Chapter 3
Mashups for web search engines

Ioannis Papadakis and Ioannis Apostolatos

Abstract Current practice in looking for information on the web states that searchers rely on large-scale web search engines to get assistance. The quality of the search results is analogous to the ability of the searchers to accurately express their information needs as keywords in the search engine’s input box. In this chapter, an attempt is made to explore the various efforts that have been made regarding the query construction/refinement phase of a search session on the web. Along these lines, a number of cases are presented that are based on intuitively-created mashups for the underlying web search engine. Particular attention is given to two query construction/refinement mashups that integrate various DBpedia datasets with the web search engine provided by Google.

©Springer-Verlag Berlin Heidelberg 2013

3.1 Introduction

The crucial role of search engines in locating meaningful information on the web is justified from the fact that currently, three out of the top-5 sites on the web (according to Alexa) are search engines. However, despite their popularity, such services do not always succeed in retrieving the right results for their users. Sometimes, queries suffer from intrinsic features of natural languages such as ambiguity and synonymy of words. Things get even worse when a top-ranked resource squeezes out from the search results list other, less popular resources with the same name.

Ioannis Papadakis
Ionian University, Department of Archives and Library Science, Ioannou Theotoki 72, Corfu, 49100, Greece e-mail: papadakis@ionio.gr

Ioannis Apostolatos
University of Piraeus, Department of Informatics, Karaoli and Dimitriou 80, Piraeus, 18534, Greece e-mail: japost@unipi.gr

Thus, it appears that search engines should not only pay attention to efficiently ranking their results, but also to provide the necessary tools that will enable their users issuing the right queries.

In this chapter we explore mashups for the query construction/refinement phase of web searches. Such mashups often refer to the collaborative knowledge accumulated in Wikipedia. As will be shown in this work, query construction/refinement mashups for web search engines have been reported prior to the advent of linked data.

This chapter presents in detail two query construction/refinement mashups that are based on DBpedia [29] and targeted towards the users of major search engines on the web. More specifically, the first approach takes advantage of a centralized mashup that is based on various DBpedia datasets that are integrated into a relational database schema. Users may interact with the provided GUI to issue queries to the underlying database that will, in turn, respond with the corresponding information. Such information is represented as recommendations for constructing and/or refining an initial query. The second approach takes advantage of a contextual mashup that is dynamically created each time a search session is initiated. Users are able to select words from the mashup and formulate a query that is ultimately redirected to Google. Both mashups explore ways to integrate large-scale web search engines with linked data to produce semantically rich queries.

The rest of this chapter is structured as follows: the next section describes the different information seeking modes a searcher may exhibit during a search session. Section 3 discusses the problems that web searchers encounter when they try to express their information needs to a search engine. Section 4 describes the various approaches that have been followed by search engines over time to aid their users in providing meaningful queries. Section 5 outlines some mashups that are targeted towards the query construction and refinement phase of a search session. The following section is devoted to two query construction/refinement mashups that are based on linked data and especially DBpedia. Both of them are discussed in detail. Finally, in the last section, among others is concluded that DBpedia-based mashups are excellent candidates for integration with web search engines that provide programmable access to their search box.

### 3.2 Web querying behaviors

According to the sense-making model for information seeking [1], searchers find themselves in various information seeking situations when looking for information. Such situations reflect their information needs, defined as the perceived needs for information that lead to using an information retrieval system [2]. Thus, it is the information retrieval system’s responsibility to bridge the knowledge gap between the searcher’s needs and the actual resources that satisfy such needs.

Web search engines have to overcome many problems to succeed in providing searchers with the right information. Thus, a major issue they have to deal with is
the fact that average searchers do not experience a standard behavior during their
information seeking process. According to the corresponding literature ([3], [38]),
there are various modes a searcher may adopt when participating in such a pro-
cess. Most recently, Spencer [4] identified four intersecting modes for information
seeking.

- The ‘known item’ information seeking mode, in which searchers know exactly
what they want and also what terms to use to verbalize it. This particular mode
is straightforward and easy to comprehend. In this mode, web search engines
experience various difficulties imposed mainly by the very nature of natural
languages. As it will be discussed later in this chapter, the way a search query
is formulated provides insights not only about the searchers’ understanding of
the problem, but also of the information necessary to address it [6].

- The ‘exploratory’ mode for information seeking states that searchers have a
general idea of what they need to know but they are not sure how to express it as
a set of terms for the search query. Usually, they recognize the right information
when they come across it, but they also may not know whether they have found
enough information. Search engines have to deal with the fact that searchers are
not able to properly formulate queries, since they do not know how to phrase
such queries [9]. Thus, the searchers’ inability to express what they are after
within the environment of the search engine, forces them to perform an initial
search and exhaustively run through the search results to learn about the domain
they are after [9] and eventually get some ideas for relevant terms and their
synonyms. This is a rather tedious process that involves running through a lot of
useless information to find comparatively smaller pieces of useful information.

- The “don’t know what I need to know” information seeking mode refers to
searchers that may think they need one thing but they actually need another.
Or, they may be looking at an information resource without a specific goal in
mind. This mode consists of searches occurring in a complex and/or unknown
domain (e.g. legal, medical, financial) as well as searches addressing the need
of keeping up-to-date. As stated in [10], searchers are asked to confront the
paradox of describing something they do not know without any help from the
search engine. Especially when it comes down to major web search engines,
searchers are offered regrettably few options to reformulate their initial query.

- Finally, the ‘re-finding’ mode addresses information seeking situations where
searchers are looking for things they have already seen. From the search en-
geine’s point of view, this mode is addressed by personalized search, where users
have to sign-in to the personalized search engine [5]. There is also the option
of bookmarking the search query, but since bookmarking can be achieved away
from the search engine, it will not concern us further in this work.
3.3 Issuing effective queries to search engines

In their quest for satisfying their information needs through the employment of a search engine, users participate in search sessions. A search session initiates when a user starts expressing his information needs according to the specific dialect of the search engine (commonly, by typing some keywords to the appropriate search box) and concludes when a user stops interacting with the search engine. From a certain point of view, a search session is comprised of two phases, namely the query construction and query refinement phases. The query construction phase refers to the attempt of a user to articulate the initial query that will bring the first results from the search engine whereas the query refinement phase refers to any subsequent attempts to ‘fine tune’ the query that has been initially issued to the underlying search engine, to improve the quality of the corresponding search results.

The query construction phase of a search session is crucial to the fulfillment of the information needs of the searcher. During the query construction phase, a searcher has to adopt his information needs to the specific dialect of the underlying search engine. However, human languages have certain features that should also be taken under consideration during the query construction phase. Thus, queries expressed as consecutive terms suffer from the possible polysemy of the corresponding words. Polysemy occurs when a word has more than one sense [7]. Consequently, a query consisting of an ambiguous word without further information that correctly disambiguates such word, may result in a search results list with completely useless information. Current practice in major web search engines’ ranking tactics states that the risk of cluttering the search results list with useless resources is greater when searchers are seeking for the less popular sense of a polysemous word.

Provided that a query is correctly disambiguated, it could be further refined to provide search results of better quality to its users. The query refinement phase of a search session should take under consideration another important feature of natural languages: synonymy of words. Synonymy occurs when two or more words share the same meaning [7]. Thus, two web pages using two different words to express the same concept, are indexed differently by the search engine. The magnitude of synonymy’s influence to search engines can be further realized by taking under consideration the fact that the probability of two persons using the same term in describing the same thing is less than 20% [8].

Evidently, as users become more dependent on the web to find information about a subject of interest, there is an ever-increasing need to equip search engines with modules that can assist information seekers select queries that express their varying search intentions in a distinguishable by the engine manner.

3.4 Aiding searchers during their search session

To address the difficulties that web users encounter while searching for information about a topic of interest, a number of techniques have emerged such as cluster-
ing, human-powered search engines, personalization, relevance feedback, etc. Such techniques will be discussed in the following sections.

### 3.4.1 Clustering

Perhaps the most commonly known clustering search engine on the web is Yippy\(^2\). Yippy is a prevailing meta-search engine that adds a sidebar containing clusters (labeled as ‘clouds’) next to the search results list. Each cluster corresponds to a topic and contains one or more items appearing to the search results list. Thus, searches are able to filter out results belonging to a specific topic. Such clusters derive from the short descriptions that accompany each item returned from the underlying search engines. Due to the automated nature of the service, Yippy finds it difficult to provide semantically distinguished labels corresponding to the actual content of the clusters. For example, consider the query ‘semantics’, which yields semantically overlapping labels for the clusters ‘Programming, Language’ and ‘XML, Language’ respectively.

### 3.4.2 Human-powered search engines

The phrase ‘human-powered’ refers to a search engine which has its results list affected by human intervention, usually by people rating individual results further up or further down [12]. The rationale behind human-powered search engines is, as a bit exageratively mentioned in [13], the fact that machines are excellent in executing code very fast but they do not share, they cannot judge, they have no appreciation. So, a search engine that enables its users to determine the position of an information resource in a search results list (capturing this way ‘the wisdom of the crowd’ [13]) will always be better than the most algorithmically efficient web search engine.

In this line of thoughts, a number of human-powered search engines have emerged. Some of them rely exclusively on their users to build search results lists (e.g. stumpedia\(^3\)), but most of them do not try to ‘reinvent the wheel’, in the sense that they behave as hybrid search engines, applying human intervention to re-rank results that are initially machine-generated. For example, Anoox\(^4\), and iRazoo\(^5\) depend on a voting system on affecting the position of an information resource within a search results list. According to the founders of such systems, people’s opinion always outweighs machine’s algorithms.

---

\(^2\)http://yippy.com  
\(^3\)http://www.stumpedia.com  
\(^4\)http://www.anoox.com  
\(^5\)http://www.irazoo.com
Another kind of human-powered search engines (also called social search engines) tries to improve its quality by applying social networking logic to the underlying workflow. For example, MySidekick\(^6\) is a human-powered search engine that allows people to find and submit information resources that are then automatically tagged with terms used during the search session. These tagged pages are then anonymously shared within the mySidekick community to provide better results for all. Another effort, based on the concept of Wikipedia, wikia\(^7\) (founded by Jimmy Wales, the founder of Wikipedia) allows users to edit, annotate, comment, delete and expand the search results.

As a common ground, it is argued that the above solutions are mainly targeted to the problem of providing better quality of search results by improving the ranking and/or the topical context of the retrieved information resources. Moreover, human-powered search engines suffer from the fact that they are still just as easy to 'game' as more traditional engines [12] and from the fact that they have to persuade their users to provide feedback (implicit or explicit) to succeed.

### 3.4.3 Relevance feedback

According to relevance feedback, queries are reformulated based on previously retrieved relevant and not relevant information resources [14]. This technique provides a controlled query alteration process designed to emphasize some terms and to deemphasize others, as required in particular search environments. However, such a technique cannot be easily applied to large-scale web search engines, where authentication is difficult to impose and diversity prevails. Moreover, explicitly asking searchers to contribute to the overall information seeking process simply won’t work. The average web user employing a search engine has no motivation in spending time this way [12]. On the contrary, financially motivated stakeholders could take advantage of the impact of relevance feedback to the search engine to promote business web sites higher in the search results lists. Even in the case of automatic (blind) relevance feedback, where terms from the top few information resources returned are automatically fed back into the query [15], success is by no means self evident. This particular technique may possibly be effective in the 'known-item' information seeking mode (see sect. 9.2), where web search engines in any case seem to perform well.

\(^6\)http://www.mysidekick.com
\(^7\)http://search.wikia.com
3.4.4 Ontology-driven approaches

Ontologies can either derive from the document collection at hand or from other sources [17]. However, integrating ontologies with web search engines has proved to be less successful in the past [18]. This is due to the fact that a) domain ontologies are expensive to produce b) they are only available for a small proportion of document sets, and c) appeal only to expert users. Even in the case where generic ontologies are employed, shortcomings in any specific technical domain have been identified [10]. However, attempts in large scale linguistic-oriented ontologies on the web like EDR [19], CYC [20] and WordNet [7], together with the evolution of the web as a collaborative environment where contributions of any kind are greatly endorsed by the overall web community, may turn things around, as far as ontology-powered query formulation is concerned.

As it will be shown later in this chapter, the advent of linked data poses another argument in favor employing ontologies and/or other external sources of information to the query construction/refinement phase of a search session.

3.4.5 Search personalization

Search personalization is the process of incorporating information about the user needs in the query processing phase. One approach to personalization is to have users describe their general search interests, which are stored as personal profiles [24]. Recent search personalization approaches on the web involve integration with some kind of external semantic structure to identify the context of each search session [23, 21, 22]. More specifically, the authors of [21] describe a profile representation using Internet domain features extracted from URLs. In [23], an effort is made to model the user context as an ontological profile by assigning implicitly derived interest scores to existing concepts deriving from the Open Directory Project (ODP) ontology. Another search personalization technique based on the ODP ontology is defined in [22]. More specifically, a user profile is built by accumulating graph-based query profiles in the same search session. In contrast to [21], the user profile is represented as a graph of the most relevant concepts of an ontology in a specific search session and not as an instance of the entire ontology.

3.4.6 Google’s approach

During the past few years, major web search engines and especially Google that seems to be the most popular one, have evolved their provided functionality. Al-

---

8 http://www.dmoz.org
though the mechanics of their approaches are not officially published, it is evident
that some of the above techniques (i.e. search personalization, relevance feedback,
human intervention in ranking) are finding their way into the provided searching
process.

More specifically, Google’s “+1”\(^{10}\) (successor of discontinued ‘Stars’ and ‘Search-
Wiki’) approach takes advantage of Google Account authentication services to iden-
tify the searchers and consequently log their personal search tactics. Such informa-
tion is also available to each user as his search history. Moreover, quite recently,
Google introduced ‘knowledge graph’ in an attempt to provide disambiguation func-
tionality and crowdsourcing recommendations to it’s users through the employment
of an underlying graph that is based on collaborative knowledge.

As far as the query construction phase of a search session is concerned, ma-
jor web search engines have made considerable progress. Autosuggest functionality
within the search box is currently provided by default; according to Google’s ap-
proach, upon issuing a query, a list of query suggestions is displayed. Consequently,
users can rapidly express their initial query by selecting and promoting the sugges-
tion that best suits their information needs. Moreover, Google provides the option
to fetch search results while users type their query (i.e. Google instant\(^{11}\)).

3.5 Integrating search engines with the semantic web:
mashup-based approaches

In the context of web search, mashups could be described as the efforts that are
targeted towards the provision of extended functionality to the traditional paradigm
of a search engine through the combination of data deriving from multiple sources.
Such functionality is often realized as innovative query construction/refinement ser-
vices aiming at enhancing users’ search experience.

The idea of enhancing the performance of query construction/refinement meth-
ods during search by creating mashups using external resources is not new. As a
matter of fact, many works suggest the utilization of large corpora [25] or ontologies
[11] for making rich integrations between user queries and document collections in
the hope of improving the quality of search results. Recently, researchers suggested
that Wikipedia may serve as the external resource to support query construction and
refinement. In this context, Wikipedia-based querying methods have been reported
in [10], [26] and [27]. Here, we briefly discuss the most important mashups and
underline their methodology and effectiveness.

Thus, in [27] a query refinement method based on Wikipedia articles is intro-
duced, which aims at improving retrieval effectiveness of poorly articulated queries
for which automatic relevance feedback is not helpful. The authors in [27] have
created their own machine-readable version of Wikipedia, which they employ to

\(^{10}\) http://www.google.com/psearch

\(^{11}\) http://www.google.com/instant
expands the initial queries. The corresponding evaluation against the TREC Robust Track [28] revealed that the Wikipedia corpus provides significant expansions to the initial query, as compared to automatic relevance feedback, provided of course that the topic of the initial query exists within the Wikipedia corpus. Finally, it should be noted that the work in [27] is intended for use in controlled repositories, not for integration with web search engines where important issues arise [16].

In the work of [10], Koru is introduced as a mashup consisting of a digital library and a query construction/refinement interface that is based on Wikipedia (figure 3.1).

According to Koru’s query topics panel, a typical search box receives the searchers query. The provided functionality dismantles the query to a list of related topics, as they have been identified in the underlying thesaurus. Such list also contains possible disambiguations of the containing topics according to the thesaurus. The list is ranked according to the likelihood that each contained topic is a relevant, significant topic for the current query. Such relevance is measured in terms of statistical and semantic significance of the topics within the underlying document collection. Searchers are able to fine-tune their query by manually excluding and/or including topics from their query in an intuitive, user-friendly manner. Each topic is also accompanied with a paragraph containing possible synonyms. Such information does not explicitly participate in the query. The purpose is to help searchers understand the topic. Moreover, searchers are able to progressively expand their initial knowledge concerning their information needs by interacting with the interface. Such interaction is realized through the employment of non-moveable pop-up windows containing topics that are related to the topic they selected from the previous list. In the case they find significant information during their interactions, they can incorporate the corresponding terms within the query by selecting them. Similar to [27], the authors of [10] have created their own machine-readable version of Wikipedia. Koru is a well-suited approach for controlled collections such as digital libraries, but practically unsuitable for web search engines; the enormous size of the information re-
sources within web search engines together with their highly dynamic and complex nature demand for completely decoupling the query interface from the underlying information resources to avoid scalability and performance problems [16]. This is not the case in Koru, where query terms are ranked according to their relevance to the document collection. Additionally, Koru’s GUI introduces new metaphors and thus entails a high learning curve to its users. According to the inductive user interfaces’ technology proposed by "Microsoft Inductive User Interface Guidelines"[12], learning curves on the web must necessarily be quick. Koru’s provided functionality incorporates gauges, multiple sliding panels, pop-ups that dominate the screen and require a certain degree of familiarity from its users, which is by no means self-evident.

Quite recently, Tweetnews[13] emerged as a mashup that integrates search engines with the popular social media service Twitter. Tweetnews was created in 2009 by Vik Singh, the architect of Yahoo! Search BOSS API[14]. Essentially, Tweetnews uses this API to bring Yahoo’s news search results[15] and reorders them based on how similar they are to the emerging topics found on Twitter for the same query. The results that the API returns, many times are irrelevant with the user’s original query. This happens because some news come higher in rank since they are ordered only by timestamp.

3.6 Integrating web search engines with the semantic web: DBpedia-based mashups

The advent of Linked Open Data – LOD and especially DBpedia facilitated the creation of many mashups in various disciplines, especially at the field of web engineering [30]. LOD is essentially the building block of the so-called ‘Web of Data’ in which anyone can publish information, link to existing information, follow links to related information, and consume and aggregate information without necessarily having to fully understand its schema nor to learn how to use a proprietary Web API [37]. The flagship of LOD is DBpedia, which contains structured content from the information created as part of the Wikipedia project. In August 2012, DBpedia contained descriptions of more than 3.5 million things, many of which are also available in 18 languages[16].

However, when it comes down to web search, LOD-mashups are not so easy to find. In this section, two mashups are presented that are both targeted towards the

13 [http://tweetnews.appspot.com](http://tweetnews.appspot.com)
exploitation of crowdsourced information provided by DBpedia, in favor of web search engines. As it will be shown later, the two approaches converge in their ultimate goal and diverge in their technical solutions.

The first mashup [31, 32] replicates specific DBpedia datasets in a traditional relational database schema and integrates such data with the search box of two major web search engines, namely Google and Yahoo!, in favor of a query construction/refinement service for their users. The service is based on a novel GUI that manages to hide the inherent complexity of semantic web terminology from its end-users.

GContext [33], the second mashup, facilitates a DBpedia-based, query construction/refinement service for Google. GContext follows a decentralized approach. It creates appropriate SPARQL queries to DBpedia’s endpoint and thus recommends its users how to construct/refine their initial query.

The following section outlines the advantages of integrating linked data with web search engines, particularly during the query construction and refinement phase of a search session.

### 3.6.1 Integrating LOD with web search engines

The aforementioned mashups underpin query construction/refinement services on top of web search engines in order to help information seekers formulating queries that express their search intentions well. This way, search engines can propose queries which adapt to the user needs. Such services are particularly helpful when users are unable to formulate accurate and specific queries (e.g. "don’t know what I need to know" mode of information seeking).

One thing that makes search engines so popular is that they enable users query the web in an intuitive yet simple manner, i.e. by submitting a few keywords to the engine’s search box. Despite the intended simplicity associated with querying the web via a web search engine, web searchers may spend too much time reformulating queries, without being able to satisfy their information needs. Search engines provide little help to users with vague knowledge of the terminology employed within relevant documents. Even if searchers succeed in locating the information sought, they often realize that their successful queries differ significantly from their initial query.

The motive behind the aforementioned mashups is to bridge the semantic gap between the initial query and the ideal query that the search engine expects from a user provided that he knows in which terms pertinent documents answer his question.

Both of the query construction/refinement services extend the functionality of the traditional search box and act as an intermediate layer between searchers and web search engines.

[http://dbpedia.org/sparql](http://dbpedia.org/sparql)
The following section presents the first mashup, which follows a centralized approach in providing DBpedia-based information to the users of a search engine during their search sessions.

### 3.6.2 A DBpedia-based, centralized mashup for web search engines

#### 3.6.2.1 Description

In this section, a mashup is presented, which assists searchers to formalize their query and pick the most suitable terms. The mashup exploits crowdsourced Wikipedia knowledge that is available via various DBpedia’s datasets. Nevertheless, the described approach can be easily extended to incorporate other LOD such as YAGO [34]. Serving as a LOD-based mashup, the described approach provides an interactive GUI that seamlessly integrates the knowledge provided by web users with web search engines. Such knowledge is modeled in a carefully designed, modular conceptual schema that supports querying against a large volume of linked data.

The underlying schema has been developed by making use of N3-formated datasets provided by DBpedia. The details of the schema construction are given in sect. 3.6.2.2.

Searchers are able to express their information needs by interacting with an accordingly designed, LOD-browsing GUI. More specifically, interactions through the GUI are converted to query/response messages that are administered by a middleware (figure 3.2).

![Fig. 3.2: Architecture of DBpedia-based, centralized mashup](image)

Queries are encapsulated in http-GET requests and responses are expressed as xml-based strings. The middleware issues the queries to the underlying datastore, which, in turn, delivers the appropriate responses to the middleware. Such responses are converted to xml and channeled back to the GUI.

---

Finally, the GUI transforms responses to keyword-based queries and issues them to the underlying web search engine. A detailed description of the approach is provided at the following sections.

### 3.6.2.2 A Wikipedia-based schema

As previously mentioned, the described system stores linked data originating from DBpedia into a datastore that is based on a conceptual schema. Several studies exist that rely upon DBpedia datasets for building highly expressive ontologies via the combination of Wikipedia and WordNet\(^\text{19}\). Two of the most widely known resources that have emerged from such efforts are the Kylin Ontology Generator (KOG) [35] and the YAGO ontology [34]. The above studies motivated researchers to build query construction/refinement services that mediate between searchers and web search engines with the help of knowledge drawn from various DBpedia datasets.

The service utilizes the following DBpedia datasets\(^\text{20}\): (i) the Wikipedia articles, (ii) the list of disambiguations that Wikipedia encodes for connecting generic articles to their specific interpretations, (iii) the categories under which the Wikipedia articles are classified, (iv) the WordNet classes to which Wikipedia articles correspond, and (v) the articles’ infoboxes that contain semantically rich properties about the considered articles. Table 9.1 summarizes the statistics of the DBpedia datasets that are employed in this mashup.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>No. of items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikipedia articles</td>
<td>2,866,994</td>
</tr>
<tr>
<td>Disambiguation entries</td>
<td>226,978</td>
</tr>
<tr>
<td>Categories entries</td>
<td>339,112</td>
</tr>
<tr>
<td>WordNet classes</td>
<td>124</td>
</tr>
<tr>
<td>Articles linked to WordNet classes</td>
<td>497,797</td>
</tr>
<tr>
<td>Infobox records</td>
<td>19,230,789</td>
</tr>
</tbody>
</table>

These datasets are organized in a conceptual schema as figure 3.3 illustrates. The classes of the schema are: (i) *Articles* that correspond to the Wikipedia articles organized as class instances, (ii) *Categories* that correspond to the appropriate categories of the Wikipedia articles, (iii) *WordNets* that store the type of every Wikipedia article.

Moreover, the relations of the schema are: (i) “clarified by”, a reflexive relation corresponding to the disambiguation pages of Wikipedia articles, which has the

\(^{19}\) [http://wordnet.princeton.edu](http://wordnet.princeton.edu)

\(^{20}\) DBpedia’s dataset dumps are readily available for downloading from: [http://wiki.dbpedia.org/Downloads](http://wiki.dbpedia.org/Downloads)
class 'Articles' as both domain and range, (ii) "in category" that connects every article to one or more appropriate categories. Another relation is (iii) "of WordNet" that connects articles associated with WordNet classes to an appropriate entity type. Finally, the Wikipedia infobox properties are key-value pairs that are expressed as datatype properties of their corresponding article instances.

Based on the above schema, a datastore is created that is incorporated into the query construction/refinement service in the hope of assisting searchers decipher the semantic orientations of their candidate queries before these are actually issued to the search engine. The datastore is serialized as a MySQL database, taking this way advantage of its fast indexing capabilities.

Given the highly dynamic nature of crowdsourced knowledge on the web, the mashup is designed in a way that it can be easily extended with existing and yet-to-appear datasets. Specifically, in case of an incoming dataset, a new class would be added to the schema (see figure 3.3) together with its corresponding relation. Moreover, a new query-response pair would be defined together with the corresponding rendering of the response from the client-side GUI, as will be explained later in this chapter.

Next, the GUI and the middleware of the query construction/refinement service are accordingly illustrated through several examples.

### 3.6.2.3 GUI and Middleware for structuring/refining queries

In this section, the discussion turns to the GUI. It extends the traditional input box of a web search engine by a) suggesting context-aware query formulations based on the first letters in the input box, and b) visualizing the semantics of the initial query (figures 3.5a to 3.3). The design principles require that the GUI should be interactive, inductive, easy to use and fast to execute. Having such requirements in mind, the auto-suggest input box illustrated in figure 3.4 is presented.

As in web search engines like Google and Yahoo!, the search box accepts the input of the user and responds with a set set of alternative query wordings. In the event of a few characters typed in, the box suggests a number of strings that can
be attached to them. The auto-complete suggestions are leveraged from the titles of Wikipedia articles. The searcher can employ any of the suggested query alternatives, or ignore them and search with his own keywords.

Up to this point, the described functionality replaces the autosuggest functionality that has been recently added to major web search engines such as Google and Yahoo! with autosuggest functionality powered by Wikipedia.

If the searcher decides to select one of the offered suggestions, an http-GET query is issued to the middleware, which, in turn, issues appropriate queries to the underlying datastore. Since the datastore is serialized in MySQL, such queries are transformed to sql-select statements. The results of each statement are encoded by the middleware in xml-based strings that are routed back to the GUI. The GUI visualizes the responses as interconnected boxes located above the search engine’s input box. Each box has a title corresponding to an article from Wikipedia and a number of labels beneath it pertaining to the article’s possible semantic relations to other elements. Searchers are able to interact with the boxes by clicking on a relation. In that case, the initial query is refined. As stated earlier in this chapter, there are currently four different types of relations (i.e. disambiguations, categories, WordNet-classes and infoboxes) implemented, although the modularity of the mashup allows for further expansion with more relations.

3.6.2.3.1 Disambiguations/WordNet classes

If the searcher clicks on a “clarified by” relation, query disambiguation is performed as follows: at first, the searcher is presented with a list of all the corresponding disambiguations that match his selected suggestion (figure 3.5a). Such disambiguations could be grouped by WordNet classes, provided they share common WordNet meaning. In such case, upon selecting the corresponding WordNet label, a second-
level disambiguation list appears. By selecting either one of the first- or second-level disambiguations, a new box containing the disambiguated entity is sketched at the right (figure 3.0f), which is connected to the previous box with a line labeled "clarified by". Simultaneously, a search query that consists of keywords deriving from the two box titles (elimination of duplicates is applied) is directed to the underlying search engine.

![Fig. 3.5a: Provided query disambiguations](image)

![Fig. 3.0f: Selected query disambiguations](image)

Under the hood, requests for disambiguations equal to http-GET requests containing parameters determining a) the id of the request, b) the type of the request (i.e. disambiguation) and c) the name of the Wikipedia article for which the disambiguations are requested. The corresponding response is an xml-based string containing the possible disambiguations as a set of <instance> elements. Below, the request: id = “q0”, type = “disambiguation”, name = “Ferrara” results to the following xml-string:

```xml
<reply type='success' iid='Ferrara' rid='disambiguation' id='q0'>
  <instance id='Ferrara'>
    <label lang='en'>Ferrara</label>
  </instance>
  <instance id='Ferrara_Fire_Apparatus'>
    <label lang='en'>Ferrara Fire Apparatus</label>
  </instance>
</reply>
```
In a similar manner, WordNet requests equal to http-GET requests containing parameters determining a) the id of the request, b) the type of the request (i.e. WordNet category) and c) the name of the Wikipedia article for which the WordNet categories are requested. The corresponding response is an xml-based string containing the possible WordNet categories as \textless instance \textgreater elements. Below, the request: id = “q0”, type = “in_wordnet_category”, name = “Ferrara” results to the following xml-string:

\begin{verbatim}
<reply type='success' id='Ferrara'
    rid='in_wordnet_category' id='q0'>
  <instance id='city'><label lang='en'>city</label></instance>
  <instance id='monument'><label lang='en'>monument</label>
</instance></reply>
\end{verbatim}

### 3.6.2.3.2 Categories

If the searcher clicks on a ”in category” relation, a pop-up menu appears containing the corresponding categories. Upon selecting one, a new box named after the selected category is sketched at the right, which is connected to the previous box with a line labeled ”in category” (figure 3.1).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{selected_category.png}
\caption{Selected category}
\end{figure}

Simultaneously, a query consisting of keywords from the two box titles is directed to the underlying search engine. Duplicates are eliminated.

A request for categories is realized as an http-GET request containing parameters determining a) the id of the request, b) the type of the request (i.e. category) and c) the name of the Wikipedia article for which the categories are requested. The corresponding response is an xml-based string containing the possible categories as \textless instance \textgreater elements. Below, the request: id = “q1”, type = “in_category”, name = “amade camal” results to the following xml-string:

\begin{verbatim}
<reply type='success' id='Amade_Camal'
    rid='in_category' id='q1'>
  <instance id='1956_births'><label lang='en'>1956 births</label></instance>
  <instance id='Living_people'><label lang='en'>Living people</label></instance>
  <instance id='Muslim_activists'><label lang='en'>Muslim activists</label></instance>
</reply>
\end{verbatim}
3.6.2.3.3 Infoboxes

Finally, if the server’s response consists of infobox properties realized as key-value pairs, the keys are displayed as labels. If the searcher clicks on a key, its corresponding value(s) appear(s) (figure 3.2). Upon selecting a value, a new box containing the selection is sketched, which is connected to the previous box with a line named after the infobox property’s key.

Then, a search query featuring the keywords deriving from the two box titles (duplicate wipe out is applied) is send to the underlying search engine. Each sketched box corresponds via its title to a part of the resulting query. The searcher controls the participation of each box in the search query by clicking on the checkbox that resides on top of each box (figure 3.3).
This way, the searcher is provided with information for determining and expressing the semantic orientation of his queries, before/while these are issued for search.

A request for infoboxes equals to an http-GET request containing parameters determining a) the id of the request, b) the type of the request (i.e. infobox) and c) the name of the Wikipedia article for which the infoboxes are requested. The corresponding response is an xml-based string containing the possible key-value(s) pairs of the infobox as `<dtprop>` elements (The key of the infobox is the value of `dtprop`’s attribute ”id” and the value of the infobox is the value of the `dtprop` element). Below, the request: id = “q2”, type = “infobox”, name = “Zathras” results to the following xml-string:

```xml
<reply iid='Zathras' qtype='infobox' id='q2' type='success'>
<dtprop id='affiliation'>Great Machine</dtprop>
<dtprop id='finish'>War Without End</dtprop>
<dtprop id='name'>Zathras</dtprop>
<dtprop id='planet'>Unknown</dtprop>
<dtprop id='portrayer'>Tim Choate</dtprop>
<dtprop id='race'>Unknown</dtprop>
<dtprop id='start'>Babylon Squared</dtprop>
</reply>
```

### 3.6.2.4 Discussion

As final notes, it should be stated that by employing the described mashup, searchers are instantly acquainted with query terms that otherwise would take them a lot of time to gather by exhaustively running through the search results of potentially vague queries. Thus, the corresponding service is particularly useful for the ‘exploratory’ and the ”don’t know what I need to know” information seeking modes that have been presented in sect. 9.2.

Additionally, the provided GUI metaphors are smoothly integrated into the traditional search engine’s GUI, since they occupy just a small portion of the screen on top of the input box, thus leaving plenty of room for the search results.

Furthermore, the simplicity of the underlying architecture not only renders the approach scalable to future enhancements with more semantically-rich datasets, but also guaranties its rapid execution. The above features are very important for web search engines where time and space play a crucial role to their success.

The employment of common web widgets such as the autosuggest box and interactive boxes as well as the absence of semantic web terminology from the GUI, renders the service fast to learn and easy to use.

Finally, it should be mentioned that if there is no available information about the user-typed terms, the overall search process does not break down and the query is transparently forwarded to the underlying search engine. Therefore, the worst case scenario is that searchers do not get any help from the service, but still their query is automatically submitted for search.
The query construction/refinement service has been integrated so far with two major web search engines (Google and Yahoo!) and can be accessed online\(^{21}\). Thus, it is believed that the integration is doable for any search engine that gives programmable access to the input box.

The next section presents GContext, another mashup that provides query construction/refinement functionality to the users of web search engines. Although the origin of the underlying semantic data is common (i.e. DBpedia), this particular mashup follows a decentralized approach in providing the corresponding functionality. Finally, the GUI is largely based on textual metaphors such as lists\(^{22}\).

### 3.6.3 GContext: A DBpedia-based, decentralized mashup for web search engines

GContext is another DBpedia-based mashup intending to provide query construction/refinement functionality to web search engines. More specifically, GContext dynamically suggests disambiguations during the query construction phase of a search session. Moreover, the service provides semantic refinements to the resulting query. The suggestions and/or the refinements originate from Wikipedia and are made available through DBpedia and other Linked Open Data – LOD providers. The GUI of the service aids searchers in selecting the appropriate context words that will ultimately reach Google, the search engine that lies beneath the service. The described approach explores ways to integrate web search engines with linked data to produce semantically rich queries. The service is presented through a number of case studies that correspond to real-world situations in information seeking on the web.

#### 3.6.3.1 Overview

The query construction/refinement service that is implemented through GContext is a two-step process: Initially, it provides autosuggest functionality by reacting to the corresponding keystrokes of a searcher. Prefix search is performed to an index that comprises words and/or phrases originating from Wikipedia and made available through DBpedia. More specifically, the underlying index contains Wikipedia’s titles. Then, upon selection of a suggestion, the information seeker is offered the chance to refine the initial query.

Every interaction (i.e. suggestion selection and/or query refinements) results in the construction of an appropriate query that reaches the underlying search engine, which, in turn, retrieves the corresponding search results. The underlying search


\(^{22}\) For more details about the specific mashup, the reader is prompted to read [32]
engine is Google’s Custom Search\textsuperscript{23}, parameterized in a way that issues queries to the entire web site population.

The query construction/refinement service treats Google as a ‘black box’, which responds with search results to the corresponding queries. Thus, the principles that govern the described approach could be applied to any search engine that allows programmable access to its search box.

### 3.6.3.2 Provided functionality

As stated earlier, users start typing the first letters of the words they believe that best describe their information needs and receive suggestions that match against Wikipedia’s titles. Such functionality facilitates query disambiguations, since Wikipedia’s disambiguations follow a pattern (i.e. `<ambiguous word>` (disambiguation info)) that is promoted by prefix search.

Then, upon selection of a suggestion, SPARQL queries are directed to DBpedia’s endpoint\textsuperscript{24} in order to retrieve the suggestion’s redirections, categories, wordnets, and infoboxes. Such information is parsed at the middleware and delivered to the GUI.

The GUI follows Google’s search metaphor and creates a side bar at the left part of the screen, which contains the redirections (captioned as ‘synonyms’), categories, wordnets and infoboxes (captioned as ‘context words’) of the selected suggestion. Since the mashup acts as a query construction/refinement service, elimination of duplicates is performed at the corresponding lists.

Each word/phrase within the lists of the side bar responds to the event of mouse-clicking. Thus, selecting a word/phrase from the lists triggers a corresponding interaction. The GUI currently provides a number of interactions. Thus, upon selecting a word/phrase, the selection is:

- appended to the search box elimination of duplicates is applied to refine the initial query. Then, the refined query reaches the search engine and results are instantly retrieved. An exception to this behavior is imposed by selecting a synonym. In this case, the selection replaces the content of the search box instead of being appended to it (see figure 3.4).
- rendered as an interactive bubble on top of the search box. The user has the option to delete the refinement by clicking on the corresponding ‘x’ sign next to the bubble, or, truncate the bubble by clicking on an individual word of the bubble (in this case, the word appears as deleted (a ‘strikethrough’ is applied to the word) and is eliminated from the search box). The user has the option to restore the truncated bubble to its original state by clicking on the deleted word. Any interaction with the bubble on top of the search box refines the initial query. Then, the refined query is directed to the underlying search engine, which, in turn, delivers the corresponding search results (see figure 3.5).

\textsuperscript{23} [http://www.google.com/cse/](http://www.google.com/cse/)

\textsuperscript{24} [http://dbpedia.org/sparql](http://dbpedia.org/sparql)
If the searcher is not satisfied with the provided interactions and their corresponding search results, he may employ the search box once again to perform prefix search on Wikipedia’s titles.

3.6.3.3 Implementation details

As stated earlier in this chapter, the autosuggest control acts as an entrypoint to the provided service and suggests queries that derive from Wikipedia’s titles. Such information is locally replicated and accordingly indexed. It could be argued that DBpedia’s own autocomplete search would be more suitable for the needs of the described service. However, DBpedia’s autocomplete search index does not meet the requirements of GContext and any attempt to post-process the corresponding results would inevitably suffer from intolerable time lag.

Moreover, the provided functionality evolves around issuing specific SPARQL SELECT queries to DBpedia’s SPARQL endpoint. The information that exists the provided GUI is dynamically gathered from the corresponding SPARQL responses. So far, the following SPARQL queries have been sketched:


http://dbpedia.org/lookup
The first query retrieves the URIs that correspond to Wikipedia’s redirection pages. Such URIs are semantically considered as synonyms for the needs of the service.

The second and third queries retrieves the subjects and the wordnets respectively of the suggestion that the user has picked from the autosuggest control. Such information is semantically considered as categories for the needs of the service.
Finally, the fourth query retrieves the key-value(s) pairs that constitute the in-fobox of the suggestion that the user has depicked from the autosuggest control. Then the information is filtered out to remove words that have low informative value as query terms. For example, the following filtering:

\[\text{FILTER}(!\text{regex}($\text{str(?p)}$, "http://dbpedia.org/property/filename"))\]

filters out the filenames that possibly occur in an infobox.

### 3.6.3.4 Evaluation

In order to evaluate the effectiveness of the approach, a comparative analysis against Google's query construction tool (i.e. autosuggest search box) is performed. The aim of this analysis is twofold: Firstly, to highlight the occasions where web search engines do not seem to perform well, and, secondly, to demonstrate the capability of the approach to fill this gap.

Google's suggestions most likely derive from some kind of statistical analysis of the queries that have been issued to the search engine over time\(^{26}\). On the other hand, the described approach matches the user's input against Wikipedia's titles.

When an information seeker expresses his information needs to Google's autosuggest service and consequently receives the corresponding suggestions, there are two possibilities that may occur:

- The service provides a suggestion corresponding to a query that returns useful search results.
- The service provides a suggestion corresponding to a query that returns either both useful and useless or just useless search results (within the scope of the first search results page).

The first possibility occurs when the autosuggest service succeeds in aiding the information seeker finding useful information. From empirical studies, it seems that Google's autosuggest service performs well when the information seeker picks suggestions that correspond to:\(^{27}\)

- popular queries in general (e.g. "beatles"),
- popular queries within the geographic region of the client computer that invoked the search engine (e.g. "coupons uk" when the search is performed within the United Kingdom),
- queries that have been issued to the search engine before from the same user (i.e. personalization),
- queries that contain unambiguous words (e.g. "afghanistan")\(^{28}\).

GContext's autosuggest service cannot possibly outperform Google's suggestions when information seekers need to compose queries that are affected by such factors.

The second possibility occurs when the suggested query corresponds to a word (or phrase) with various meanings (or 'senses') and the information seeker is interested in the least popular one. For example, an information need about 'jaguar' the animal (not the car) corresponds to the suggestion 'jaguar' from Google, which, in turn, corresponds to a search results list (at least within the scope of the first page) full of resources about the famous car and just one resource about the animal. Suggestions provided by Google lead to even more useless search results as the number of possible meanings of an ambiguous word rises. The situation gets even worse when a popular resource (e.g.

---

\(^{26}\) The mechanics of a highly competitive service like Google's autosuggest service are not formally described. Consequently, any attempt to interpret them is subjective.


\(^{28}\) Apart from the above factors there may be other, statistical factors that affect the quality of Google's suggestions.
movie) is named after a word that literary means something else. Consider, for example, the term 'ajax', which has more than 20 senses according to Wikipedia. The described service provides a suggestion that correctly expresses the implied information need (i.e. 'Jaguar (animal)') and consequently results in a list full of useful resources about such a need. The service benefits from the fact that disambiguated words/phrases in Wikipedia appear as article titles that provide contextual words within parentheses after the ambiguous word/phrase. Thus, an autosuggest service that performs prefix search over an index of such literals results in a list of semantically disambiguated suggestions.

Moreover, the service provides query refinements originating from various repositories of the lod-cloud. The semantic nature of such refinements contradicts the statistical nature of the refinements that are currently provided by Google (i.e. time-based, location-based, type-based, language-based, etc). Nevertheless, refinements of both services are considered useful and complementary to each other. Thus, they may co-exist in a yet-to-come, integrated environment.

3.7 Conclusions - lessons learned

In this chapter, the web search engines domain is presented in the context of the particular mashup efforts that have emerged the past few years. The chapter is especially focused on mashups integrating major web search engines and collaborative knowledge to provide value-added query construction and refinement services to their users.

Such knowledge is largely provided by Wikipedia and made readily available in linked data flavor from DBpedia, although a number of individual efforts that are based on Wikipedia have been reported prior to the advent of DBpedia. Along these lines, a couple of DBpedia-based mashups have been presented (a centralized and a decentralized one), offering query construction/refinement functionality to major web search engines like Google and Yahoo!. The modularity of the described approaches renders them applicable to any web search engine that provides programmable access it’s search box. A comparison between the two mashups concludes that the centralized one provides quicker response time to its users. On the other hand, the decentralized mashup eliminates the need to synchronize the information that is being employed by the service and it’s origin (i.e. DBpedia).

As final notes, it is believed that major web search engines should pay closer attention to the issues that arise at the query formulation phase of a search session. Without proper assistance, searchers may end up issuing queries that are not aligned with their information needs. As described throughout this chapter, query formulation issues are largely attributed to the hidden semantics of the corresponding queries. The inherent ability of semantic web technologies and LOD in particular to address such issues states that in the near future we will see the major web search engines incorporating LOD as part of their default functionality. Along these lines, the ‘knowledge graph’ announced by Google seems to endorse such thoughts by integrating the collaborative knowledge provided by FreeBase.

---

29 Wikipedia’s 'ajax' disambiguation article: http://en.wikipedia.org/wiki/Ajax
30 http://lod-cloud.net
31 http://www.freebase.com
References