Combining concepts and language models for information access

Meij, E.J.

Citation for published version (APA):
Meij, E. J. (2010). Combining concepts and language models for information access
In Chapter 5 we have used annotated documents to obtain a conceptual representation of a query model: a conceptual query model. As we have seen there, leveraging textual observations associated with concepts during query modeling significantly improves end-to-end retrieval performance. In this chapter we further investigate the process of mapping queries to concepts, a procedure we call conceptual mapping. We do so in a more general context, by linking large numbers of actual search engine queries (taken from a transaction log) to DBpedia [15], which is an ontology extracted from Wikipedia. The methods presented and evaluated in this chapter serve as a precursor to the next chapter. There, we evaluate retrieval performance when using the natural language text associated with concepts that are obtained using the methods presented here.

Performing a conceptual mapping between queries to concepts could serve several purposes. For one, in the case of a collection of documents annotated using concepts, the obtained concepts may be used to match the documents to the query. They may also be used to obtain a contribution to the textual query model, similar to the method presented in the preceding chapter. Furthermore, such mappings may serve to retrieve concepts themselves. The INEX Entity Ranking track, for example, provides a use-case for retrieving entities (which are defined as Wikipedia articles). As we have seen in Chapter 2, other uses for conceptual mappings also include natural language interfaces to databases or knowledge repositories.

Conceptually mapping queries is not only interesting from an IR point of view, but also has clear benefits for the semantic web (SW) community in that it provides an easy access method into the Linked Open Data (LOD) cloud (of which DBpedia is a part—cf. Figure 6.1). A significant task towards building and maintaining the semantic web is link generation. Links allow a person or machine to explore and understand the web of data more easily: when you have linked data, you can find related data [32]. The LOD [32, 36, 37] initiative extends the web by publishing various open data sets and by setting links between items (or concepts) from different data sources in a (semi-)automated fashion [15, 27, 307]. The resulting data commons is termed the Linked Open Data cloud, and provides
a key ingredient for realizing the semantic web. At the time of writing, the LOD cloud contains millions of concepts from over one hundred structured data sets.

Unstructured data resources—such as textual documents or queries submitted to a search engine—can be enriched by mapping their content to structured knowledge repositories like the LOD cloud. This type of enrichment may serve multiple goals, such as explicit anchoring of the data resources in background knowledge or ontology learning and population. The former enables new forms of intelligent search and browsing; authors or readers of a piece of text may find mappings to the LOD cloud to supply useful pointers, for example, to concepts capturing or relating to the contents of the document. In ontology learning applications, mappings may be used to learn new concepts or relations between them [324]. Recently, data-driven methods have been proposed to map phrases appearing in full-text documents to Wikipedia articles. For example, Mihalcea and Csomai [226] propose incorporating linguistic features in a machine learning framework to map phrases in full-text documents to Wikipedia articles—this approach is further improved upon by Milne and Witten [230]. Because of the connection between Wikipedia and DBpedia [15], such data-driven linking methods help us to establish links between textual documents and the LOD cloud, with
DBpedia being one of the key interlinking hubs in the cloud. Indeed, we consider DBpedia to be an integral part of and, as such, a perfect entry point into the LOD cloud.

Search engine queries are one type of unstructured data that could benefit from being mapped to a structured knowledge base such as DBpedia. Semantic mappings of this kind can be used to support users in their search and browsing activities, for example by (i) helping the user acquire contextual information, (ii) suggesting related concepts or associated terms that may be used for search, and (iii) providing valuable navigational suggestions. In the context of web search, various methods exist for helping the user formulate his or her queries [10, 144, 217]. For example, the Yahoo! search interface features a so-called “searchassist,” that suggests important phrases in response to a query. While these suggestions inherit natural language semantics, they lack any formal semantics, however, which we address in this chapter by mapping queries to DBpedia concepts. In the case of a specialized search engine with accompanying knowledge base, automatic mappings between natural language queries and concepts aid the user in exploring the contents of both the collection and the knowledge base [41]. They can also help a novice user understand the structure and specific nomenclature of the domain. Furthermore, when the items to be retrieved are also annotated (e.g., using concepts from the LOD cloud through RDFa, microformats, or any other kind of annotation framework), the semantic mappings on the queries can be used to facilitate matching at the semantic level or an advanced form of query-based faceted result presentation. This can partly be achieved by simply using a richer indexing strategy of the items in the collection together with conventional querying mechanisms. Generating conceptual mappings for the queries, however, can improve the matching and help clarify the structure of the domain to the end user.

Once a conceptual mapping has been established, the links between a query and a knowledge repository can be used to create semantic profiles of users based on the queries they issue. They can also be exploited to enrich items in the LOD cloud, for instance by viewing a query as a (user-generated) annotation of the items it has been linked to, similar to the way in which a query can be used to label images that a user clicks on as the result of a search [320]. As we have shown in [227], this type of annotation can, for example, be used to discover aspects or facets of concepts. In this chapter, we focus on the task of automatically mapping free text search engine queries to the LOD cloud, in particular DBpedia. As an example of the task, consider the query “obama white house.” The query mapping algorithm we envision should return links to the concepts labeled BARACK OBAMA and WHITE HOUSE.

Queries submitted to a search engine are particularly challenging to map to structured knowledge repositories, as they tend to consist of only a few terms and are much shorter than typical text documents [144, 300]. Their limited length
implies that we have far less context than in regular text documents. Hence, we cannot use previously established approaches that rely on context such as shallow parsing or part-of-speech tagging [226]. To address these issues, we propose a novel method that leverages the textual representation of each concept as well as query-based and concept-based features in a machine learning framework. At the same time, working with search engine queries entails that we do have search history information available that provides a form of contextual anchoring. In this chapter, we employ this query-specific kind of context as a separate type of feature.

Our approach to conceptual mapping of queries to concepts can be summarized as follows. First, given a query, we use language modeling for IR to retrieve the most relevant concepts as potential targets for mapping. We then use supervised machine learning methods to decide which of the retrieved concepts should be mapped and which should be discarded. In order to train the machine learner, we examined close to 1000 search engine queries and manually mapped over 600 of these to relevant concepts in DBpedia.¹

The research questions we address in this chapter are the following.

RQ 3. Can we successfully address the task of mapping search engine queries to concepts using a combination of information retrieval and machine learning techniques?

A typical approach for mapping text to concepts is to apply some form of lexical matching between concept labels and terms, typically using the context of the text for disambiguation purposes. What are the results of applying this method to our task? What are the results when using a purely retrieval-based approach? How do these results compare to those of our proposed method?

a. What is the best way of handling a query? That is, what is the performance when we map individual n-grams in a query instead of the query as a whole?

b. As input to the machine learning algorithms we extract and compute a wide variety of features, pertaining to the query terms, concepts, and search history. Which type of feature helps most? Which individual feature is most informative?

c. Machine learning generally comes with a number of parameter settings. We ask: what are the effects of varying these parameters? What are the effects of varying the size of the training set, the fraction of positive examples, as well as any algorithm-specific parameters? Furthermore, we provide the machine learning step with a small set of candidate concepts. What are the effects of varying the size of this set?

¹The queries, assessments, and extracted features are publicly available for download at http://ilps.science.uva.nl/resources/jws10_annotations.
6.1. The Task

Our main contributions are as follows. We propose and evaluate two variations of a novel and effective approach for mapping queries to DBpedia and, hence, the LOD cloud. We accompany this with an extensive analysis of the results, of the robustness of our methods, and of the contributions of the features used. We also facilitate future work on the problem by making our used resources publicly available.

The remainder of this chapter is structured as follows. Sections 6.1 and 6.2 detail the query mapping task and our approach. Our experimental setup is described in Section 6.3 and our results are presented in Section 6.4. Section 6.5 follows with a discussion and detailed analysis of the results and we end with a concluding section.

6.1 The Task

The query mapping task that we address in this chapter is the following. Given a query submitted to a search engine, identify the concepts that are intended by the user issuing the query, where the concepts are taken from a structured knowledge base. We address our task in the setting of a digital archive, specifically, the Netherlands Institute for Sound and Vision (“Sound and Vision”). Sound and Vision maintains a large digital audiovisual collection, currently containing over a million objects and updated daily with new television and radio broadcasts.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:comment</td>
<td>Barack Hussein Obama II (born August 4, 1961) is the 44th and current President of the United States. He is the first African American to hold the office. Obama previously served as the junior United States Senator from Illinois, from January 2005 until he resigned after his election to the presidency in November 2008.</td>
</tr>
<tr>
<td>dbpprop:abstract</td>
<td>Barack Hussein Obama II (born August 4, 1961) is the 44th and current President of the United States. He is the first African American to hold the office. Obama previously served as the junior United States Senator from Illinois, from January 2005 until he resigned after his election to the presidency in November 2008. Originally from Hawaiii, Obama is a graduate of Columbia University and Harvard Law School, where he was the president of the Harvard Law Review and where he received a doctorate in law. He was a community organizer [...]</td>
</tr>
</tbody>
</table>
Users of the archive’s search facilities consist primarily of media professionals who use the online search interface to locate audiovisual items to be used in new programs such as documentaries and news reviews. The contents of the audiovisual items are diverse and cover a wide range of topics, people, places, and more. Furthermore, a significant part (around 50%) of the query terms are informational; consisting of either general keywords or proper names [142].

Because of its central role in the LOD initiative, our knowledge source of choice for semantic query suggestion is DBpedia. Thus, in practical terms, the task we are facing is: given a query (within a session, for a given user), produce a ranked list of concepts from DBpedia that are intended by the query. These concepts can then be used, for example, to suggest relevant multimedia items associated with each concept, to suggest linked geodata from the LOD cloud, or to suggest contextual information, such as text snippets from a Wikipedia article.

### 6.2 Approach

Our approach for mapping search engine queries to concepts consists of two stages. In the first stage, we select a set of candidate concepts. In the second stage, we use supervised machine learning to classify each candidate concept as being intended by the query or not.

In order to find candidate concepts in the first stage, we leverage the textual descriptions (rdfs:comment and/or dbpprop:abstract in the case of DBpedia) of the concepts as each description of a concept may contain related words, synonyms, or alternative terms that refer to the concept. An example is given in...
6.2. Approach

<table>
<thead>
<tr>
<th>N-gram (Q)</th>
<th>Candidate concepts</th>
</tr>
</thead>
</table>
| obama white house     | **WHITE HOUSE; WHITE HOUSE STATION; PRESIDENT COOLIDGE;**
                      | **SENSATION WHITE**                                     |
| obama white           | **MICHELLE OBAMA; BARACK OBAMA; DEMOCRATIC PRE-ELECTIONS 2008;**
                      | **JANUARY 17**                                          |
| white house           | **WHITE HOUSE; WHITE HOUSE STATION; SENSATION WHITE;**
                      | **PRESIDENT COOLIDGE**                                  |
| obama                 | **BARACK OBAMA; MICHELLE OBAMA; PRESIDENTIAL ELECTIONS 2008;**
                      | **HILLARY CLINTON**                                     |
| white                 | **COLONEL WHITE; EDWARD WHITE; WHITE COUNTY;**          |
                      | **WHITE PLAINS ROAD LINE**                              |
| house                 | **HOUSE; ROYAL OPERA HOUSE; SYDNEY OPERA HOUSE; FULL HOUSE** |

**Table 6.2:** An example of generating n-grams for the query “obama white house” and retrieved candidate concepts, ranked by retrieval score. Correct concepts in boldface.

Table 6.1, while the Wikipedia article it is extracted from is shown in Figure 6.2. From this example it is clear that the use of such properties for retrieval improves recall (we find BARACK OBAMA using the terms “President of the United States”) at the cost of precision (we also find BARACK OBAMA when searching for “John McCain”). In order to use the concept descriptions, we adopt a language modeling for information retrieval framework to create a ranked list of candidate concepts. This framework will be further introduced in Section 6.2.1.

Since we are dealing with an ontology extracted from Wikipedia, we have several options with respect to which textual representation(s) we use. Natural possibilities include: (i) the title of the article (similar to a lexical matching approach where only the rdfs:label is used), (ii) the first sentence or paragraph of an article (where a definition should be provided according to the Wikipedia guidelines [342]), (iii) the full text of the article, (iv) the anchor texts of the incoming hyperlinks from other articles, and (v) a combination of any of these. For our experiments we aim to maximize recall and use the combination of all available fields with or without the incoming anchor texts. In Section 6.5.2 we discuss the relative performance of each field and of their combinations.

For the first stage, we also vary the way we handle the query. In the simplest case, we take the query as is and retrieve concepts for the query in its entirety. As an alternative, we consider extracting all possible n-grams from the query, generating a ranked list for each, and merging the results. An example of what happens when we vary the query representation is given in Table 6.2 for the query “obama white house.” From this example it is clear why we differentiate between the two ways of representing the query. If we simply use the full query on its own (first row), we miss the relevant concept BARACK OBAMA. However, as can be seen from the last two rows, considering all n-grams also introduces noise.

In the second stage, a supervised machine learning approach is used to clas-
sify each candidate concept as either relevant or non-relevant or, in other words, to decide which of the candidate concepts from the first stage should be kept as viable concepts for the query in question. In order to create training material for the machine learning algorithms, we asked human annotators to assess search engine queries and manually map them to relevant DBpedia concepts. More details about the test collection and manual annotations are provided in Section 6.3. The machine learning algorithms we consider are Naive Bayes, Decision Trees, and Support Vector Machines [326, 344] which are further detailed in Section 6.2.2. As input for the machine learning algorithms we need to extract a number of features. We consider features pertaining to the query, concept, their combination, and the session in which the query appears; these are specified in Section 6.2.3.

6.2.1 Ranking Concepts

We base our concept ranking framework within the language modeling paradigm as introduced in Chapter 2. For the n-gram based scoring method, we extract all n-grams from each query \( Q \) (where \( 1 \leq n \leq |Q| \)) and create a ranked list of concepts for each individual n-gram, \( Q \). For the full query based reranking approach, we use the same method but add the additional constraint that \( n = |Q| \). The problem of ranking DBpedia concepts given \( Q \) can then be formulated as follows. Each concept \( c \) should be ranked according to the probability \( P(c|Q) \) that it was generated by the n-gram, which can be rewritten using Bayes’ rule as:

\[
P(c|Q) = \frac{P(Q|c)P(c)}{P(Q)}. \tag{6.1}
\]

Here, for a fixed n-gram \( Q \), the term \( P(Q) \) is the same for all concepts and can be ignored for ranking purposes. The term \( P(c) \) indicates the prior probability of selecting a concept, which we assume to be uniform. Assuming independence between the individual terms \( q \in Q \) (cf. Eq. 2.3) we obtain

\[
P(c|Q) \propto P(c) \prod_{q \in Q} P(q|c)^{n(q,Q)}, \tag{6.2}
\]

where the probability \( P(q|c) \) is determined by looking at the textual relations as illustrated in Table 6.1. It is smoothed using Bayes smoothing with a Dirichlet prior (cf. Eq. 2.7).

6.2.2 Learning to Select Concepts

Once we have obtained a ranked list of possible concepts for each n-gram, we turn to concept selection. In this stage we need to decide which of the candidate concepts are most viable. We use a supervised machine learning approach that takes as input a set of labeled examples (query to concept mappings) and several features of these examples (detailed below). More formally, each query \( Q \) is
### 6.2. Approach

#### N-gram features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEN(Q)</td>
<td>Number of terms in the phrase Q</td>
</tr>
<tr>
<td>IDF(Q)</td>
<td>Inverse document frequency of Q</td>
</tr>
<tr>
<td>WIG(Q)</td>
<td>Weighted information gain using top-5 retrieved concepts</td>
</tr>
<tr>
<td>QE(Q)</td>
<td>Number of times Q appeared as whole query in the query log</td>
</tr>
<tr>
<td>QP(Q)</td>
<td>Number of times Q appeared as partial query in the query log</td>
</tr>
<tr>
<td>QEQP(Q)</td>
<td>Ratio between QE and QP</td>
</tr>
<tr>
<td>SNIL(Q)</td>
<td>Does a sub-n-gram of Q fully match with any concept label?</td>
</tr>
<tr>
<td>SNCL(Q)</td>
<td>Is a sub-n-gram of Q contained in any concept label?</td>
</tr>
</tbody>
</table>

#### Concept features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INLINKS(c)</td>
<td>The number of concepts linking to c</td>
</tr>
<tr>
<td>OUTLINKS(c)</td>
<td>The number of concepts linking from c</td>
</tr>
<tr>
<td>GEN(c)</td>
<td>Function of depth of c in the SKOS category hierarchy [230]</td>
</tr>
<tr>
<td>CAT(c)</td>
<td>Number of associated categories</td>
</tr>
<tr>
<td>REDIRECT(c)</td>
<td>Number of redirect pages linking to c</td>
</tr>
</tbody>
</table>

#### N-gram + concept features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF(c, Q)</td>
<td>Relative phrase frequency of Q in c, normalized by length of c</td>
</tr>
<tr>
<td>TF_f(c, Q)</td>
<td>Relative phrase frequency of Q in representation f of c, normalized by length of f</td>
</tr>
<tr>
<td>POS_n(Q, c)</td>
<td>Position of nth occurrence of Q in c, normalized by length of c</td>
</tr>
<tr>
<td>SPR(c, Q)</td>
<td>Spread (distance between the last and first occurrences of Q in c)</td>
</tr>
<tr>
<td>TF · IDF(c, Q)</td>
<td>The importance of Q for c</td>
</tr>
<tr>
<td>RIDF(c, Q)</td>
<td>Residual IDF (difference between expected and observed IDF)</td>
</tr>
<tr>
<td>χ²(c, Q)</td>
<td>χ² test of independence between Q in c and in collection Coll</td>
</tr>
<tr>
<td>QCT(c, Q)</td>
<td>Does q contain the label of c?</td>
</tr>
<tr>
<td>TCQ(c, Q)</td>
<td>Does the label of c contain q?</td>
</tr>
<tr>
<td>TEQ(c, Q)</td>
<td>Does the label of c equal q?</td>
</tr>
<tr>
<td>SCORE(c, Q)</td>
<td>Retrieval score of c w.r.t. Q</td>
</tr>
<tr>
<td>RANK(c, Q)</td>
<td>Retrieval rank of c w.r.t. Q</td>
</tr>
</tbody>
</table>

#### History features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCHH(c)</td>
<td>Number of occurrences of label of c appears as query in history</td>
</tr>
<tr>
<td>CCIHH(c)</td>
<td>Number of occurrences of label of c appears in any query in history</td>
</tr>
<tr>
<td>CCIIHHH(c)</td>
<td>Number of times c is retrieved as result for any query in history</td>
</tr>
<tr>
<td>CCCC(c)</td>
<td>Number of times label of c equals title of any result for any query in history</td>
</tr>
<tr>
<td>QCCIHHQ(c)</td>
<td>Number of times title of any result for any query in history contains label of c</td>
</tr>
<tr>
<td>QCQHHQ(c)</td>
<td>Number of times title of any result for any query in history equals Q</td>
</tr>
<tr>
<td>QQCCHHQ(c)</td>
<td>Number of times title of any result for any query in history contains Q</td>
</tr>
<tr>
<td>QCQH(c)</td>
<td>Number of times Q appears as query in history</td>
</tr>
<tr>
<td>QCCHQ(c)</td>
<td>Number of times Q appears in any query in history</td>
</tr>
</tbody>
</table>

| Table 6.3: Features used, grouped by type. Detailed descriptions in Section 6.2.3. |

associated with a ranked list of concepts c and a set of associated relevance assessments for the concepts. The latter is created by considering all concepts that
any annotator used to map $Q$ to $c$. If a concept was not selected by any of the annotators, we consider it to be non-relevant for $Q$. Then, for each query in the set of annotated queries, we consider each combination of n-gram $Q$ and concept $c$ an instance for which we create a feature vector.

The goal of the machine learning algorithm is to learn a function that outputs a relevance status for any new n-gram and concept pair given a feature vector of this new instance. We choose to compare a naive bayes (NB) classifier, with a support vector machine (SVM) classifier and a decision tree classifier (J48)—a set representative of the state-of-the-art in classification. These algorithms will be further introduced in Section 6.3.3.

### 6.2.3 Features Used

We employ several types of features, each associated with either an n-gram, concept, their combination, or the search history. Unless indicated otherwise, when determining the features, we consider any consecutive terms in $Q$ as a phrase, that is, we do not assume term independence.

#### N-gram Features

These features are based on information from an n-gram and are listed in Table 6.3 (first group). $IDF(Q)$ indicates the relative number of concepts in which $Q$ occurs, which is defined as $IDF(Q) = \log(|Coll|/df(Q))$, where $|Coll|$ indicates the total number of concepts and $df(Q)$ the number of concepts in which $Q$ occurs [18]. $WIG(Q)$ indicates the weighted information gain, which was proposed by Zhou and Croft [359] as a predictor of the retrieval performance of a query. It uses the set of all candidate concepts retrieved for this n-gram, $C_Q$, and determines the relative probability of $Q$ occurring in these documents as compared to the collection. Formally:

$$WIG(Q) = \frac{\frac{1}{|C_Q|} \sum_{c \in C_Q} \log(P(Q|c)) - \log(P(Q))}{\log P(Q)}.$$  

$QE(Q)$ and $QP(Q)$ indicate the number of times the n-gram $Q$ appears in the entire query logs as a complete or partial query respectively.

#### Concept Features

Table 6.3 (second group) lists the features related to a DBpedia concept. This set of features is related to the knowledge we have of the candidate concept, such as the number of other concepts linking to or from it, the number of associated categories (the count of the DBpedia property `skos:subject`), and the number of redirect pages pointing to it (the DBpedia property `dbpprop:redirect`).
**N-gram + Concept Features**

This set of features considers the combination of an n-gram and a concept (Table 6.3, third group). We consider the relative frequency of occurrence of the n-gram as a phrase in the Wikipedia article corresponding to the concept, in the separate document representations (title, content, anchor texts, first sentence, and first paragraph of the Wikipedia article), the position of the first occurrence of the n-gram, the distance between the first and last occurrence, and various IR-based measures \([18]\). Of these, \(RIDF\) \([68]\) is the difference between expected and observed IDF for a concept, which is defined as

\[
RIDF(c, Q) = \log \left( \frac{|Coll|}{df(Q)} \right) + \log \left( 1 - \exp \left( \frac{-n(Q,Coll)}{|Coll|} \right) \right).
\]

We also consider whether the label of the concept (\(rdfs:label\)) matches \(Q\) in any way and we include the retrieval score and rank as determined by using Eq. 6.2.

**History Features**

Finally, we consider features based on the previous queries that were issued in the same session (Table 6.3, fourth group). These features indicate whether the current candidate concept or n-gram occurs (partially) in the previously issued queries or retrieved candidate concepts respectively.

In Section 6.4 we compare the effectiveness of the feature types listed above for our task, whilst in Section 6.5.5 we discuss the relative importance of each individual feature.

---

**6.3 Experimental Setup**

In this section we introduce the experimental environment and the experiments that we perform to answer the research questions for this chapter. We start by detailing our data sets and then introduce our evaluation methods and manual assessments.

**6.3.1 Data**

Two main types of data are needed for our experiments, namely search engine queries and a structured knowledge repository. We have access to a set of 264,503 queries issued between 18 November 2008 to 15 May 2009 to the audiovisual catalog maintained by Sound and Vision. Sound and Vision logs the actions of users on the site, generating session identifiers and time stamps. This allows for a series of consecutive queries to be linked to a single search session, where a session is identified using a session cookie. A session is terminated once the user closes the
### 6. Linking Queries to Concepts

#### 6.3.2 Training Data

For training and testing purposes, five assessors were asked to manually map queries to DBpedia concepts using the interface depicted in Figure 6.3. The assessors were presented with a list of sessions and the queries in them. Once a session

<table>
<thead>
<tr>
<th>Session ID</th>
<th>Query ID</th>
<th>Query (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>jyq4navmztg</td>
<td>715681456</td>
<td>santa claus canada</td>
</tr>
<tr>
<td>jyq4navmztg</td>
<td>715681569</td>
<td>santa claus emigrants</td>
</tr>
<tr>
<td>jyq4navmztg</td>
<td>715681598</td>
<td>santa claus australia</td>
</tr>
<tr>
<td>jyq4navmztg</td>
<td>715681633</td>
<td>christmas sun</td>
</tr>
<tr>
<td>jyq4navmztg</td>
<td>715681789</td>
<td>christmas australia</td>
</tr>
<tr>
<td>jyq4navmztg</td>
<td>715681896</td>
<td>christmas new zealand</td>
</tr>
<tr>
<td>jyq4navmztg</td>
<td>715681952</td>
<td>christmas overseas</td>
</tr>
</tbody>
</table>

**Table 6.4:** An example of queries issued in a (partial) session, translated to English.

**Figure 6.3:** Screen dump of the web interface the annotators used to manually link queries to concepts. On the left the sessions, in the middle a full-text retrieval interface, and on the right the made annotations.

This data set is analyzed and described more fully in [142], an example is given in Table 6.4. All queries are Dutch language queries (although we emphasize that nothing in our approach is language dependent). As the “history” of a query, we take all queries previously issued in the same user session. The DBpedia version we use is the most recently issued Dutch language release (3.2). We also downloaded the Wikipedia dump from which this DBpedia version was created (dump date 20080609); this dump is used for all our text-based processing steps and features.
had been selected, they were asked to find the most relevant DBpedia concepts (in the context of the session) for each query therein. Our assessors were able to search through Wikipedia using the fields described in Section 6.2.1. Besides indicating relevant concepts, the assessors could also indicate whether a query was ambiguous, contained a typographical error, or whether they were unable to find any relevant concept at all. For our experiments, we removed all the assessed queries in these “anomalous” categories and were left with a total of 629 assessed queries (out of 998 in total) in 193 randomly selected sessions. In our experiments we primarily focus on evaluating the actual mappings to the LOD cloud and discard queries which the assessors deemed too anomalous to confidently map to any concept. In this subset, the average query length is 2.14 terms per query and each query has 1.34 concepts annotated on average. In Section 6.5.1 we report on the inter-annotator agreement.

6.3.3 Parameters

As to retrieval, we use the entire Wikipedia document collection as background corpus and set $\mu$ to the average length of a Wikipedia article [356], i.e., $\mu = 315$ (cf. Eq. 2.7). Initially, we select the 5 highest ranked concepts as input for the concept selection stage. In Section 6.5.3 we report on the influence of varying the number of highest ranked concepts used as input.

As indicated earlier in Section 6.2.2, we use the following three supervised machine learning algorithms for the concept selection stage: J48, Naive Bayes and Support Vector Machines. The implementations are taken from the Weka machine learning toolkit [344]. J48 is a decision tree algorithm and the Weka implementation of C4.5 [253]. The Naive Bayes classifier uses the training data to estimate the probability that an instance belongs to the target class, given the presence of each feature. By assuming independence between the features these probabilities can be combined to calculate the probability of the target class given all features [154]. SVM uses a sequential minimal optimization algorithm to minimize the distance between the hyperplanes which best separate the instances belonging to different classes, as described in [246]. In the experiments in the next section we use a linear kernel. In Section 6.5.3 we discuss the influence of different parameter settings to see whether fine-grained parameter tuning of the algorithms has any significant impact on the end results.

6.3.4 Testing and Evaluation

We define the mapping of search engine queries to the LOD cloud as a ranking problem. The system that implements a solution to this problem has to return a ranked list of concepts for a given input query, where a higher rank indicates a higher degree of relevance of the concept to the query. The best performing
method puts the most relevant concepts towards the top of the ranking. The assessments described above are used to determine the relevance status of each of the concepts with respect to a query. We employ several measures that were introduced in Chapter 3.

To verify the generalizability of our approach, we perform 10-fold cross validation [344]. This also reduces the possibility of errors being caused by artifacts in the data. Thus, we use 90% of the annotated queries for training and validation and the remainder for testing in each of the folds. The reported scores are averaged over all folds, and all evaluation measures are averaged over the queries used for testing. In Section 6.5.3 we discuss what happens when we vary the size of the folds. For determining the statistical significance of the observed differences between runs we use a one-way ANOVA test to determine if there is a significant difference ($p \leq 0.05$) as introduced in Section 3.2.2.

6.4 Results

In the remainder of this section we report on the experimental results and use them to answer the research questions for this chapter. Here, we compare the following approaches for mapping queries to DBpedia:

(i) a baseline that retrieves only those concepts whose label *lexically matches* the query,

(ii) a *retrieval baseline* that retrieves concepts based solely on their textual representation in the form of the associated Wikipedia article with varying textual fields,

(iii) *n-gram based reranking* that extracts all n-grams from the query and uses machine learning to identify the best concepts, and

(iv) *full query based reranking* that does not extract n-grams, but calculates feature vectors based on the full query and uses machine learning to identify the best concepts.

In the next section we further analyze the results along multiple dimensions, including the effects of varying the number of retrieved concepts in the first stage, varying parameters in the machine learning models, the most informative individual features and types, and the kind of errors that are made by the machine learning algorithms.

6.4.1 Lexical Match

As our first baseline we consider a simple heuristic which is commonly used [12, 28, 94, 114, 142, 200]. For this baseline we select concepts that lexically match
6.4. Results

Table 6.5: An example of the concepts obtained using various lexical matching constraints for the query “joseph haydn” (translated to English). In this case, the annotators only linked the concept Joseph Haydn.

<table>
<thead>
<tr>
<th>Concept</th>
<th>QCL</th>
<th>QCL-LCQ</th>
<th>QCL-LSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joseph Haydn</td>
<td>JOSEPH HAYDN</td>
<td>JOSEPH HAYDN</td>
<td>JOSEPH HAYDN</td>
</tr>
<tr>
<td>Joseph Haydn Operas</td>
<td></td>
<td>JOSEPH HAYDN OPERAS</td>
<td></td>
</tr>
<tr>
<td>Joseph Haydn Symphonies</td>
<td></td>
<td></td>
<td>JOSEPH HAYDN SYMPHONIES</td>
</tr>
</tbody>
</table>

The query, subject to various constraints. This returns concepts where consecutive terms in the rdfs:label are contained in the query or vice versa. An example for the query “joseph haydn” is given in Table 6.5. We then rank the concepts based on the language modeling score of their associated Wikipedia article given the query (cf. Eq. 6.2).

Table 6.6: Lexical match baseline results using lexical matching between labels and query to select concepts.

<table>
<thead>
<tr>
<th>Method</th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>QCL</td>
<td>0.3956</td>
<td>0.3140</td>
<td>0.4282</td>
<td>0.4117</td>
<td>0.4882</td>
</tr>
<tr>
<td>QCL-LCQ</td>
<td>0.4286</td>
<td>0.3485</td>
<td>0.4881</td>
<td>0.4564</td>
<td>0.5479</td>
</tr>
<tr>
<td>QCL-LSO</td>
<td>0.4160</td>
<td>0.2747</td>
<td>0.3435</td>
<td>0.3775</td>
<td>0.4160</td>
</tr>
<tr>
<td>Oracle</td>
<td>0.5808</td>
<td>0.4560</td>
<td>0.5902</td>
<td>0.5380</td>
<td>0.6672</td>
</tr>
</tbody>
</table>

Table 6.6 shows the scores when using lexical matching for mapping search engine queries. The results in the first row are obtained by only considering the concepts whose label is contained in the query (QCL). This is a frequently taken but naive approach and does not perform well, achieving a P1 score of under 40%. The second row relaxes this constraint and also selects concepts where the query is contained in the concept label (QCL-LCQ). This improves the performance somewhat.

One issue these approaches might have, however, is that they might match parts of compound terms. For example, the query “brooklyn bridge” might not only match the concept BROOKLYN BRIDGE but also the concepts BROOKLYN and BRIDGE. The approach taken for the third row (QCL-LSO) therefore extracts all n-grams from the query, sorts them by the number of terms, and checks whether the label is contained in each of them. If a match is found, the remaining, smaller n-grams are skipped.

The last row (“oracle”) shows the results when we initially select all concepts whose terms in the label matches with any part of the query. Then, we keep only those concepts that were annotated by the assessors. As such, the performance of this run indicates the upper bound on the performance that lexical matching might obtain. From these scores we conclude that, although lexical matching
is a common approach for matching unstructured text with structured data, it
does not perform well for our task and we need to consider additional kinds of
information pertaining to each concept.

### 6.4.2 Retrieval Only

As our second baseline, we take the entire query as issued by the user and em-
ploy Eq. 6.2 to rank DBpedia concepts based on their textual representation; this
technique is similar to using a search engine and performing a search within
Wikipedia. We use either the textual contents of the Wikipedia article (“content-
only”—which includes only the article’s text) or a combination of the article’s text,
the title, and the anchor texts of incoming links (“full text”).

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>full text</td>
<td>0.5636</td>
<td>0.5216</td>
<td>0.6768</td>
<td>0.6400</td>
<td>0.7535</td>
</tr>
<tr>
<td>content-only</td>
<td>0.5510</td>
<td>0.5134</td>
<td>0.6632</td>
<td>0.6252</td>
<td>0.7363</td>
</tr>
</tbody>
</table>

**Table 6.7:** Results for the retrieval only baseline which ranks concepts using the entire
query Q and either the content of the Wikipedia article or the full text associated with
each DBpedia concept (including title and anchor texts of incoming hyperlinks).

Table 6.7 shows the results of this method. We note that including the title and
anchor texts of the incoming links results in improved retrieval performance over-
all. This is a strong baseline; on average, over 65% of the relevant concepts are
correctly identified in the top-5 and, furthermore, over 55% of the relevant con-
cepts are retrieved at rank 1. The success rate indicates that for 75% of the queries
at least one relevant concept is retrieved in the top-5. In Section 6.5.2 we further
discuss the relative performance of each textual representation as well as various
combinations.

### 6.4.3 N-gram based Concept Selection

Table 6.8 (last row) shows the concepts obtained for the second baseline and the
query “challenger wubbo ockels.” Here, two relevant concepts are retrieved at
ranks 1 and 4. When we look at the same results for all possible n-grams in the
query, however, one of the relevant concepts is retrieved at the first position for
each n-gram. This example and the one given earlier in Table 6.2 suggest that
it will be beneficial to consider all possible n-grams in the query. In this section
we report on the results of extracting n-grams from the query, generating fea-
tures for each, and subsequently applying machine learning algorithms to decide
which of the suggested concepts to keep. The features used here are described in
Section 6.2.2.
6.4. Results

Table 6.8: An example of the concepts obtained when using retrieval only for the
n-grams in the query “challenger wubbo ockels” (translated to English), ranked by re-
trieval score. Concepts annotated by the human annotators for this query in boldface.

<table>
<thead>
<tr>
<th>N-gram</th>
<th>Candidate concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>challenger</td>
<td>SPACE SHUTTLE CHALLENGER; CHALLENGER; BOMBARDIER CHALLENGER;</td>
</tr>
<tr>
<td></td>
<td>STS-61-A; STS-9</td>
</tr>
<tr>
<td>wubbo</td>
<td>WUBBO OCKELS; SPACELAB; CANON OF GRONINGEN; SUPERBUS;</td>
</tr>
<tr>
<td></td>
<td>ANDRÉ KUIPERS</td>
</tr>
<tr>
<td>ockels</td>
<td>WUBBO OCKELS; SPACELAB; SUPERBUS; CANON OF GRONINGEN;</td>
</tr>
<tr>
<td></td>
<td>ANDRÉ KUIPERS</td>
</tr>
<tr>
<td>challenger wubbo</td>
<td>WUBBO OCKELS; STS-61-A; SPACE SHUTTLE CHALLENGER; SPACELAB;</td>
</tr>
<tr>
<td></td>
<td>STS-9</td>
</tr>
<tr>
<td>wubbo ockels</td>
<td>WUBBO OCKELS; SPACELAB; SUPERBUS; CANON OF GRONINGEN;</td>
</tr>
<tr>
<td></td>
<td>ANDRÉ KUIPERS</td>
</tr>
<tr>
<td>challenger wubbo</td>
<td>WUBBO OCKELS; STS-61-A; SPACE SHUTTLE CHALLENGER; SPACELAB;</td>
</tr>
<tr>
<td></td>
<td>STS-9</td>
</tr>
</tbody>
</table>

Table 6.9: Results for n-gram based concept selection. ▲ ▼ and ◦ indicate that a
score is significantly better, worse, or statistically indistinguishable respectively. The
leftmost symbol represents the difference with the baseline, the next with the J48
run, and the rightmost with the NB run.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>0.5636</td>
<td>0.5216</td>
<td>0.6768</td>
<td>0.6400</td>
<td>0.7535</td>
</tr>
<tr>
<td>J48</td>
<td>0.6586 ▲</td>
<td>0.5648 ◦</td>
<td>0.7253 ◦</td>
<td>0.7348 ▲</td>
<td>0.7989 ◦</td>
</tr>
<tr>
<td>NB</td>
<td>0.4494 ▼▼</td>
<td>0.4088 ▼▼</td>
<td>0.6948 ◦ ◦</td>
<td>0.7278 ◦ ◦</td>
<td>0.7710 ◦ ◦</td>
</tr>
<tr>
<td>SVM</td>
<td>0.7998 ▲▲▲</td>
<td>0.6718 ◾▲</td>
<td>0.7556 ◦ ◦ ◦</td>
<td>0.8131 ▲ ◾ ◾</td>
<td>0.8240 ◦ ◦ ◦</td>
</tr>
</tbody>
</table>

Table 6.9 shows the results of applying the machine learning algorithms on the
extracted n-gram features. We note that J48 and SVM are able to improve upon
the baseline results from the previous section, according to all metrics. The Naive
Bayes classifier performs worse than the baseline in terms of P1 and R-precision.
SVM clearly outperforms the other algorithms and is able to obtain scores that are
very high, significantly better than the baseline on all metrics. Interestingly, we
see that the use of n-gram based reranking has both a precision enhancing effect
for J48 and SVM (the P1 and MRR scores go up) and a recall enhancing effect.

6.4.4 Full Query-based Concept Selection

Next, we turn to a comparison of n-gram based and full-query based concept
selection. Using the full-query based concept selection method, we take each
query as is (an example is given in the last row of Table 6.8) and generate a
single ranking to which we apply the machine learning models.
Table 6.10 shows the results when only the full query is used to generate a ranked list of concepts. We again observe that SVM significantly outperforms J48, NB, and the baseline. For both the J48 and NB classifiers we see a significant increase in precision (P1). Naive Bayes, for which precision was significantly worse than all other methods on n-gram based concept selection, performs significantly better than the other machine learning algorithms using full query reranking. The increase in precision comes at a loss in recall for NB. The MRR scores for J48 are no longer significantly higher than the baseline. Both J48 and NB produce fewer false positives when classifying full query data instead of n-gram based query data. This means that fewer incorrect concepts end up in the ranking which in turn results in a higher precision.

Interestingly, this increase in precision is not accompanied by a loss in recall. In particular, the SVM classifier is able to distinguish between correct and incorrect concepts when used on the full query data. These scores are the highest obtained so far and this approach is able to return almost 90% of all relevant concepts. This result is very encouraging and shows that the approach taken handles the mapping of search engine queries to the LOD cloud extremely well.

6.5 Discussion

In this section, we further analyze the results presented in the previous section and answer the remaining research questions. We first look at the inter-annotator agreement between the assessors. We then turn to the performance of the different textual representations of the Wikipedia content that we use. Further, we consider the robustness of the performance of our methods with respect to various parameter settings, provide an analysis of the influence of the feature types on the end results, and also report on the informativeness of the individual features. We conclude with an error analysis to see which queries are intrinsically difficult to map to the DBpedia portion of the LOD cloud.

Unless indicated otherwise, all results on which we report in this section use the best performing approach from the previous section, i.e., the SVM classifier with a linear kernel using the full queries (with ten-fold cross-validation when applicable).
6.5. Discussion

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>full text</td>
<td>0.5636</td>
<td>0.5216</td>
<td>0.6768</td>
<td>0.6400</td>
<td>0.7535</td>
</tr>
<tr>
<td>content</td>
<td>0.5510</td>
<td>0.5134</td>
<td>0.6632</td>
<td>0.6252</td>
<td>0.7363</td>
</tr>
<tr>
<td>title</td>
<td>0.5651</td>
<td>0.5286</td>
<td>0.6523</td>
<td>0.6368</td>
<td>0.7363</td>
</tr>
<tr>
<td>anchor</td>
<td><strong>0.6122</strong></td>
<td><strong>0.5676</strong></td>
<td><strong>0.7219</strong></td>
<td><strong>0.6922</strong></td>
<td><strong>0.8038</strong></td>
</tr>
<tr>
<td>first sentence</td>
<td>0.5495</td>
<td>0.5106</td>
<td>0.6523</td>
<td>0.6203</td>
<td>0.7268</td>
</tr>
<tr>
<td>first paragraph</td>
<td>0.5447</td>
<td>0.5048</td>
<td>0.6454</td>
<td>0.6159</td>
<td>0.7190</td>
</tr>
<tr>
<td>title + content</td>
<td>0.5604</td>
<td>0.5200</td>
<td>0.6750</td>
<td>0.6357</td>
<td>0.7353</td>
</tr>
<tr>
<td>title + anchor</td>
<td>0.5934</td>
<td>0.5621</td>
<td>0.7164</td>
<td>0.6792</td>
<td>0.7991</td>
</tr>
<tr>
<td>title + content + anchor</td>
<td>0.5714</td>
<td>0.5302</td>
<td>0.6925</td>
<td>0.6514</td>
<td>0.7724</td>
</tr>
<tr>
<td>title + 1st sentence + anchor</td>
<td>0.5856</td>
<td>0.5456</td>
<td>0.6965</td>
<td>0.6623</td>
<td>0.7755</td>
</tr>
<tr>
<td>title + 1st paragraph + anchor</td>
<td>0.5777</td>
<td>0.5370</td>
<td>0.6985</td>
<td>0.6566</td>
<td>0.7771</td>
</tr>
</tbody>
</table>

| Table 6.11: Results of ranking concepts based on the full query using different textual representations of the Wikipedia article associated with each DBpedia concept. |

6.5.1 Inter-annotator Agreement

To assess the agreement between annotators, we randomly selected 50 sessions from the query log for judging by all annotators. We consider each query-concept pair to be an item of analysis for which each annotator expresses a judgment (“a good mapping” or “not a good mapping”) and on which the annotators may or may not agree. However, our annotation tool does not produce any explicit labels of query-concept pairs as being “incorrect,” since only positive (“correct”) judgments are generated by the mappings. Determining the inter-annotator agreement on these positive judgments alone might bias the results and we adopt a modified approach to account for the missing non-relevance information, as we will now explain.

We follow the same setup as used for the results presented earlier by considering 5 concepts per query. In this case, the 5 concepts were sampled such that at least 3 were mapped (judged correct) by at least one of the annotators; the remaining concepts were randomly selected from the incorrect concepts. We deem a concept “incorrect” for a query if the query was not mapped to the concept by any annotator. For the queries where fewer than 3 correct concepts were identified, we increased the number of incorrect concepts to keep the total at 5. The rationale behind this approach is that each annotator looks at at least 5 concepts and selects the relevant ones. The measure of inter-annotator agreement that we are interested in is determined, then, on these 5 concepts per query. Also similar to the results reported earlier, we remove the queries in the “anomalous” categories.

The value for Cohen’s $\kappa$ is 0.5111, which indicates fair overall agreement ($\kappa$ ranges from –1 for complete disagreement to +1 for complete agreement) [13, 77, 179]. Krippendorff’s $\alpha$ is another statistic for measuring inter-annotator agree-
ment that takes into account the probability that observed variability is due to chance. Moreover, it does not require that each annotator annotates each document [13, 123]. The value of $\alpha$ is 0.5213. As with the $\kappa$ value, this indicates a fair agreement between annotators. It is less, however, than the level recommended by Krippendorff for reliable data ($\alpha = 0.8$) or for tentative reliability ($\alpha = 0.667$). The values we obtain for $\alpha$ and $\kappa$ are therefore an indication as to the nature of relevance with respect to our task. What one person deems a viable mapping given his or her background, another might find not relevant. Voorhees [328] has shown, however, that moderate inter-annotator agreement can still yield reliable comparisons between approaches (in her case TREC information retrieval runs, in our case different approaches to the mapping task) that are stable when one set of assessments is substituted for another. This means that, although the absolute inter-annotator scores indicate a fair agreement, the system results and comparisons thereof that we obtain are valid.

6.5.2 Textual Concept Representations

One of our baselines ranks concepts based on the full textual representation of each DBpedia concept, as described in Section 6.4.1. Instead of using the full text, we evaluate what the results are when we rank concepts based on each individual textual representation and based on combinations of fields. Table 6.11 lists the results. As per the Wikipedia authoring guidelines [342], the first sentence and paragraph should serve as an introduction to, and summary of, the important aspects of the contents of the article. In Table 6.11, we have also included these fields. From the table we observe that the anchor texts emerge as the best descriptor of each concept and using this field on its own obtains the highest absolute retrieval performance. However, the highest scores obtained using this approach are still significantly lower than the best performing machine learning method reported on earlier.

6.5.3 Robustness

Next, we discuss the robustness of our approach. Specifically, we investigate the effects of varying the number of retrieved concepts in the first stage, of varying the size of the folds, of balancing the relative amount of positive and negative examples in the training data, and the effect of varying parameters in the machine learning models.

Number of Concepts

The results in Section 6.4 were obtained by selecting the top 5 concepts from the first stage for each query, under the assumption that 5 concepts would give a good balance between recall and precision (motivated by the fact there are 1.34
6.5. Discussion

concepts annotated per query on average). Our intuition was that, even if the initial stage did not place a relevant concept at rank 1, the concept selection stage could still consider this concept as a candidate (given that it appeared somewhere in the top 5). We now test this assumption by varying the number of concepts returned for each query.

Figure 6.4 shows the effect of varying the number of retrieved concepts ($K$) in the first stage on various retrieval measures. On nearly all metrics the best performance is achieved when using the top 3 concepts from the initial stage for concept selection, although the absolute difference between using 3 and 5 terms is minimal for most measures. As we have observed above, most relevant concepts are already ranked very high by the initial stage. Further, from the figure we conclude that using only the top 1 is not enough and results in the worst performance. In general, one might expect recall to improve when the number of concepts grows. However, since each query only has 1.34 concepts annotated on average, recall can not improve much when considering larger numbers of candidate concepts. Finally, increasing the number of concepts mainly increases the number of non-relevant concepts in the training data, which may result in a bias towards classifying concepts as not relevant by a machine learning algorithm.

Balancing the Training Set

Machine learning algorithms are sensitive to the distribution of positive and negative instances in the training set. The results reported so far do not perform any kind of resampling of the training data and take the distribution of the class labels (whether the current concept is selected by the assessors) as is.
### 6. Linking Queries to Concepts

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>balanced</td>
<td>0.5777</td>
<td>0.4383</td>
<td>0.5436</td>
<td>0.5960</td>
<td>0.6150</td>
</tr>
<tr>
<td>random sampling</td>
<td>0.8833</td>
<td>0.8666</td>
<td>0.8975</td>
<td>0.8406</td>
<td>0.9053</td>
</tr>
</tbody>
</table>

**Table 6.12:** Comparison of sampling methods.

In order to determine whether reducing the number of non-relevant concepts in the training data has a positive effect on the performance, we experiment using a balanced and a randomly distributed training set. The balanced set reduces the number of negative examples such that the training set contains as many positive examples as negative examples. On the other hand, the random sampled set follows the empirical distribution in the data. Table 6.12 shows that balancing the training set causes performance to drop. We thus conclude that including a larger number of negative examples has a positive effect on retrieval performance and that there is no need to perform any kind of balancing for our task.

### Splitting the Data

Ideally, the training set used to train the machine learning algorithms is large enough to learn a model of the data that is sufficiently discriminative; also, a test set should be large enough to test whether the model generalizes well to unseen instances.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>50-50</td>
<td>0.8809</td>
<td>0.8601</td>
<td>0.8927</td>
<td>0.8338</td>
<td>0.9016</td>
</tr>
<tr>
<td>75-25</td>
<td>0.8812</td>
<td>0.8599</td>
<td>0.8927</td>
<td>0.8344</td>
<td>0.9015</td>
</tr>
<tr>
<td>90-10</td>
<td>0.8833</td>
<td>0.8666</td>
<td>0.8975</td>
<td>0.8406</td>
<td>0.9053</td>
</tr>
</tbody>
</table>

**Table 6.13:** Comparison of using different sizes for the training and test sets used for cross-validation. A 50-50 split uses the smallest training set (training and test set are equally sized), a 75-25 split uses 75% for training and 25% for testing, a 90-10 split uses 90% for training and 10% for testing.

Table 6.13 shows the results when we vary the size of the folds used for cross-validation using the SVM classifier on the full query based concept selection. Here, we compare the 90-10 split reported on above so far with a 50-50 and a 75-25 split. From this table we observe that there is no significant difference between the results on various splits. In practical terms this means that the amount of training data can be greatly reduced, without a significant loss in performance. This in turn means that the labor-intensive, human effort of creating annotations can be limited to a few hundred annotations in order to achieve good performance.
Machine Learning Model Parameters

Next, we look at important parameters of the three machine learning algorithms we evaluate.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full query based concept selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linear</td>
<td>0.8833</td>
<td>0.8666</td>
<td>0.8975</td>
<td>0.8406</td>
<td>0.9053</td>
</tr>
<tr>
<td>gaussian</td>
<td>0.8833</td>
<td>0.8666</td>
<td>0.8975</td>
<td>0.8406</td>
<td>0.9053</td>
</tr>
<tr>
<td>polynomial</td>
<td>0.8738</td>
<td>0.7859</td>
<td>0.8415</td>
<td>0.8364</td>
<td>0.8876</td>
</tr>
<tr>
<td>N-gram based concept selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linear</td>
<td>0.7998</td>
<td>0.6718</td>
<td>0.7556</td>
<td>0.8131</td>
<td>0.8240</td>
</tr>
<tr>
<td>gaussian</td>
<td>0.8241</td>
<td>0.6655</td>
<td>0.7849</td>
<td>0.8316</td>
<td>0.8641</td>
</tr>
<tr>
<td>polynomial</td>
<td>0.7967</td>
<td>0.6251</td>
<td>0.7660</td>
<td>0.8205</td>
<td>0.8589</td>
</tr>
</tbody>
</table>

Table 6.14: Comparison of using different kernels for the SVM machine learning algorithm.

Table 6.14 shows the results of using different kernels for the SVM classifier, specifically a linear, a gaussian, and a polynomial kernel. On the full query data there is no difference between the linear and gaussian kernel and on the n-gram data there is only a small difference. The polynomial kernel performs the worst in both cases, but again the difference is insignificant as compared to the results attained using the other kernels. The values listed in Table 6.14 are obtained using the optimal parameter settings for the kernels. Figure 6.5 (b) shows a sweep of the complexity parameter for the gaussian kernel. A higher degree of complexity penalizes non-separable points and leads to overfitting, while if the value is too low SVM is unable to learn a discriminative model. For the polynomial kernel we limited our experiments to a second order kernel, as the increase in training times on higher order kernels made further experimentation prohibitive. The fact that there is little difference between the results of using various kernels shows that, for the purpose of reranking queries, a simple linear model is enough to achieve optimal or close to optimal performance. A more complex model leads to limited or no improvement and increased training times.

Table 6.15 shows the results of binning versus kernel density estimation (using a gaussian kernel). As was the case with SVM, there is only a small difference between the results on the full query data. The results on the n-gram data do show a difference; binning performs better in terms of recall while kernel density estimation achieves higher precision, which is probably caused by the kernel method overfitting the data.

Figure 6.5 (a) shows the effect of varying the level of pruning for the J48 algorithm on the full query data, where a low number relates to more aggressive
6. Linking Queries to Concepts

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full query based concept selection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>binning</td>
<td>0.6925</td>
<td>0.5897</td>
<td>0.6865</td>
<td>0.6989</td>
<td>0.7626</td>
</tr>
<tr>
<td>kernel</td>
<td>0.6897</td>
<td>0.5973</td>
<td>0.6882</td>
<td>0.6836</td>
<td>0.7455</td>
</tr>
<tr>
<td><strong>N-gram based concept selection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>binning</td>
<td>0.4494</td>
<td>0.4088</td>
<td>0.6948</td>
<td>0.7278</td>
<td>0.7710</td>
</tr>
<tr>
<td>kernel</td>
<td>0.5944</td>
<td>0.3236</td>
<td>0.4884</td>
<td>0.5946</td>
<td>0.6445</td>
</tr>
</tbody>
</table>

Table 6.15: Comparison of using different probability density estimation methods for the NB classifier.

Figure 6.5: (a) The effect of adjusting the complexity parameter for SVM with a gaussian kernel. Note that the x-axis is on a log scale. (b) The effect of adjusting the pruning parameter for the J48 learning algorithm. A lower number means more aggressive pruning.

An exploration of the machine learning model parameters shows that SVM is the best classifier for our task: even with optimized parameters the Naive Bayes and J48 classifiers do not achieve better results.

6.5.4 Feature Types

In Section 6.2.3 we identified four groups of features, relating to the n-gram (“N”), concept (“C”), their combination (“N+C”), or the session history (“H”). We will now zoom in on the performance of these groups. To this end we perform an ablation experiment, where each of these groups is removed from the training data.
6.5. Discussion

<table>
<thead>
<tr>
<th>Excluded feature types</th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>–</td>
<td>0.7998</td>
<td>0.6718</td>
<td>0.7556</td>
<td>0.8131</td>
<td>0.8240</td>
</tr>
<tr>
<td>H</td>
<td>0.6848</td>
<td>0.5600</td>
<td>0.6285</td>
<td>0.6902</td>
<td>0.6957</td>
</tr>
<tr>
<td>C</td>
<td>0.4844</td>
<td>0.3895</td>
<td>0.4383</td>
<td>0.4875</td>
<td>0.4906</td>
</tr>
<tr>
<td>H; C</td>
<td>0.2233</td>
<td>0.1233</td>
<td>0.1733</td>
<td>0.2233</td>
<td>0.2233</td>
</tr>
</tbody>
</table>

Table 6.16: Results of removing specific feature types from the training data for the SVM classifier and n-gram based concept selection. ▼ and ◦ indicate that a score is significantly worse or statistically indistinguishable respectively. The leftmost symbol represents the difference with the all features run, the next with the without history features run, and the rightmost symbol the without concept features run.

<table>
<thead>
<tr>
<th>Excluded feature types</th>
<th>P1</th>
<th>R-prec</th>
<th>Recall</th>
<th>MRR</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>–</td>
<td>0.8833</td>
<td>0.8666</td>
<td>0.8975</td>
<td>0.8406</td>
<td>0.9053</td>
</tr>
<tr>
<td>H; C</td>
<td>0.8833</td>
<td>0.8666</td>
<td>0.8975</td>
<td>0.8406</td>
<td>0.9053</td>
</tr>
<tr>
<td>N; N+C</td>
<td>0.1000</td>
<td>0.0000</td>
<td>0.0500</td>
<td>0.1000</td>
<td>0.1000</td>
</tr>
<tr>
<td>N+C</td>
<td>0.0556</td>
<td>0.0222</td>
<td>0.0370</td>
<td>0.0556</td>
<td>0.0556</td>
</tr>
<tr>
<td>H; N+C</td>
<td>0.0333</td>
<td>0.0000</td>
<td>0.0167</td>
<td>0.0333</td>
<td>0.0333</td>
</tr>
</tbody>
</table>

Table 6.17: Results of removing specific feature types for the SVM classifier and full query based concept selection. Not all possible combinations are included in the results; all unlisted combinations have either scores of zero or the same score as when using all feature types. The leftmost symbol represents the difference with the all features run, the next with the without n-gram+concept and n-gram features run, and the rightmost symbol the without n-gram+concept features run.

N-gram based Concept Selection

Table 6.16 shows the results using n-gram based concept selection. It turns out that both the n-gram specific and n-gram + concept specific features are required for successful classification: when these groups are removed, none of the relevant concepts are identified. From this table we further observe that removing the history features results in a drop in performance, albeit a small one. When the concept features are removed, the resulting performance drops even further and their combined removal yields very low scores. So, although some feature types contribute more to the final performance, each is needed to arrive at the highest scores.

Full-query Based Concept Selection

Table 6.17 shows the results using full-query based concept selection. In this case, the effect of removing both history and concept based features does not influence the results at all. This can in part be explained by the fact that most history fea-
tures are based on the counts of the query in various parts of the session. Since we now have a single n-gram (the full query), these counts turn into binary features and may therefore be less discriminative. This is in stark contrast with the n-gram based features that do have a significant effect on performance on all metrics. Similar to the n-gram based data, these features are essential for full query based concept selection. Finally, we observe that there are some dependencies among the types of features. When we remove both the n-gram+concept features and the history features, the performance is worse than when we remove only the n-gram+concept features (although not significantly so).

**Upshot**

In sum, all feature types contribute to the performance in the case of n-gram based concept selection. The highest scores are obtained, however, using full query based concept selection. In this case, the history and concept based features do not contribute to the results.

### 6.5.5 Feature Selection

Several methods exist for automatically determining the most informative features given training instances and their class labels. In this section we report on using an information gain based algorithm for feature selection [350].

<table>
<thead>
<tr>
<th>N-gram based concept selection</th>
<th>Full query based concept selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.119 RANK(c, Q)</td>
<td>0.190 RANK(c, Q)</td>
</tr>
<tr>
<td>0.107 ID</td>
<td>0.108 TEQ(c, Q)</td>
</tr>
<tr>
<td>0.052 INLINKS(c)</td>
<td>0.080 INLINKS(c)</td>
</tr>
<tr>
<td>0.040 TF_{anchor}(c,Q)</td>
<td>0.056 ID</td>
</tr>
<tr>
<td>0.038 OUTLINKS(c)</td>
<td>0.041 OUTLINKS(c)</td>
</tr>
<tr>
<td>0.037 TF_{title}(c,Q)</td>
<td>0.033 SCORE(c,Q)</td>
</tr>
<tr>
<td>0.031 TEQ(c,Q)</td>
<td>0.025 REDIRECT(c)</td>
</tr>
</tbody>
</table>

**Table 6.18:** Results of calculating the information gain with respect to the class label for all features (truncated after 7 features). The higher this score, the more informative a feature is.

Table 6.18 shows the features with the highest information gain values for both n-gram and full query based reranking. The rank at which the retrieval framework puts a concept with respect to an n-gram is most informative. Also, the number of in- and outlinks, and whether the n-gram matches the concept’s label are good indicators of the relevance status of a concept. ID is the internal identifier of each concept and not a feature that we explicitly implemented. However, it turns out that some DBpedia concepts have a higher a priori probability of getting selected. Indeed, in our manually created assessments 854 concepts are identified, 505
of which are unique; some of the repetitions are caused because of a persisting information need in the user sessions: when a user rewrites her query by adding or changing part of the query, the remaining concepts remain the same and were annotated as such.

For n-gram based concept selection, the number of in- and outlinks, rank, ID, and whether the concept label equals the query are also strong indicators of relevance for given phrase and concept. Added to these, however, are the frequency of the n-gram in the title or in the anchor texts in this case.

6.5.6 Error Analysis

Finally, we provide an analysis of the errors that were made by the machine learning algorithms. To this end, we first examine the relationship between mapping performance and the frequency of the query in the entire query log. We separate all queries in two groups, one for those queries where our approach successfully mapped concepts and one where it failed. In the first group, the average query frequency is 23 (median 2, std. dev. 85.6). In the second group, the average frequency is 6 (median 1, std. dev. 19.5). So, although it seems our approach works best for frequently occurring queries, the high standard deviation indicates that the frequencies are spread out over a large range of values.

Table 6.19 shows examples of correctly and incorrectly mapped queries, together with their relative frequency of occurrence in the entire query log. This table provides further indication that the frequency of a query is not a determining factor in the successful outcome of our method. Rather, it is the retrieval framework that puts concepts that contain query terms with a relatively high frequency in the top of the ranking. For example, besides being the queen of the Netherlands, Beatrix is also the name of one of the characters in the movie Kill Bill.

To further investigate the errors being made, we have manually inspected the output of the algorithms and classified the errors into several classes. Since we formulate the mapping search engine queries to LOD task as a ranking problem, we are primarily interested in the false positives—these are the concepts the classifier identified as correct for a query but which the annotators did not select. The classes in which the classifiers make the most mistakes are:

- **ambiguous (5%)** A query may map to more than one concept and the annotators did not explicitly mark the query as being ambiguous.

- **match with term in content (15%)** Part of the query occurs frequently in the textual representation of the concept, while the concept itself is not relevant. For example, the query “red lobster” matches with the concept **RED CROSS**.
6. Linking Queries to Concepts

Table 6.19: Examples of correctly and incorrectly mapped queries (translated to English), with their relative frequency of occurrence in the entire query log. Concepts annotated by the human annotators in boldface. Wouter Bos is a Dutch politician and Beatrix is the Dutch queen.

<table>
<thead>
<tr>
<th>Freq. ($\times 10^{-4}$)</th>
<th>Query</th>
<th>Mapped concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Well performing queries</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64.0 %</td>
<td>wouter bos</td>
<td>WOUTER BOS</td>
</tr>
<tr>
<td>18.9 %</td>
<td>moon landing</td>
<td>MOON LANDING</td>
</tr>
<tr>
<td>22.2 %</td>
<td>vietnam war</td>
<td>VIETNAM WAR</td>
</tr>
<tr>
<td>1.67 %</td>
<td>simple minds</td>
<td>SIMPLE MINDS</td>
</tr>
<tr>
<td>1.11 %</td>
<td>spoetnik</td>
<td>SPOETNIK</td>
</tr>
<tr>
<td>1.11 %</td>
<td>sarkozy agriculture</td>
<td>NICOLAS SARKOZY; AGRICULTURE</td>
</tr>
<tr>
<td>0.557 %</td>
<td>universal soldier</td>
<td>UNIVERSAL SOLDIER</td>
</tr>
<tr>
<td><strong>Poorly performing queries</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57.9 %</td>
<td>gaza</td>
<td>DOROTHEUS OF GAZA</td>
</tr>
<tr>
<td>2.78 %</td>
<td>wedding beatrix</td>
<td>KILL BILL; WILLEM OF LUXEMBURG; MASAKO OWADA</td>
</tr>
<tr>
<td>1.11 %</td>
<td>poverty netherlands 1940s</td>
<td>1940-1949; IMMIGRATION POLICY; MEXICAN MIRACLE</td>
</tr>
<tr>
<td>0.557 %</td>
<td>poverty thirties</td>
<td>1930-1939; HUMAN DEVELOPMENT INDEX</td>
</tr>
<tr>
<td>0.557 %</td>
<td>rabin funeral</td>
<td>BILL CLINTON; HUSSEIN OF JORDAN</td>
</tr>
<tr>
<td>0.557 %</td>
<td>eurovision songfestival 1975</td>
<td>EUROVISION SONGFESTIVAL; MELODIFESTIVALEN 1975</td>
</tr>
<tr>
<td>0.557 %</td>
<td>cold war netherlands</td>
<td>COLD WAR; WATCHTOWER; WESTERN BLOC</td>
</tr>
</tbody>
</table>

- **substring (4%)** In this case a substring of the query is matched to a concept, for example the concept BROOKLYN is selected for the query “brooklyn bridge.” While this might be considered an interesting suggestion, it is incorrect since the annotators did not label it so.

- **too specific—child selected (10%)** A narrower concept is selected where the broader is correct. For example, when the concept EUROVISION SONGFESTIVAL 1975 is selected for the query “songfestival.”

- **too broad—parent selected (6%)** The inverse of the previous case. For example, the concept EUROVISION is selected for the query “eurovision songfestival 2008.”

- **related (10%)** A related concept is selected. For example when the concept CUBA CRISIS is selected for the query “cuba kennedy.” Another example is the concept INDUSTRIAL DESIGN for the query “walking frame.”

- **sibling (4%)** A sibling is selected, e.g., EUROVISION SONGFESTIVAL 1975 instead of EUROVISION SONGFESTIVAL 2008.
• **same concept, different label (6%)** When there is more than one applicable concept for the query and the annotators used only one, e.g., in the case of New York and New York City.

• **erroneous (25%)** The final category is where the classifiers selected the right concept, but it was missed by the annotators.

From these classes we conclude that the largest part of the errors are not attributable to the machine learning algorithms but rather to incomplete or imperfect human annotations. Another class of interesting errors is related to the IR framework we use. This sometimes produces “fuzzy” matches when the textual representation of the concept contains the query terms with a high frequency (e.g., selecting Cuba Crisis for the query “cuba kennedy”). Some of these errors are not wrong per se, but interesting since they do provide mappings to related concepts. Marking them as wrong is partly an artifact of our evaluation methodology, which determines a priori which concepts are relevant to which queries, so as to ensure the reusability our evaluation resources. We have chosen this approach also for practical reasons, since the same annotations are used to generate the training data for the machine learners. In future work, we intend to perform a large-scale post-hoc evaluation in which we directly evaluate the generated mappings to the LOD cloud.

### 6.6 Summary and Conclusions

In this chapter we have introduced the task of mapping search engine queries to the LOD cloud and presented a method that uses supervised machine learning methods to learn which concepts are used in a query. We consider DBpedia to be an integral part of, and interlinking hub for, the LOD cloud, which is why we focused our efforts on mapping queries to this ontology.

Our approach first retrieves and ranks candidate concepts using a framework based on language modeling for information retrieval. We then extract query, concept, and history-specific feature vectors for these candidate concepts. Using manually created annotations we inform a machine learning algorithm, which then learns how to best select candidate concepts given an input query.

Our results were obtained using the Dutch version of DBpedia and queries from a log of the Netherlands Institute for Sound and Vision. Although these resources are in Dutch, the framework we have presented is language-independent. Moreover, the approach is also generic in that several of the employed features can be used with ontologies other than DBpedia.

In this chapter we have reported upon extensive analyses to answer the following research questions.
RQ 3. Can we successfully address the task of mapping search engine queries to concepts using a combination of information retrieval and machine learning techniques? A typical approach for mapping text to concepts is to apply some form of lexical matching between concept labels and terms, typically using the context of the text for disambiguation purposes. What are the results of applying this method to our task? What are the results when using a purely retrieval-based approach? How do these results compare to those of our proposed method?

Our best performance was obtained using Support Vector Machines and features extracted from the full input queries. The best performing run was able to locate almost 90% of the relevant concepts on average. Moreover, this particular run achieved a precision@1 of 89%, meaning that for this percentage of queries the first suggested concept was relevant.\footnote{Our results can be partially explained by the fact that we have decided to focus on the quality of the suggested concepts and as such removed “anomalous” queries from the evaluation, i.e., queries with typos or that were too ambiguous or vague for human assessors to be able to assign a concept to. Ideally, one would have a classifier at the very start of the query linking process which would predict whether a query falls in one of these categories. Implementing and evaluating such a classifier is an interesting—and challenging—research topic in itself but falls beyond the scope of this thesis.}

We find that simply performing a lexical match between the queries and concepts did not perform well and neither did using retrieval alone, i.e., omitting the concept selection stage. When applying our proposed method, we found significant improvements over these baselines and the best approach incorporates both information retrieval and machine learning techniques. In sum, we have shown that search engine queries can be successfully mapped to concepts from the Linked Open Data Cloud.

RQ 3a. What is the best way of handling a query? That is, what is the performance when we map individual n-grams in a query instead of the query as a whole?

The best way of handling query terms is to model them not as separate n-grams, but as a single unit—a finding also interesting from an efficiency viewpoint, since the number of n-grams is quadratic in the length of the query.

RQ 3b. As input to the machine learning algorithms we extract and compute a wide variety of features, pertaining to the query terms, concepts, and search history. Which type of feature helps most? Which individual feature is most informative?

As became clear from Table 6.16 and 6.18, DBpedia related features such as inlinks and outlinks and redirects were helpful. We also found that features pertaining to both the concept and query (such as the term frequency of the query in various textual representations of the concepts) were essential in obtaining good classification performance. Such information may not exist in other ontologies.
RQ 3c. Machine learning generally comes with a number of parameter settings. We ask: what are the effects of varying these parameters? What are the effects of varying the size of the training set, the fraction of positive examples, as well as any algorithm-specific parameters? Furthermore, we provide the machine learning step with a small set of candidate concepts. What are the effects of varying the size of this set?

With respect to the machine learning algorithms, we find that reducing the quantity of training material caused only a marginal decline in performance. This means, in practical terms, that the amount of labor-intensive human annotations can be greatly reduced. Furthermore, our results indicate that the performance is relatively insensitive to the setting of various machine learning model parameters; optimizing these will improve the absolute scores but not change the ranking of machine learning models (when ranked by their performance). As to the size of the initial concept ranking that is given as input to the machine learning model, we find that the optimal number is three; the performance declines above this value.

The concepts suggested by our method may be used to provide contextual information, related concepts, navigational suggestions, or an entry point into the Linked Open Data cloud. We have shown that the optimal way of obtaining such conceptual mappings between queries and concepts involves both concept ranking and filtering. This approach outperforms other ones, including lexical matching and using retrieval alone. However, the queries we have used in this chapter are specific to the given system and domain. Although the concepts we link to are taken from the general domain, the used queries raise questions about the generalizability of the results when queries are taken from other, broader domains. In the next chapter we address this issue, by applying the same approach to query sets taken from the TREC evaluation campaign, including a set of queries taken from a commercial web search engine's query log. There, we use them for query modeling, by sampling terms from the Wikipedia articles associated with the mapped concepts using the same method as the one presented in Chapter 5. Furthermore, we also compare the performance with an approach using solely relevance feedback methods, as detailed in Chapter 4.