown\blacktriangledown}$	14	45	41	$0.416^{\blacktriangledown\blacktriangledown}$	29	17	54	$0.320^{\blacktriangledown\blacktriangledown}$	32	8	60
FSDM	$0.286^{\circ\circ}$	19	49	32	$0.499^{\circ\circ}$	37	19	44	$0.396^{\circ\circ}$	47	10	43
LambdaMART												
(fixed-E)	$0.293^{\vartriangle\circ}$	12	74	14	$0.498^{\circ\circ}$	29	31	40	$0.387^{\circ\circ}$	34	14	52
(fixed-T)	$0.177^{\blacktriangledown\blacktriangledown}$	4	35	61	$0.299^{\blacktriangledown\blacktriangledown}$	11	17	72	$0.224^{\blacktriangledown\blacktriangledown}$	11	5	84
(fixed-Q)	$0.131^{\blacktriangledown\blacktriangledown}$	6	28	66	$0.214^{\blacktriangledown\blacktriangledown}$	10	14	76	$0.158^{\blacktriangledown\blacktriangledown}$	9	6	85
(fixed-O)	$0.245^{\blacktriangledown\blacktriangledown}$	16	46	38	$0.445^{\triangledown\blacktriangledown}$	27	21	52	$0.329^{\blacktriangledown\blacktriangledown}$	25	13	62
(oblivious)	$0.239^{\blacktriangledown\blacktriangledown}$	9	50	41	$0.434^{\blacktriangledown\blacktriangledown}$	21	22	57	$0.329^{\blacktriangledown\blacktriangledown}$	28	7	65
(switching)	0.282°	8	74	18	0.486°	27	33	40	0.377°	29	16	55
(mixing)	**0.297**				**0.502**				**0.390**			
***T* queries (100 queries)**												
BM25	$0.190^{\triangledown\blacktriangledown}$	17	37	46	$0.193^{\blacktriangledown\blacktriangledown}$	27	18	55	$0.141^{\blacktriangledown\blacktriangledown}$	27	9	64
FSDM	$0.211^{\circ\blacktriangledown}$	22	34	44	$0.223^{\circ\blacktriangledown}$	30	18	52	$0.167^{\circ\blacktriangledown}$	31	10	59
LambdaMART												
(fixed-E)	$0.146^{\blacktriangledown\blacktriangledown}$	7	32	61	$0.140^{\blacktriangledown\blacktriangledown}$	10	22	68	$0.111^{\blacktriangledown\blacktriangledown}$	13	9	78
(fixed-T)	$\mathbf{0.289}^{\blacktriangle\circ}$	17	65	18	$\mathbf{0.327}^{\blacktriangle\circ}$	32	33	35	$\mathbf{0.219}^{\blacktriangle\circ}$	39	12	49
(fixed-Q)	$0.195^{\triangledown\blacktriangledown}$	14	39	47	$0.215^{\triangledown\blacktriangledown}$	25	22	53	$0.137^{\blacktriangledown\blacktriangledown}$	17	9	74
(fixed-O)	$0.202^{\circ\blacktriangledown}$	16	38	46	$0.211^{\triangledown\blacktriangledown}$	24	19	57	$0.149^{\blacktriangledown\blacktriangledown}$	28	9	63
(oblivious)	$0.216^{\circ\blacktriangledown}$	12	52	36	$0.225^{\circ\blacktriangledown}$	28	22	50	$0.146^{\blacktriangledown\blacktriangledown}$	28	10	62
(switching)	$0.232^{\blacktriangledown}$	4	66	30	$0.260^{\blacktriangledown}$	16	36	48	$0.185^{\blacktriangledown}$	21	12	67
(mixing)	0.271				0.303				0.210			
***Q* queries (100 queries)**												
BM25	$0.060^{\blacktriangledown\blacktriangledown}$	3	61	36	$0.108^{\blacktriangledown\blacktriangledown}$	5	52	43	$0.077^{\blacktriangledown\blacktriangledown}$	6	37	57
FSDM	$0.061^{\blacktriangledown\blacktriangledown}$	6	58	36	$0.127^{\blacktriangledown\blacktriangledown}$	9	53	38	$0.098^{\blacktriangledown\blacktriangledown}$	9	38	53
LambdaMART												
(fixed-E)	$0.038^{\blacktriangledown\blacktriangledown}$	0	60	40	$0.070^{\blacktriangledown\blacktriangledown}$	3	51	46	$0.050^{\blacktriangledown\blacktriangledown}$	6	34	60
(fixed-T)	$0.104^{\blacktriangledown\blacktriangledown}$	9	69	22	$0.199^{\triangledown\blacktriangledown}$	17	51	32	$0.144^{\triangledown\triangledown}$	19	39	42
(fixed-Q)	$\mathbf{0.143}^{\circ\circ}$	8	88	4	$\mathbf{0.273}^{\circ\circ}$	12	74	14	$\mathbf{0.203}^{\circ\circ}$	19	53	28
(fixed-O)	$0.045^{\blacktriangledown\blacktriangledown}$	2	59	39	$0.080^{\blacktriangledown\blacktriangledown}$	3	51	46	$0.062^{\blacktriangledown\blacktriangledown}$	7	35	58
(oblivious)	$0.091^{\blacktriangledown\blacktriangledown}$	5	69	26	$0.177^{\blacktriangledown\blacktriangledown}$	11	54	35	$0.132^{\blacktriangledown\blacktriangledown}$	14	38	48
(switching)	0.142°	7	89	4	0.267°	11	75	14	0.198°	18	53	29
(mixing)	0.141				0.266				0.194			
***O* queries (100 queries)**												
BM25	$0.235^{\circ\blacktriangledown}$	18	45	37	$0.282^{\circ\triangledown}$	38	12	50	$0.113^{\circ\circ}$	34	4	62
FSDM	$0.258^{\circ\circ}$	25	41	34	$0.308^{\circ\circ}$	39	15	46	$0.119^{\circ\circ}$	39	5	56
LambdaMART												
(fixed-E)	$0.236^{\circ\blacktriangledown}$	16	47	37	$0.268^{\blacktriangledown\blacktriangledown}$	27	15	58	$0.100^{\blacktriangledown\blacktriangledown}$	23	3	74
(fixed-T)	$0.237^{\circ\triangledown}$	20	43	37	$0.275^{\circ\blacktriangledown}$	35	12	53	$0.103^{\circ\blacktriangledown}$	32	2	66
(fixed-Q)	$0.140^{\blacktriangledown\blacktriangledown}$	8	27	65	$0.157^{\blacktriangledown\blacktriangledown}$	10	12	78	$0.057^{\blacktriangledown\blacktriangledown}$	11	1	88
(fixed-O)	$0.259^{\circ\circ}$	14	66	20	$0.304^{\circ\circ}$	39	19	42	$0.113^{\circ\triangledown}$	36	7	57
(oblivious)	$0.221^{\blacktriangledown\blacktriangledown}$	14	43	43	$0.267^{\triangledown\blacktriangledown}$	27	15	58	$0.105^{\circ\blacktriangledown}$	28	3	69
(switching)	0.254°	9	73	18	0.305°	36	25	39	0.116°	33	8	59
(mixing)	**0.264**				**0.312**				**0.123**			

as our ranking adaptation strategies help, with the switching strategy improving even further compared to the oblivious one by up to 56% in P@10, 51% in nDCG@10, and 50% in MAP. Figure 5.3 further illustrates the consistent improvements in terms of nDCG@100 attained by our intent-aware strategies (here represented by the mixing strategy) compared to the intent-agnostic oblivious baseline. Indeed, not only does mixing improve more queries than it hurts compared to oblivious, but it also shows larger increases and smaller decreases throughout queries of all four intents. Analyzing each intent separately, the most noticeable difference can be observed for Q queries, with mixing performing better for 50% of the queries and losing in only 10%. For E and T queries, the differences in nDCG are not as high, but mixing is still superior for 60% of the queries. The smallest gap between the two strategies appears in O queries, although once again mixing performs better for 60% the queries.



(a) All queries

(b) E queries

(c) T queries

(d) Q queries

(e) O queries

Figure 5.3: Differences in nDCG@100 between LambdaMART (mixing) and LambdaMART (oblivious) across: (a) all queries; (b) E queries; (c) T queries; (d) Q queries; (e) O queries. Positive values indicate mixing is better.

5.4.2 Analysis by Query Length

Continuing our detailed analysis, Table 5.7 breaks down the results from Table 5.5 according to the length of each query. In particular, we consider three groups of queries: short queries, with 1 or 2 terms (74 queries); medium queries, with 3 or 4 terms (193 queries); and long queries, with 5 or more terms (133 queries). From Table 5.7, we observe relatively higher performances of all approaches on short queries compared to those of other lengths. FSDM delivers a particularly strong performance on this group, with only a small gap from our mixing strategy, which is the overall best. This can be explained by FSDM's previously discussed effectiveness on E queries, which have only 2.7 terms on average. Compared to the oblivious strategy, mixing brings substantial and significant improvements, once again demonstrating the benefits of an intent-aware ranking adaptation. For medium and long queries (5 or more terms), both of our intent-aware strategies bring even more pronounced improvements compared to all intent-agnostic baselines, with the top performing mixing strategy outperforming the oblivious strategy by up to 32% in P@10, 30% in nDCG@10, and 36% in MAP. This tendency is somewhat expected given the effective performance observed in Table 5.7 for the proposed intent-aware strategies on Q queries, which are typically longer (8 terms on average).

5.4.3 Analysis by Query Difficulty

To complete our breakdown analysis, we regroup all 400 queries in our investigation according to their difficulty. In particular, we consider three groups: difficult queries, with 3 or less relevant results in the ground-truth (108 queries); moderate queries, with 4 to 20 relevant results (184 queries); and easy queries, with more than 20 relevant results (108 queries). The results of this investigation are shown in Table 5.8. From the table, we note as expected that difficult queries generally incur in reduced precision at early ranks (as measured by both P@10 and nDCG@10), while easy queries tend to penalize recall at lower ranks (as measured by MAP). Nevertheless, our intent-aware adaptation strategies are once again the most effective across all groups of queries, with the mixing strategy consistently providing the overall best results. For difficult queries (3 or less relevant results), compared to the oblivious strategy, mixing improves by up to 19% in P@10, 24% in nDCG@10, and 26% in MAP. For easy queries (21 or more relevant results), improvements are as high as 24% in P@10, 27% in nDCG@10, and 39% in MAP.

Table 5.7: Effectiveness breakdown by query length.

	P				**nDCG**				**MAP**			
	@10	−	=	+	**@10**	−	=	+	**@100**	−	=	+
	1 or 2 terms (74 queries)											
BM25	$0.253^{\blacktriangledown\blacktriangledown}$	8	28	38	$0.363^{\blacktriangledown\blacktriangledown}$	18	12	44	$0.268^{\blacktriangledown\blacktriangledown}$	21	3	50
FSDM	$0.323^{\circ\circ}$	15	34	25	$0.456^{\circ\circ}$	28	9	37	$0.331^{\circ\circ}$	36	3	35
LambdaMART												
(fixed-E)	$0.324^{\circ\circ}$	8	50	16	$0.453^{\circ\circ}$	21	19	34	$0.327^{\circ\circ}$	23	5	46
(fixed-T)	$0.227^{\blacktriangledown\blacktriangledown}$	6	28	40	$0.315^{\blacktriangledown\blacktriangledown}$	11	12	51	$0.217^{\blacktriangledown\blacktriangledown}$	9	3	62
(fixed-Q)	$0.170^{\blacktriangledown\blacktriangledown}$	8	20	46	$0.217^{\blacktriangledown\blacktriangledown}$	10	7	57	$0.143^{\blacktriangledown\blacktriangledown}$	8	1	65
(fixed-O)	$0.265^{\blacktriangledown\blacktriangledown}$	11	28	35	$0.398^{\blacktriangledown\blacktriangledown}$	19	9	46	$0.269^{\blacktriangledown\blacktriangledown}$	22	2	50
(oblivious)	$0.276^{\blacktriangledown\blacktriangledown}$	7	35	32	$0.394^{\blacktriangledown\blacktriangledown}$	15	12	47	$0.278^{\blacktriangledown\blacktriangledown}$	21	3	50
(switching)	0.324°	8	52	14	0.455°	21	19	34	0.324°	21	5	48
(mixing)	**0.337**				**0.465**				**0.332**			
	3 or 4 terms (193 queries)											
BM25	$0.196^{\blacktriangledown\blacktriangledown}$	27	82	84	$0.264^{\blacktriangledown\blacktriangledown}$	54	32	107	$0.161^{\blacktriangledown\blacktriangledown}$	47	21	125
FSDM	$0.221^{\circ\blacktriangledown}$	38	81	74	$0.308^{\triangledown\blacktriangledown}$	57	41	95	$0.198^{\triangledown\blacktriangledown}$	56	25	112
LambdaMART												
(fixed-E)	$0.182^{\blacktriangledown\blacktriangledown}$	18	91	84	$0.253^{\blacktriangledown\blacktriangledown}$	36	46	111	$0.161^{\blacktriangledown\blacktriangledown}$	35	24	134
(fixed-T)	$0.234^{\circ\blacktriangledown}$	30	96	67	$0.307^{\triangledown\blacktriangledown}$	58	42	93	$0.182^{\blacktriangledown\blacktriangledown}$	61	20	112
(fixed-Q)	$0.165^{\blacktriangledown\blacktriangledown}$	17	74	102	$0.234^{\blacktriangledown\blacktriangledown}$	30	46	117	$0.150^{\blacktriangledown\blacktriangledown}$	28	27	138
(fixed-O)	$0.209^{\blacktriangledown\blacktriangledown}$	26	95	72	$0.284^{\blacktriangledown\blacktriangledown}$	53	42	98	$0.170^{\blacktriangledown\blacktriangledown}$	45	26	122
(oblivious)	$0.208^{\blacktriangledown\blacktriangledown}$	23	90	80	$0.296^{\blacktriangledown\blacktriangledown}$	51	39	103	$0.183^{\blacktriangledown\blacktriangledown}$	49	19	125
(switching)	$0.240^{\blacktriangledown}$	12	137	44	$0.349^{\blacktriangledown}$	47	71	75	0.227°	49	35	109
(mixing)	**0.265**				**0.376**				**0.239**			
	5 or more terms (133 queries)											
BM25	$0.119^{\blacktriangledown\blacktriangledown}$	17	78	38	$0.167^{\blacktriangledown\blacktriangledown}$	27	55	51	$0.106^{\blacktriangledown\blacktriangledown}$	31	34	68
FSDM	$0.114^{\blacktriangledown\blacktriangledown}$	19	67	47	$0.171^{\blacktriangledown\blacktriangledown}$	30	55	48	$0.115^{\triangledown\blacktriangledown}$	34	35	64
LambdaMART												
(fixed-E)	$0.092^{\blacktriangledown\blacktriangledown}$	9	72	52	$0.116^{\blacktriangledown\blacktriangledown}$	12	54	67	$0.072^{\blacktriangledown\blacktriangledown}$	18	31	84
(fixed-T)	$0.141^{\circ\blacktriangledown}$	14	88	31	$0.206^{\circ\triangledown}$	26	59	48	$0.134^{\circ\triangledown}$	31	35	67
(fixed-Q)	$0.124^{\blacktriangledown\blacktriangledown}$	11	88	34	$0.186^{\blacktriangledown\blacktriangledown}$	17	69	47	$0.120^{\blacktriangledown\blacktriangledown}$	20	41	72
(fixed-O)	$0.114^{\blacktriangledown\blacktriangledown}$	11	86	36	$0.148^{\blacktriangledown\blacktriangledown}$	21	59	53	$0.094^{\blacktriangledown\blacktriangledown}$	29	36	68
(oblivious)	$0.121^{\blacktriangledown\blacktriangledown}$	10	89	34	$0.181^{\blacktriangledown\blacktriangledown}$	21	62	50	$0.116^{\blacktriangledown\blacktriangledown}$	28	36	69
(switching)	0.156°	8	113	12	0.230°	22	79	32	$0.150^{\triangledown}$	31	49	53
(mixing)	**0.160**				**0.236**				**0.158**			

Recalling *Q4*, the results in this section demonstrate the consistency of our intent-aware ranking adaptation strategies for semantic query annotation. Overall, both the switching and the mixing strategies achieve generally improved results for queries of different target intents, lengths, and difficulty levels, often significantly. Particularly, question-oriented queries (the Q intent), long queries (queries with 5 or more terms), and moderate to easy queries (queries with 4 or more relevant results) are the ones that benefit the most from our intent-aware approach.

Table 5.8: Effectiveness breakdown by query difficulty.

	P				**nDCG**				**MAP**			
	@10	−	=	+	**@10**	−	=	+	**@100**	−	=	+
Difficult: 3 or less relevant results (108 queries)												
BM25	$0.059^{\circ\blacktriangledown}$	8	75	25	$0.224^{\blacktriangledown\blacktriangledown}$	17	59	32	$0.179^{\blacktriangledown\blacktriangledown}$	19	43	46
FSDM	$0.064^{\circ\circ}$	11	74	23	$0.256^{\triangledown\blacktriangledown}$	18	60	30	$0.214^{\circ\triangledown}$	23	45	40
LambdaMART												
(fixed-E)	$0.057^{\blacktriangledown\blacktriangledown}$	1	88	19	$0.218^{\blacktriangledown\blacktriangledown}$	7	65	36	$0.177^{\blacktriangledown\blacktriangledown}$	13	43	52
(fixed-T)	$0.062^{\circ\triangledown}$	7	81	20	$0.230^{\triangledown\blacktriangledown}$	17	57	34	$0.180^{\blacktriangledown\blacktriangledown}$	21	41	46
(fixed-Q)	$0.051^{\blacktriangledown\blacktriangledown}$	4	79	25	$0.208^{\blacktriangledown\blacktriangledown}$	6	64	38	$0.173^{\blacktriangledown\blacktriangledown}$	9	47	52
(fixed-O)	$0.056^{\triangledown\blacktriangledown}$	4	81	23	$0.213^{\blacktriangledown\blacktriangledown}$	10	61	37	$0.168^{\blacktriangledown\blacktriangledown}$	13	44	51
(oblivious)	$0.063^{\circ\triangledown}$	5	87	16	$0.259^{\triangledown\blacktriangledown}$	15	59	34	$0.213^{\triangledown\blacktriangledown}$	20	41	47
(switching)	0.069°	1	101	6	0.308°	8	82	18	0.260°	16	59	33
(mixing)	**0.075**				**0.322**				**0.268**			
Moderate: 4 to 20 relevant results (184 queries)												
BM25	$0.210^{\blacktriangledown\blacktriangledown}$	27	81	78	$0.260^{\blacktriangledown\blacktriangledown}$	57	27	100	$0.194^{\blacktriangledown\blacktriangledown}$	58	12	114
FSDM	$0.245^{\circ\blacktriangledown}$	40	71	73	$0.308^{\circ\blacktriangledown}$	64	32	88	$0.235^{\circ\triangledown}$	73	14	97
LambdaMART												
(fixed-E)	$0.214^{\blacktriangledown\blacktriangledown}$	23	86	75	$0.260^{\blacktriangledown\blacktriangledown}$	45	38	101	$0.196^{\blacktriangledown\blacktriangledown}$	48	14	122
(fixed-T)	$0.210^{\blacktriangledown\blacktriangledown}$	17	88	79	$0.264^{\blacktriangledown\blacktriangledown}$	38	43	103	$0.196^{\blacktriangledown\blacktriangledown}$	44	13	127
(fixed-Q)	$0.165^{\blacktriangledown\blacktriangledown}$	17	69	98	$0.208^{\blacktriangledown\blacktriangledown}$	34	41	108	$0.148^{\blacktriangledown\blacktriangledown}$	29	18	137
(fixed-O)	$0.218^{\blacktriangledown\blacktriangledown}$	29	81	74	$0.274^{\blacktriangledown\blacktriangledown}$	51	32	101	$0.195^{\blacktriangledown\blacktriangledown}$	57	17	110
(oblivious)	$0.214^{\blacktriangledown\blacktriangledown}$	19	92	73	$0.276^{\blacktriangledown\blacktriangledown}$	46	40	98	$0.202^{\blacktriangledown\blacktriangledown}$	53	13	118
(switching)	$0.261^{\blacktriangledown}$	15	132	37	$0.329^{\blacktriangledown}$	48	62	74	$0.245^{\blacktriangledown}$	51	25	108
(mixing)	**0.279**				**0.345**				**0.257**			
Easy: 21 or more relevant results (108 queries)												
BM25	$0.255^{\blacktriangledown\blacktriangledown}$	17	32	59	$0.259^{\blacktriangledown\blacktriangledown}$	25	13	70	$0.094^{\blacktriangledown\blacktriangledown}$	22	3	83
FSDM	$0.275^{\triangledown\blacktriangledown}$	21	37	50	$0.292^{\blacktriangledown\blacktriangledown}$	33	13	62	$0.107^{\blacktriangledown\blacktriangledown}$	30	4	74
LambdaMART												
(fixed-E)	$0.239^{\blacktriangledown\blacktriangledown}$	11	39	58	$0.244^{\blacktriangledown\blacktriangledown}$	17	16	75	$0.089^{\blacktriangledown\blacktriangledown}$	15	3	90
(fixed-T)	$0.328^{\circ\circ}$	26	43	39	$0.338^{\circ\circ}$	40	13	55	$0.125^{\circ\blacktriangledown}$	36	4	68
(fixed-Q)	$0.232^{\blacktriangledown\blacktriangledown}$	15	34	59	$0.233^{\blacktriangledown\blacktriangledown}$	17	17	74	$0.089^{\blacktriangledown\blacktriangledown}$	18	4	86
(fixed-O)	$0.268^{\blacktriangledown\blacktriangledown}$	15	47	46	$0.283^{\blacktriangledown\blacktriangledown}$	32	17	59	$0.104^{\blacktriangledown\blacktriangledown}$	26	3	79
(oblivious)	$0.283^{\blacktriangledown\blacktriangledown}$	16	35	57	$0.292^{\blacktriangledown\blacktriangledown}$	26	14	68	$0.103^{\blacktriangledown\blacktriangledown}$	25	4	79
(switching)	$0.328^{\triangledown}$	12	69	27	$0.352^{\triangledown}$	34	25	49	$0.134^{\blacktriangledown}$	34	5	69
(mixing)	**0.350**				**0.371**				**0.143**			

5.5 Success and Failure Cases

For more detailed qualitative analysis, we focus on discussing some success and failure cases of the LambdaMART (switching) model, showing query examples where intent-awareness can contribute to a better result when compared to the agnostic LambdaMART (oblivious) model, and examples where it cannot. To this end, we selected queries with the largest difference in terms of MAP between the LambdaMART (switching) and the agnostic LambdaMART (oblivious), meaning a hight improvement on the final ranking. To investigate the opposite case, where intent-awareness did not help to improve the final ranking, we also selected queries with the smallest difference in terms of MAP.

We observe that, in general, the fact that LambdaMART (switching) is composed of different models, trained specifically for different intents, makes each model prioritize the most suitable features for each intent, helping LambdaMART (switching) perform better than the agnostic one (oblivious). In particular, comparing the performance of the intent-aware model, LambdaMART (switching), with LambdaMART (oblivious) for question answering queries (the Q set), we can observe that LambdaMART (switching) can behave more properly to answer questions than LambdaMART (oblivious). In the query QALD2_tr-15 “Who created Goofy?”, LambdaMART (switching) has a MAP score of 1.0, bringing the only relevant entity (“Art Babbitt”) in the top of the ranking, while in LambdaMART (oblivious) it does not appear in the top 100 results, that is, the MAP score is 0. This can be explained by the ranking model trained specifically to question intent in LambdaMART (switching), which considers, as the second and third most important features, the *No. of direct relations with query entities* and *No. of matched relations with query terms* (features #8 and #9 from Table 3.2), respectively. In this example, the candidate entity “Art Babbitt” is directly related to the query entity “Goofy” by the term “created”, also present in the query, and contributing to put this entity on the top of the results, while in LambdaMART (oblivious) these two features are the 7th and 15th in the list of importance, respectively.

Another example that illustrates that the proper use of features benefits intent-aware models, is the query SemSearch_ES-135 “spring shoes canada” (from the E set). In LambdaMART (switching), the relevant entity “Spring store” appears on the top of the ranking with a MAP score of 1.0, while in LambdaMART (oblivious) in the 3rd place, with a MAP score of 0.33. In LambdaMART (switching), the most important feature for this query, is the *TF-IDF bigrams sum in the similar entities field*, benefiting the candidate “Spring store”, which has the bigram “spring store” as an alternative name in the *similar entities* field. In LambdaMART (oblivious), this feature is the 3rd most important.

For most of the cases where the intent-aware LambdaMART (switching) was worse than agnostic LambdaMART (oblivious), the difference in the MAP score between both were small, and can be explained by the mistakenly use of inappropriate features, caused by a mistaken annotation or by the inadequate relations on the knowledge base. An example of mistaken annotation is the query SemSearch_ES-142 "windsor hotel philadelphia", where LambdaMART (switching) has a MAP score of 0.64 and mistakenly put the candidate entity "Windsor Hotel Montreal" on the top of the ranking because DBPedia Spotlight annotated the query with this entity and LambdaMART (switching) considered the feature *Matching entity* in the query, mistakenly promoting this entity to the top, while LambdaMART (oblivious) has a MAP score of 0.92 and did not consider the feature *Matching entity*.

An example of inadequate relations on the knowledge base is the query QALD2_te-76 "List the children of Margaret Thatcher", where LambdaMART (switching) has a MAP score of 0.04, putting non relevant entities, like "Geoffrey Howe" and "Douglas Hurd" on the top, leaving the two relevant entities ("Carol Thatcher" and "Mark Thatcher") in the 17th and 28th position of the ranking, respectively. This happened because there are many repeated relationships between the query entity "Margaret Thatcher" and the entity "Geoffrey Howe" (and "Douglas Hurd" as well), causing the feature *No. of direct relations with query entities*, considered by LambdaMART (switching), to be overvalued for these candidates. Since LambdaMART (oblivious) does not give such importance to this feature, it can put the two relevant entities on the top of the ranking, with a MAP score of 1.0.

5.6 Summary

In Chapter 3 we proposed an intent-aware framework for learning to rank semantic query annotations and, in Chapter 4, we introduced the research questions inherent to the proposed framework and gave details of the experimental setup aimed to answer them. In this chapter, we presented the experimental evaluation which thoroughly validated the effectiveness of our framework.

In Section 5.1 we answered the first research question, which is about the effectiveness of using different ranking models for different query intents. We listed the feature importance of each ranking model, showing that each one gives importance to different features. We also computed the Spearman's rank correlation between them to further show that they are really distinct. Besides, we tested queries of each intent in all ranking models to show that different queries benefit from ranking models optimized to their intent.

In Section 5.2 we answered the second research question about the accuracy in intent prediction. We presented the accuracy of a range of traditional classification algorithms, showing that logistic regression can obtain the highest accuracy, predicting 77% of query intents correctly. We also tested how much noise in intent prediction our framework can withstand to still be superior to agnostic approaches, getting a value of 50% of noise, a considerable amount.

In Section 5.3, we presented the effectiveness of our two strategies (switching and mixing) when compared to different baselines, showing that both strategies significantly outperform agnostic methods, with the mixing strategy getting better scores. In Section 5.4, we presented a further analysis, comparing both of our strategies with agnostic baselines for queries with different intents, lengths and difficulties. In all cases, at least one of the two strategies outperformed the baselines.

In Section 5.5, we discussed some success and failure cases where our intent-awareness framework could contribute to a better result when compared to the agnostic baseline, showing that our approach benefits from the use of appropriate features corresponding to the query intent. On the other hand, mistaken annotations on the knowledge base or the misuse of inappropriate features may harm our approach.

With this chapter, we conclude the experimental evaluation of our intent-aware semantic query annotation framework. In the next chapter, we recap on the contributions of this thesis and future work.

Chapter 6

Conclusions and Future Work

With the growth of Internet access, mainly through mobile devices, seeking for information using search engines has become a common task of our daily lives, increasing the number of searches and consequently their diversity. This evolution makes the area of information retrieval remain challenging, even over the years. A search can have different goals and expect different types of results, so it is important that search engines be aware of this diversity, to adapt their searching strategy according to the user's intent and properly deliver the expected answer. With this aim, we proposed a framework for learning to rank semantic annotations, which detects the intent behind the query and adapts the final ranking according to it.

The following sections summarize the conclusions drawn from our investigation and the main contributions of this work, to finally conclude with directions for future works.

6.1 Summary of Contributions

In the following, we summarize the main contributions of this thesis.

An intent-aware framework for learning semantic query annotations. In Chapter 3 we proposed a framework for semantic query annotations that is sensitive to the user's search intent, comprising three main components: (i) intent-specific learning to rank, aimed to produce ranking models optimized for different intents; (ii) query intent classification, aimed to estimate the probability of each query conveying each possible intent; and (iii) intent-aware ranking adaptation, aimed to promote the most relevant annotations given the detected intents.

An analysis of the specificity of several content and structural features for different query intents. In Chapter 3, we proposed a set of features based on textual content and also semantic features derived from the structure of the knowledge base. These features were used in different ranking models, optimized according to the different query intents. In Chapter 5 we made an analysis of the features for each ranking model, investigating the most relevant features in each intent, correlating them and discussing their specificities.

A thorough validation of the proposed framework. In Chapter 5, we thoroughly validated our intent-aware framework in contrast to state-of-the-art intent agnostic approaches from different aspects, including an intent agnostic approach which uses the same learning-to-rank method used by our framework, to demonstrate the advantages of considering the query intent while ranking. In particular, in Section 5.4 we performed several breakdown analyses, comparing results for different query intents, lengths and difficulty levels. We also made a qualitative analysis of our approach in Section 5.5, discussing success and failure cases.

6.2 Summary of Conclusions

We presented a framework for learning to rank semantic annotations suitable to the intent of each individual query. Our approach predicts the intent of a target query and adapts the ranking produced for this query using one of two strategies: *switching*, which applies a ranking model trained on queries of the same intent as predicted for the target query, or *mixing*, which combines the results of multiple intent-specific ranking models according to their predicted likelihood for the target query. Extensive experiments on a publicly available benchmark demonstrated the effectiveness of our approach for semantic query annotation, with significant improvements compared to state-of-the-art intent-agnostic approaches. The results also attested the consistency of the observed improvements for queries of different intents, lengths, and difficulty levels.

6.3 Directions for Future Work

In the future, we plan to assess the impact of intent-aware learning on frameworks other than learning to rank. Preliminary results in this direction show that the FSDM baseline, which is based on the Markov random fields framework, can be improved with an intent-aware approach to hyperparameter tuning, although with less marked gains compared to the ones observed in our experiments with feature-based models using learning to rank.

For future work in the context of intent awareness, we intend to evaluate our approach with a larger intent taxonomy, including more queries with less common intents such as attribute and relation queries. We also intend to focus on the problem of question answering, inspired by the effectiveness of the natural language question model in our experiments, we can explore specific features for question answering and contrast with state-of-the-art approaches for this task. We can also apply the proposed framework in tasks of query expansion to deal with the fundamental issue of vocabulary mismatch in information retrieval, according to the identified intent.

Finally, another direction for future work is the enhancement and maintenance of the knowledge base, exploring strategies to enrich it with data from the open web and also discuss strategies to deal with data temporality, e.g. the query "us president" will expect different answers over time.

Bibliography

Alonso, O. and Zaragoza, H. (2008). Exploiting semantic annotations in information retrieval: ESAIR '08. *SIGIR Forum*, 42(1):55--58.

Balog, K., de Vries, A. P., Serdyukov, P., Thomas, P., and Westerveld, T. (2009). Overview of the TREC 2009 Entity track. In *Proceedings of the 18th Text REtrieval Conference.*

Balog, K. and Neumayer, R. (2013). A test collection for entity search in dbpedia. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 737--740.

Bast, H., Buchhold, B., Haussmann, E., et al. (2016). Semantic search on text and knowledge bases. *Foundations and Trends® in Information Retrieval*, 10(2-3):119--271.

Bi, B., Ma, H., Hsu, B.-J. P., Chu, W., Wang, K., and Cho, J. (2015). Learning to recommend related entities to search users. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 139--148.

Blanco, R., Mika, P., and Vigna, S. (2011). Effective and efficient entity search in rdf data. In *International Semantic Web Conference*, pages 83--97.

Blanco, R., Mika, P., and Zaragoza, H. (2010). Entity search track submission by yahoo! research barcelona. In *Proceedings of the 19th international conference on World Wide Web*, volume 10.

Brenes, D. J., Gayo-Avello, D., and Pérez-González, K. (2009). Survey and evaluation of query intent detection methods. In *Proceedings of the 2009 Workshop on Web Search Click Data*, pages 1--7.

Broder, A. (2002). A taxonomy of web search. *SIGIR Forum*, 36(2):3--10.

Bron, M., Balog, K., and de Rijke, M. (2010). Ranking related entities: components and analyses. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1079--1088.

Bron, M., Balog, K., and De Rijke, M. (2013). Example based entity search in the web of data. In *European Conference on Information Retrieval*, pages 392--403.

Campinas, S., Delbru, R., Rakhmawati, N. A., Ceccarelli, D., and Tummarello, G. (2011). Sindice bm25f at semsearch 2011. In *Proceedings of the 4th International Semantic Search Workshop*.

Chapelle, O. and Chang, Y. (2011). Yahoo! learning to rank challenge overview. In *Yahoo! Learning to Rank Challenge*, pages 1--24.

Craswell, N. and Hawking, D. (2004). Overview of the TREC 2004 Web track. In *Proceedings of the 13rd Text REtrieval Conference*.

de Vries, A. P., Vercoustre, A.-M., Thom, J. A., Craswell, N., and Lalmas, M. (2007). Overview of the INEX 2007 Entity Ranking track. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 245–251.

Elbassuoni, S. and Blanco, R. (2011). Keyword search over rdf graphs. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 237--242.

Elbassuoni, S., Ramanath, M., Schenkel, R., Sydow, M., and Weikum, G. (2009). Language-model-based ranking for queries on rdf-graphs. In *Proceedings of the 18th ACM International Conference on Information and Knowledge Management*, pages 977--986.

Fetahu, B., Gadiraju, U., and Dietze, S. (2015). Improving entity retrieval on structured data. In *International Semantic Web Conference*, pages 474--491.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189--1232.

Geng, X., Liu, T.-Y., Qin, T., Arnold, A., Li, H., and Shum, H.-Y. (2008). Query dependent ranking using k-nearest neighbor. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115--122.

Guy, I. (2016). Searching by talking: analysis of voice queries on mobile web search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 35--44.

Herzig, D. M., Mika, P., Blanco, R., and Tran, T. (2013). Federated entity search using on-the-fly consolidation. In *International Semantic Web Conference*, pages 167--183.

Jansen, B. J., Spink, A., and Saracevic, T. (2000). Real life, real users, and real needs: A study and analysis of user queries on the Web. *Information Processing and Management*, 36(2):207--227.

Kang, I.-H. and Kim, G. (2003). Query type classification for web document retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 64--71.

Krovetz, R. (1993). Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191--202.

Liu, T.-Y. et al. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225--331.

Lucchese, C., Nardini, F. M., Orlando, S., Perego, R., and Tonellotto, N. (2015). Speeding up document ranking with rank-based features. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 895--898.

Metzler, D. and Croft, W. B. (2005). A markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 472--479.

Neumayer, R., Balog, K., and Nørvåg, K. (2012). On the modeling of entities for ad-hoc entity search in the web of data. In *Proceedings of the 34th European Conference on IR Research*, pages 133--145.

Nikolaev, F., Kotov, A., and Zhiltsov, N. (2016). Parameterized fielded term dependence models for ad-hoc entity retrieval from knowledge graph. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 435--444.

Ogilvie, P. and Callan, J. (2003). Combining document representations for known-item search. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 143--150.

Peng, J., Macdonald, C., and Ounis, I. (2010). Learning to select a ranking function. In *Proceedings of the 32nd Annual European Conference on Information Retrieval*, pages 114--126.

Pérez-Agüera, J. R., Arroyo, J., Greenberg, J., Iglesias, J. P., and Fresno, V. (2010). Using bm25f for semantic search. In *Proceedings of the 3rd International Semantic Search Workshop*, page 2.

Pound, J., Mika, P., and Zaragoza, H. (2010). Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th International Conference on World Wide Web*, pages 771--780.

Robertson, S. E., Walker, S., Hancock-Beaulieu, M., Gatford, M., and Payne, A. (1995). Okapi at TREC-4. In *Proceedings of the 4th Text REtrieval Conference.*

Rocha, C., Schwabe, D., and Aragao, M. P. (2004). A hybrid approach for searching in the semantic web. In *Proceedings of the 13th International Conference on World Wide Web*, pages 374--383.

Rose, D. E. and Levinson, D. (2004). Understanding user goals in web search. In *Proceedings of the 13th International Conference on World Wide Web*, pages 13--19.

Santos, R. L. T., Macdonald, C., and Ounis, I. (2010a). Selectively diversifying web search results. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1179--1188.

Santos, R. L. T., Macdonald, C., and Ounis, I. (2010b). Voting for related entities. In *Adaptivity, Personalization and Fusion of Heterogeneous Information*, pages 1--8.

Santos, R. L. T., Macdonald, C., and Ounis, I. (2011). Intent-aware search result diversification. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595--604.

Tonon, A., Demartini, G., and Cudré-Mauroux, P. (2012). Combining inverted indices and structured search for ad-hoc object retrieval. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 125--134.

Tsur, G., Pinter, Y., Szpektor, I., and Carmel, D. (2016). Identifying web queries with question intent. In *Proceedings of the 25th International Conference on World Wide Web*, pages 783--793.

Wu, Q., Burges, C. J., Svore, K. M., and Gao, J. (2008). Ranking, boosting, and model adaptation. Technical report, Microsoft Research.

Yom-Tov, E., Fine, S., Carmel, D., and Darlow, A. (2005). Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 512--519.

Zhiltsov, N. and Agichtein, E. (2013). Improving entity search over linked data by modeling latent semantics. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 1253--1256.

Zhiltsov, N., Kotov, A., and Nikolaev, F. (2015). Fielded sequential dependence model for ad-hoc entity retrieval in the web of data. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253--262.