Domain specific information retrieval system

Article · July 2009

CITATIONS
2

READS
1,223

3 authors, including:

Sandi Pohorec
University of Maribor
19 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Milan Zorman
University of Maribor
92 PUBLICATIONS **467** CITATIONS
SEE PROFILE

Domain specific information retrieval system

SANDI POHOREC, MATEJA VERLIČ, MILAN ZORMAN
Laboratory for system design
University of Maribor, Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
SLOVENIA
sandi.pohorec@uni-mb.si, mateja.verlic@uni-mb.si, milan.zorman@uni-mb.si, http://www.lsd.uni-mb.si

Abstract: - In the fast evolving world that we live in today, the search for information has become increasingly important. Of course the tool that we use almost exclusively for this task is an internet search engine. We are all very familiar with the capabilities of large scale search engines, such as Google. Naturally a question arises if there is really a need to develop another search engine although on a smaller, local, domain oriented level. We believe that the answer will, at least in the foreseeable future, be positive. The reasons for this are: access to privileged information, personalization and custom ranking functions. The paper describes a centralized search service for domain specific content. The approach uses automated indexing for various content that can be in the form of a relational database, web service, web portal or page, various document formats and other structured or unstructured data. The indexed data is accessible through a highly optimized and personalized search service.

Key-Words: - information search, personalization, indexes, crawling, distributed data sources, ranking functions.

1 Introduction

We have looked at the problem of developing a search engine through the perspective of one organization. The organization's domain includes all areas of interests of every single member. The indexed content is global by its location, although the only sites indexed are those that are related to the target domain. The main objective is to provide an optimal search system for a particular organization. The examples given in the paper are such that the organization is the faculty where we work (academic environment). The sources that are indexed can be in a variety of forms and formats. The system supports web sites and services, unstructured data (documents), structured data (XML files) and data bases. The system provides personalization of content and automatically uses user credentials to limit access to restricted content.

2 Architecture of modern web search engines

Modern web search engines [1] use a loosely coupled architecture on two levels (crawling, searching). The joining factor between the two is the search index. This allows that the crawling and searching can be relatively independent. Their internal data structures, functions and programming logic can change at any time as long as they still conform to the design of the search index. This independency along with the fact that web crawling is an extremely time consuming task has led to the loosely coupled architecture.

We continue with the examination of the three main components (crawling, indexing, searching) individually.

2.1 Crawling

Web crawler also known as a spider has the task of continuously visiting web pages, downloading their content, transforming from various formats to plain text and finally the update of the search index. Fig. 1 shows the high level architecture of crawling. The most important factor in understanding the crawler is to know its scope. A crawler can be global, such as "GoogleBot", or local (only indexes certain domains, only sites in a certain language or even just a single web site). The strategies that limit the crawler are the following: limitation to link following (the crawler follows only links that link to a certain site or domain or it can follow all links), it can crawl using the technique known as depth-first (only one link on any depth is crawled) or breadth-first (every link is followed before descending a level), it can use narrow crawling (only certain content is crawled) and it can crawl the deep part of internet (content that is hidden and access is restricted to authorized users only). Whether the search index is up to date and to what degree depends on what time interval of sequential visits was selected for the crawler. An important factor that describes a crawler is its behavior during the actual crawling. A crawler should conform to rules provided in the robots.txt file. The file restricts which areas a particular crawler can visit and provides guidelines for general crawling behavior. The speed of crawling is essential so that the crawler does not overload the targeted web site. Certain sites such as Wikipedia provide data dumps of their content in order to avoid extensive crawling. The final important factor of a crawler is the degree of parallelism. This determines whether only one crawler operates on any given time or

are there multiple crawlers that operate simultaneously. If there is more than one then the workload allocation algorithm is also an important factor.

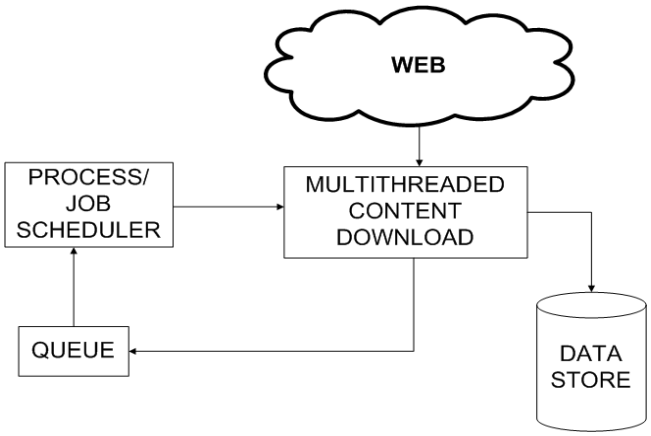

Fig. 1 High level architecture of a crawler

2.2 Search index

The sole purpose of a search index is to diminish the time consumption of searches. If the search engine would not have an index, the search would consist of consecutive reading and searching on every document in the target data. Obviously that would take a considerable amount of time. For a target set of 10.000 documents the search on an index would be completed within milliseconds, sequential reading of every document would be completed in no less than a couple of hours. The crucial factors in index designs are: the way content is added to the index (multiple crawlers are supported or not), the physical storage technique (large scale engine have their own file system such as Google's Big Tables), expected size of the index (whether the content that is crawled is global or not), search time, the ratio between search time and time needed to add content to the index, the maintenance of the index for an extended time period and error handling capabilities. The most common data structures that are used to store the index are: suffix trees, trees, inverse indexes, citation indexes, document-term matrices and many others.

2.3 Search

Search is conducted when a user submits the search string. The search string is a formalization of a user's need for information. It is typical that the search strings are not structured and can have ambiguous interpretations and meanings. Three most common search query types are: information query (thousands of possible results), navigation query (the user is trying to navigate to a known site, such as www.wseas.us) and a transaction query (where the search query is used to conduct a transaction, such as the purchase of an automobile. A crucial factor for understanding search queries is the experience of the user [2]. The most important data on search are: queries (number of search terms, use of logic and modifiers), sessions (types of queries in a session, number of pages visited) and terms (rank/frequency distribution and the most highly used search terms) [3].

3 Overview of the presented system

The system we developed has many of the components basically the same as any of the existing search engines. Unfortunately this cannot be avoided since the development of a customized search engine brings with itself the heavy burden of having to develop and test every component of the search engine not just the ones we wish to upgrade or modify. As we did in section 2 for the general search engine architecture we will examine major components of our system in the next subsections. We will focus only on the major differences between the standard implementation and the one we have developed.

3.1 Specific features

We felt that there is a need to develop a custom search engine intended to provide its users with unique personalization capabilities and result ranking features. The main initiative for this is because the leading search engines cannot provide those capabilities. They are globally oriented and use ranking factors and functions that are intended to work on a global scale. Their crawlers cannot access content beyond the publicly indexable web barrier. The presented search engine is oriented locally although it does index global content. The content indexed is limited only by its inclusion in the search domain, not geographical or language based considerations. The target domain is everything that is connected with the faculty including the courses thought. Therefore for example some parts of the well known W3CSchools site are indexed because web development is taught on multiple courses. The crawler is composed of multiple applications that run concurrently, so it is an example of a parallel crawler. The indexed content is tagged with part-of-speech (POS) [4] tags. Also an ensemble [5] of specialized taggers is used to tag the meaning of words in both the general and domain specific sense. As a main feature the personalization component features a time component and groups of interest areas. In the following subsection we present the individual components of the system in greater detail.

3.2 Data gathering

Any major search engine has only one way to access the documents, web pages or anything else it indexes. The

crawler crawls through the public available content and transfers that data to the indexing server.
In our system there are multiple possible sources of data and therefore we use a number of different approaches to collect it. Fig. 1 shows the data sources that the crawler can index: web content, structured data, unstructured data and data from various applications and databases.
A specialized component is dedicated to the complex task of scheduling the indexing jobs. The indexing intervals are source dependent and occur very often for some sources. The interval is based on the probability of observed content change described in [6]. The metric is calculated with the following equation:

$$Probability\ (content\ changed) = 1 - e^{-RT} \qquad (1)$$

In the equation R stands for rate of change of every data source and T stands for the time span for which the probability is being calculated.
Each content type has a dedicated indexer and the individual indexers run concurrently.
Web content is indexed with a traditional web spider. The spider runs with multiple simultaneous connections. Every connection is opened to a different web site. This is to assure that the indexer does not overload the targeted sites. The content indexed is the HTML content of the sites as well as various attachments. The crawler respects the robots.txt standard for robots exclusion [7].

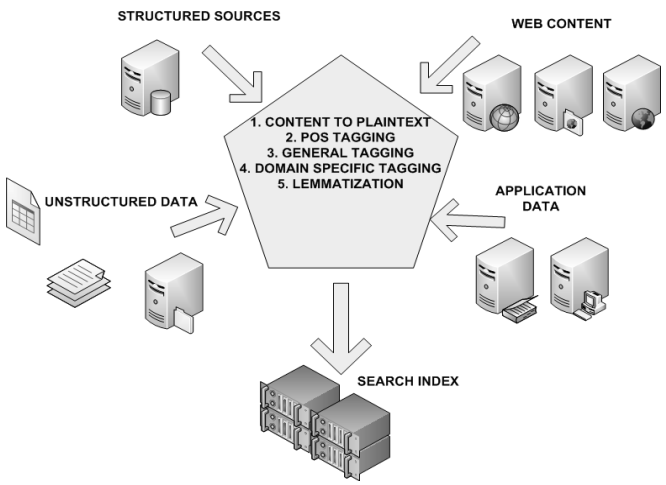

Fig. 2: Architecture of our crawler

The database indexing component operates in two possible modes: (i) fully automated indexing or (ii) indexing according to a provided schema for a particular database. The fully automated indexing relies on the schema of the database. In a relational database the schema describes the data structure and their relations and is an implementation of the conceptual model (ER). We take full use of the fact that the schema gives a detailed description of domain concepts. For the automated indexing the following elements of the schema are important: table relations, attribute data types, attribute constraints (unique) and of course the primary and foreign table keys.
Structured data indexer automatically tags content according to the structure of the document. The database, structured and unstructured data indexers all work in their respective tasks on the local server therefore they share a common component that provides data transfer capabilities. The data transfer is done via web service that is compliant with the MTOM mechanism [8].
All content that is not in plain-text form is transformed to it. The transformation is done with a number of specialized plugins that are compliant with the iFilter interface standard. The iFilter interface is used by the Windows Indexing Service and the newer Windows Desktop Search to index various file formats. We have implemented this component in order to make the task of supporting new file types a simple one. The architecture of the component allows registering of a new iFilter plugin during normal system runtime, therefore any new content type can be indexed without the need to stop normal operations.

3.3 Tagging of content

The tagging is in two phases. In the first the POS tag is assigned to every word. In the second an ensemble of specialized taggers is used for a more detailed tagging focusing on the meaning of particular words rather than POS.
The POS tagging is done with an optimized variation of the Hidden Markov Model Approach (HMM) [9]. The HMM is a statistical parser. Statistical parsers wok by assigning probabilities to possible parses of a sentence locating the most probable parse. We have implemented a variation that uses string matching techniques to determine which of the sentences in the learning corpora is similar to the one being tagged. The technique works in three steps. In step one we tag known words (with the use of dictionary and corpora). Then the sentence is tagged with a pattern. The pattern is a single string (word) in which every character represents the part-of-speech of a single word. The works with unknown POS tags are represented by a question mark. For an example the pattern "NVN?" represents a sentence where the first word is a Noun (N), the second one is a Verb (V), the third again a Noun (N) and the POS of the fourth is unknown (?). By using full-text search operators in step two similar sentences are selected from the learning set (examples of grammatically correct sentences). Then in step three the most probable POS tags are assigned to the unknown words. As an additional factor in the probability calculation the ensemble of specialized taggers is used (the same one used in the second phase of content tagging). An example of this is the use of a name

tagger that specializes in the recognition of person names. If the name tagger tags an unknown word as a person name then obviously the POS of that word is a Noun. The degree of influence of the taggers on the probabilities depends on the reliability of a particular tagger.

The basic idea of how our ensemble should work is the same as the ensemble data mining method of boosting. The method is simply explained with the following example. Suppose you as a patient have certain symptoms. Instead of consulting one doctor, you choose to consult several. Suppose you assign weights to the value or worth of each doctor's diagnosis, based on the accuracies of previous diagnoses they have made. The final diagnosis is then a combination of the weighted diagnoses. This is the essence behind boosting. We have thought at our ensemble in a similar way with the difference that the doctors from the example are each a specialist for a particular area. Their worth is based both on accuracies of previous tagging and the classification of the tagger.

The specialized ensemble of taggers consist of four classes of taggers: (I) tagger that uses a database of examples (II) tagger that generates examples according to some logic (III) tagger that uses regular expressions and (IV) a tagger that uses a combination of taggers from other classes. Some of the taggers implemented include: a name tagger (class I), abbreviation tagger (class I), tagger based on generated values (class II), web and email address tagger (class III) and others.

3.4 Personalization

We will provide an examination of the personalization features by viewing the most important aspects; context, interest areas and content classification. The personalization component provides services that enable it to use data from other application, web services and datastores in various formats. Every data provider is a member of an ensemble. An infering component (»conductor«) has the task of selecting appropriate ensemble members for a particular query. Every component is associated with semantic information that allow the automata to infer where, when and how it should be used. The conductor uses the ensemble members semantic description with the data available for the query to select members thar are relevant to the query. The relevant members are then called upon to retrieve the data at their disposal and submit it to the infering component. The infereing component then provides a ranked list of relevant key words that the search engine uses to personalize the ranking function. Fig. 3 gives an overview of the process.

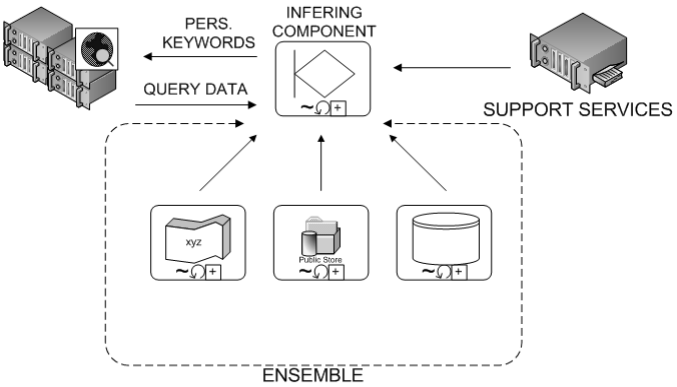

Fig. 3: Personalization process

3.4.1 Context

The personalization is based on the search history and the information gathered from the user profile. Also various other sources are incorporated.

Again we turn to our academic enviroment for usage examples. The information for staff can be automatically gathered from their bibilography and current research projects. For students the curiculum and course timetables can be used to determine the context of a given search. For instance the student »John Doe« logs in a computer terminal at 9:00. At 9:15 he starts a search on the faculty web page.From the IP address and the timestamp the personalization system can easily infer who the user (logon information from the domain server provided by the supporting services) is and what course he is taking currently (course timetable). This automatically determines the search context [10] (by using the key words from the course descriptions) and the search can use the key words from the course description as the context guidelines.

3.4.2 Interest areas

When the current context cannot be determined with a high enough confidence the personalization is based on the users interest areas. The interes areas are a generalization of the current context. For instance if a user is looking for information on loops in the C++ programming language, this would be generalized as the interest area of »programming languages«. This generalization is possible because of the domain specific ontology that was created in order to be able to make this type of generalization procedure. The ontology consists of semantic information and reltionships on every course thaught at the faculty, general information on the faculty it's staff and students. Every course has a few keywords associatiated with every single topic it covers. The topics are linked to various research areas. For instance the hypothetic course »Introduction to algorithms« that lectures on »Statistical taggers« would be linked to the »natural language processing« research area.

3.4.3 Content classification

The first step in content classification is the transformation to plain-text. The classes in which the content is classified are determined by the domain specific ontology. The main properties of documents are obtained with the use of the TFIDF (Term Frequency Inverse Document Frequecny) metric. The metric is defined as follows:

$$d^{(i)} = TF(W_i, d)IDF(W_i) \quad (2).$$

The IDF is defined: $IDF(W_i) = \log\frac{D}{DF(W_i)}$ and D is the number of documents, DF(W) is the number of documents in which the word (W) occurs at least once and TF(W, d) is the number of word W occurancies in the document d.

Documents are assigned to their respecitve classes with the comparion of the class description from the ontology and the main features of the document. For documents that cannot be automatically classified with this method we use the cosine similarity. The cosine similarity is used to find *k* documest that are most similar to the new one. The classes attached to those documents are then used for the new one. Cosine similarity between two documents (d_i and d_j) is defined as follows (3):

$$\cos(d_i, d_j) = \frac{\sum_k d_{ik} d_{jk}}{\sqrt{\sum_l d_{il}^2 \sum_m d_{jm}^2}} \quad (3).$$

4 Conclusion

The presented system provides features that cannot be provided by global search engines. The indexed content is tagged with an optimized version of the well-known HMM model that uses patterns in order to properly tag words on the fly, with almost no data preparation procedure necessary. As a key feature it uses domain specific ontology and a ensemble of specialized taggers to tag the indexed content with the meaning the words have in the target domain. The specialized ensemble of taggers is designed in such a way that they acquire their data autonomously from their respective sources. The is important because it eliminates the need for manual work and oversight.

The domain specific search system we have presented features true personalization and custom ranking to provide the user with effortless information retrieval capabilities. However, as we have come to realize, users expect consistency in their search results. Since the results are personalized they are not consistent over time. The search results are very dependent on the profile of the user. The better the profile the more the results deviate from the standard (the one a user without a profile would see) result set. It is very important to provide the user with the ability to influence the use of personalization (or to choose not to use it at all). This provides the features of the engine to those users that are willing to use them or find them useful.

For future work we will focus on allowing the user to create his own custom domain. The definition of the domain will be a group of search sources (web pages, structured, unstructured data and documents) that the user will select either manually or on the recommendation of the search system. Basically this will result in personal search "agents" that are fully personalized because they are essentially "built" by the users themselves. The users will be able to specify which content to index, provide custom ranking of sites and use search results bookmarks that will be stored for them on the search server, making them accessible anywhere.

References:

[1] Steve Lawrence, Lee C. GILES, Searching the World Wide Web, *Science*, 1998, Vol.280, No. 5360, pp. 98-100.

[2] Ingrid Hsieh-Yee, Effects of search experience and subject knowledge on the search tactics of novice and experienced searchers, *Journal of the American Society for Information Science*, Vol. 44, No. 3, pp. 161-174, 1993.

[3] B. J. Jansen, A. Spink, J. Bateman, T. Saracevic, Real life information retrieval: a study of user queries on the Web, ACM SIGIR Forum, Vol. 32, No. 1, pp. 5-17, 1998.

[4] Christopher D. Manning, Hinrich Schuetze: *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.

[5] Thomas G. Dietterich, Ensemble Methods in Machine Learning, *Proceedings of the First International Workshop on Multiple Classifier Systems*, pp.1-15, June 21-23, 2000.

[6] Brian E. Brewington, George Cybenko, How Dynamic is the Web? , *Proceedings of the Ninth International World Wide Web Conference* (WWW9), 2000.

[7] Martijn Koster, A Standard for Robots Exclusion, http://www.robotstxt.org/wc/robots.html, 1994.

[8] W3C, SOAP Message Transmission Optimization Mechanism, http://www.w3.org/TR/soap12-mtom/, 2005.

[9] E. Charniak, Statistical techniques for natural language parsing. *AI Magazine*, 18(4), 1997.

[10] Steve Lawrence, Context in Web Search, *IEEE Data Engineering Bulletin*, Vol. 23, No. 3, 2000.