Full-Text Search Engines for Databases

László Kovács
University of Miskolc, Hungary

Domonkos Tikk
Budapest University of Technology and Economics, Hungary

INTRODUCTION

Current databases are able to store several Tbytes of free-text documents. The main purpose of a database from the user's viewpoint is the efficient information retrieval. In the case of textual data, information retrieval mostly concerns the selection and the ranking of documents. The selection criteria can contain elements that apply to the content or the grammar of the language. In the traditional database management systems (DBMS), text manipulation is restricted to the usual string manipulation facilities, i.e. the exact matching of substrings. Although the new SQL1999 standard enables the usage of more powerful regular expressions, this traditional approach has some major drawbacks. The traditional string-level operations are very costly for large documents as they work without task-oriented index structures.

The required full-text management operations belong to text mining, an interdisciplinary field of natural language processing and data mining. As the traditional DBMS engine is inefficient for these operations, database management systems are usually extended with a special full-text search (FTS) engine module. We present here the particular solution of Oracle; there for making the full-text querying more efficient, a special engine was developed that performs the preparation of full-text queries and provides a set of language and semantic specific query operators.

BACKGROUND

Traditional DBMS engines are not adequate to meet the users' requirements on the management of free-text data as they handles the whole text field as an atom (Codd, 1985). A special extension to the DBMS engine is needed for the efficient implementation of text manipulating operations. There is a significant demand on the market on the usage of free text and text mining operations, since information is often stored as free text. Typical application areas are, e.g., text analysis in medical systems, analysis of customer feedbacks, and bibliographic databases. In these cases, a simple character-level string matching would retrieve only a fraction of related documents, thus an FST engine is required that can identify the semantic similarities between terms.

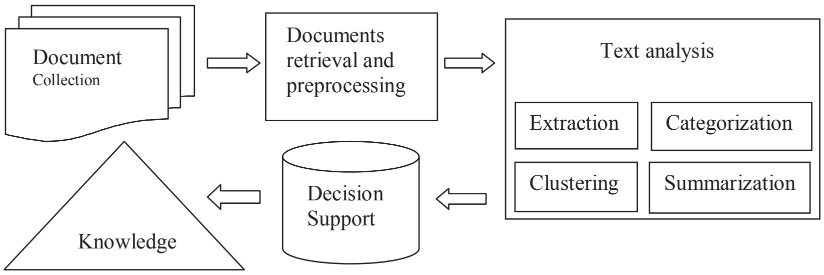
There are several alternatives for implementing an FTS engine. In some DBMS products, such as Oracle, Microsoft SQLServer, Postgres, and mySQL, a built-in FTS engine module is implemented. Some other DBMS vendors extended the DBMS configuration with a DBMS-independent FTS engine. In this segment the main vendors are: SPSS LexiQuest (SPSS, 2007), SAS Text Miner (SAS, 2007), dtSearch (dtSearch, 2007), and Statistica Text Miner (Statsoft, 2007).

The market of FTS engines is very promising since the amount of textual information stored in databases rises steadily. According to the study of Meryll Lynch (Blumberg & Arte, 2003), 85% of business information are text documents – e-mails, business and research reports, memos, presentations, advertisements, news, etc. – and their proportion still increases. In 2006, there were more than 20 billion documents available on the Internet (Chang, 2006). The estimated size of the pool increases to 550 billion documents when the documents of the hidden (or deep) web – which are e.g. dynamically generated ones – are also considered.

TEXT MINING

The subfield of document management that aims at processing, searching, and analyzing text documents is *text mining*. The goal of text mining is to discover the non-trivial or hidden characteristics of individual documents or document collections. Text mining is an

Figure 1. The text mining module



application oriented interdisciplinary field of machine learning which exploits tools and resources from computational linguistics, natural language processing, information retrieval, and data mining.

The general application schema of text mining is depicted in Figure 1 (Fan, Wallace, Rich & Zhang, 2006). For giving a brief summary of text mining, four main areas are presented here: information extraction, text categorization/classification, document clustering, and summarization.

Information Extraction

The goal of information extraction (IE) is to collect the text fragments (facts, places, people, etc.) from documents relevant to the given application. The extracted information can be stored in structured databases. IE is typically applied in such processes where statistics, analyses, summaries, etc. should be retrieved from texts. IE includes the following subtasks:

- named entity recognition – recognition of specified types of entities in free text, see e.g. Borthwick, 1999; Sibanda & Uzuner, 2006,
- co-reference resolution – identification of text fragments referring to the same entity, see e.g. Ponzetto & Strube, 2006,
- identification of roles and their relations – determination of roles defined in event templates, see e.g. Ruppenhofer et al, 2006.

Text Categorization

Text categorization (TC) techniques aim at sorting documents into a given category system (see Sebastiani, 2002 for a good survey). In TC, usually, a classifier model is built based on the content of a set of sample documents, which model is then used to classify unseen documents. Typical application examples of TC include among many others:

- document filtering – such as e.g. spam filtering, or newsfeed (Lewis, 1995);
- patent document routing – determination of experts in the given fields (Larkey, 1999);
- assisted categorization – helping domain experts in manual categorization with valuable suggestions (Tikk et al, 2007),
- automatic metadata generation (Liddy et al, 2002),

Document Clustering

Document clustering (DC) methods group elements of a document collection based on their similarity. Here again, documents are usually clustered based on their content. Depending on the nature of the results, one can have partitioning and hierarchical clustering methods. In the former case, there is no explicit relation among the clusters, while in the latter case a hierarchy of clusters is created. DC is applied for e.g.:

- clustering the results of (internet) search for helping users in locating information (Zamir et al, 1997),
- improving the speed of vector space based information retrieval (Manning et al, 2007),
- providing a navigation tool when browsing a document collection (Käki, 2005).