Dwebic: An Intelligent Search Engine based on Default Description Logics

Patrick Lambrix and Nahid Shahmehri and Niclas Wahlöf
Department of Computer and Information Science
Linköping University
S-581 83, Linköping, Sweden
{patla,nahsh}@ida.liu.se

Abstract

In companies a large amount of information is maintained that is accessible via network communication tools. This makes searching for a particular piece of information a difficult task. We identified two major difficulties that are experienced using existing search engines. These difficulties are that many of the existing search engines only take syntactic information into account and therefore do not find all relevant documents, and that many of the retrieved documents do not contain relevant information.

In this paper we present a search engine that tackles both of these problems. The system, Dwebic, is based on a description logic knowledge base that deals with strict taxonomic information as well as with default information. Dwebic has a world-wide web interface, but our approach does not depend on the world-wide web.

1 Introduction

The amount and complexity of information on the Internet is growing at an enormous rate. The bulk of information makes it hard to find a specific piece of information without knowing its exact location. Even in companies a large amount of information is maintained and the same problems occur although in a smaller scale. Two major weaknesses have been identified in existing approaches. A first weakness is the fact that information is not found because the search engines only make use of syntactic search and do not take semantics into account. Another weakness is the fact that users are provided with a large set of solutions to a query, many of which are not relevant to the user. Existing Internet search tools such as AltaVista, Yahoo! and Lycos are only able to solve these problems to a limited extent since they use simple word indexing techniques.

In many cases, simple indexing is not enough. As an example we tried to find information on French wines on the web using AltaVista and got 34 340 hits. Then we looked for Bordeaux wines and got 5000 hits. However, we also found that 2000 of these hits were not found during the search for French wines. This means that the recall of the search engine should be improved. What would be needed to find those 2000 hits during the search for French wines is that the system could make use of the fact that Bordeaux wines are French wines.

Another problem in retrieval of web pages is the large amount of documents that are retrieved for a given query. Many of these documents are not interesting for a user. Usually when a user is looking for information about a topic, the user is most interested in ‘typical’ information about the topic. Therefore, if we could enhance the search engine such that information about typicality is taken into account, we may improve the precision of the search engine.

In this paper we report on a search engine, Dwebic, that tries to deal with the problems of recall and precision. Our goal is to extend the capabilities of search engines based on syntactic similarities with search and retrieval based on semantic content. We claim that description logics extended with defaults are a suitable formalism on which to build such search tools. The description logic can be used to represent both ontological and object-specific knowledge and can therefore be used to diminish the recall problem. The same language can also be used to express queries. It has been shown before [2] that description logics can model the boolean as well as the vector model of information retrieval and thus more complex queries can be asked. The use of defaults allows for representation of typicality and thus tackles the precision problem. Since each document must be given a description in the description logic our method is primarily useful for search at a local server. We observe, however, that much of the description of a given document can be extracted automatically [4]. Although our implemented system uses documents in HTML format, our framework is not dependent on this.

The conceptual model of Dwebic is shown in figure 1. A query is expanded using strict information and default information. (In Dwebic the user may actually choose whether to search for keywords, to use the strict information, or to use the strict and default information.) It is documents that satisfy this expanded query that are returned by the search engine. One possible implementation of this model (and this is the one on which
A default rule is an expression of the form $A(x) \triangleright B(x)$ where $A$ and $B$ are concept names and $x$ is a variable. Default rules can be instantiated with individuals (closed defaults). A maximally extended knowledge base can be constructed from an initial knowledge base by applying a number of closed defaults. These closed defaults are the generating defaults of the maximally extended knowledge base. As there may be different choices for which defaults to apply, we may obtain different maximally extended knowledge bases from an original knowledge base. We use a preference relation between defaults that is more complex than the simple subsumption between the preconditions of the defaults. Intuitively, we prefer the default $A(x) \triangleright B(x)$ over $C(x) \triangleright D(x)$ if there is a sequence of concepts $A_1, ..., A_n$ such that between each adjacent pair of concepts in the sequence we have a subsumption relation or a default rule. This allows for the use of specificity (as is usual in default reasoning) as well as for influence of the \textit{applicable defaults} on the selection of the default to apply. The maximally extended knowledge bases that are constructed using the preference between defaults are called $B$-extensions. Further, using the preference between defaults we define the notion of \textit{support} as a base for a preference relation. A support can be viewed as ‘evidence’ for preferring one $B$-extension over another. Using these notions we can define the \textit{preferential instance test} as a standard instance test on the intersection of the preferred $B$-extensions. This test allows for deciding when it is ‘plausible’ to conclude membership relations.

The remainder of this paper is organized as follows. For the sake of brevity we concentrate on the Dwebic application and therefore only briefly describe the framework on which Dwebic is based (section 2). The Dwebic system is described in section 3.

## 2 Default Description Logic

The default description logic on which Dwebic is based is described in [3]. To be able to include default information \textit{default rules} can be used in our knowledge bases. A default rule is an expression of the form $A(x) \triangleright B(x)$ where $A$ and $B$ are concept names and $x$ is a variable. Default rules can be instantiated with individuals (closed defaults). A maximally extended knowledge base can be constructed from an initial knowledge base by applying a number of closed defaults. These closed defaults are the generating defaults of the maximally extended knowledge base. As there may be different choices for which defaults to apply, we may obtain different maximally extended knowledge bases from an original knowledge base. We use a preference relation between defaults that is more complex than the simple subsumption between the preconditions of the defaults. Intuitively, we prefer the default $A(x) \triangleright B(x)$ over $C(x) \triangleright D(x)$ if there is a sequence of concepts $A_1, ..., C_n$ such that between each adjacent pair of concepts in the sequence we have a subsumption relation or a default rule. This allows for the use of specificity (as is usual in default reasoning) as well as for influence of the \textit{applicable defaults} on the selection of the default to apply. The maximally extended knowledge bases that are constructed using the preference between defaults are called $B$-extensions. Further, using the preference between defaults we define the notion of \textit{support} as a base for a preference relation. A support can be viewed as ‘evidence’ for preferring one $B$-extension over another. Using these notions we can define the \textit{preferential instance test} as a standard instance test on the intersection of the preferred $B$-extensions. This test allows for deciding when it is ‘plausible’ to conclude membership relations.

### 3 Application: Dwebic

We have implemented this framework and the preferential instance test inference on top of CLASSIC[1] (LISP version 2.0). The implementation is a module containing a set of LISP functions that extend the CLASSIC API. We have chosen not to let the default extension affect other parts of the system such as the classification hierarchy. Therefore the standard operations in CLASSIC have the same behavior as before. The preferential instance test works as explained in the previous section.

In addition to the framework described in the previous section we have implemented two other useful extensions. The first is that we allow the user to define a preference between defaults with the same prerequisite. If the user decides to define this preference the algorithm takes this preference into account when deciding which defaults to use while constructing a maximally extended knowledge base. Another extension is that we have implemented an extra mode, in which the algorithm adds default information to the knowledge base only when the addition does not make the query’s answer set empty.

We have used our default extension of CLASSIC as the basis for our implementation. The system architecture of Dwebic is as described in figure 1. In Dwebic actual HTML documents are represented by individuals which have a role “url” containing the url of the HTML document. The documents belong to different classes. The objects that we are interested in are represented by individuals which store the urls of the HTML documents that contain information about them as fillers for their role “url”. On top of the default description logic knowledge base we have implemented a world-wide web interface. Users interact with the knowledge base by entering their query in a form through the web browser. The system returns a list of selectable links to the relevant documents as answer to the query.

Using Dwebic as a search engine has at least the following advantages:

- Strict information can be used to enhance a query such that documents that contain more specific information are retrieved as well. In this case the recall of the system is improved.
- Defaults can be used to find documents which have been given a limited or underspecified description. In this case default reasoning is performed on the given documents (default reasoning at assert time). This allows for better matching with respect to queries.
- Defaults can be used to restrict the answer set of documents to a query thereby only providing documents concerning answers that are ‘typical’. This allows us to improve the precision while querying. In this case default reasoning is performed at query time.
- The query language for Dwebic has the expressivity of the underlying CLASSIC system. Queries
In the actual system a user can choose whether to use keyword search or search using concept descriptions. The user can also decide whether only strict information should be used or whether default information should be used as well.

4 Conclusion

In this paper a search engine is described that tackles the problem of enhancing the precision and recall for retrieval of documents. The main techniques that we apply are the use of subsumption information and the use of default information. The use of subsumption information allows for the retrieval of documents that include information about the desired topic as well as information about more specific topics. The use of default information allows for retrieving of documents that include typical information about a topic. The strict and default information are represented in an extension of description logics that can deal with defaults. We have tested the system on small-scale databases with promising results. However, more evaluation of the approach for larger databases is necessary. We have also used the approach for a retrieval system, Dwines [3], that given a meal proposes which wine to drink with the meal. Also in this system strict as well as default information is taken into account.

References


