Mining Generalized Query Patterns from Web Logs

Charles X. Ling*
Dept. of Computer Science, Univ. of Western Ontario, Canada
ling@csd.uwo.ca

Jianfeng Gao
Microsoft Research China
jfgao@microsoft.com

Huajie Zhang*
Dept. of Computer Sciences, Univ. of Western Ontario, Canada
hzhang@csd.uwo.ca

Weining Qian*
Dept. of Computer Science, Fudan University, China
wnqian@fudan.edu.cn

Hongjiang Zhang
Microsoft Research China
hjzhang@microsoft.com

Abstract

User logs of a popular search engine keep track of user activities including user queries, user click-through from the returned list, and user browsing behaviors. Knowledge about user queries discovered from user logs can improve the performance of the search engine. We propose a data-mining approach that produces generalized query patterns or templates from the raw user logs of a popular commercial knowledge-based search engine that is currently in use. Our simulation shows that such templates can improve search engine's speed and precision, and can cover queries not asked previously. The templates are also comprehensible so web editors can easily discover topics in which most users are interested.

1. Introduction

Data mining is a process of discovering implicit and useful knowledge from large datasets [Fayyad, et al, 1996]. Knowledge about datasets can be as simple as statistical information such as means and deviations, but data mining aims at finding higher level or more powerful knowledge, such as classifiers, predictive or descriptive rules, Bayesian networks, cluster descriptions, and so on. Two most important vertical applications of data mining nowadays are in the financial industry and the Internet (World Wide Web).

We study data mining application on the Internet in this paper. In particular, we apply data mining to discover useful and implicit knowledge from web logs, which keep traces of information when users visit web pages on web servers. The purpose of web-log mining is to improve web performance (which is defined precisely later in the paper) by utilizing the mined knowledge. Indeed, data mining is very promising since popular web sites get millions of hits each single day, traditional methods or human would be infeasible to analyze such logs.

The log we study in this paper is from a popular commercial knowledge-based search engine that is currently in use. The web of the search engine allows users to submit keyword queries to a search engine to find articles about the queries, and to browse articles organized under a hierarchical structure of several levels. The log keeps users' queries and their clicks, as well as their browsing activities. The file is quite large: one-day log would be over several hundreds megabytes in size.

It is well known that queries for web search engines are often too short to contain sufficient information to discriminate ambiguous documents. Therefore, additional knowledge, such as informative search terms

*This work was done while those authors were visiting Microsoft Research China.

and their frequency, the corresponding subject categories and the associated documents, and the history of previous user queries and users behaviors, are often helpful to improve the performance of search engines.

The purpose of our project is to find useful "templates", rules associating keywords of the queries with the target articles. If many users keying in similar queries clicked on the same article, then templates summarizing the queries will provide a direct answer to the future, similar queries. This process would be much faster than the traditional search methods. Since templates are evaluated by their accuracy (see later), answers provided by the templates are potentially accurate as well. In addition, templates with generalized keywords (see later) can match new queries that were not asked previously, making templates more general. Therefore, improving speed and precision is the first goal of our project.

The second goal of the project is to discover comprehensible descriptions on the topics in which users are mostly interested. The web editors can study those topics, adding more articles if necessary. Therefore, we need to build a user-friendly interface, providing different and clear views of the templates discovered.

Past work on web-log mining has been done. However, most has focused on mining to change the web structure for easier browsing [Craven, et al, 1999; Sundaresan and Yi, 2000], predicting browsing behaviors for pre-fetching [Zaine, et al, 1998; Boyan, 1996], or predicting user preference for active advertising [Pei, et al, 2000; Perkowitz, 1997]. Some work has been done on mining user logs for improving search engine's performance. The most notable example is the Google search engine [Google], in which data mining is used to gather statistical information for better weighting and ranking of the documents. We design a data mining system that discovers useful patterns or templates for the web server. See Section 4 for more reviews of other work and its relation to ours.

The rest of the paper is organized in the following order: Section 2 describes our mining algorithm in detail. Section 3 presents a user interface that shows results of the mining system. Section 4 relates our mining algorithm to the previous work. Last, Section 5 discusses future work and concludes the paper.

2. Mining Generalized Query Templates

In this Section, we describe in detail how data mining is used to improve search engine's performance. The web of the search engine consists of a search engine, with well-versed articles (written by experts of various domains) as answers to the user queries, and a well-organized hierarchy of articles for browsing (but one cannot use both interchangeably). The users' activities are recorded in the raw IIS log files, which keep a lot of information about each user's access to the web server. The log files are large: one-day log files are about 700 megabytes in size.

A log pre-processing program is written to extract user queries and their corresponding articles clicked. The search engine is a keyword-based search engine, and therefore, almost all queries are simply lists of keyword(s). The results of the log pre-processing program is as follows:

$keyword_1$, $keyword_{2,}$ …; $article_1$, $article_2$, …

……

Each line indicates that a user query contains $keyword_1$, $keyword_{2,}$ and so on, and the corresponding user click-throughs are $article_1$, $article_{2,}$ etc.

We find that most queries have a very small number of keywords: 52.5% of queries contain only one keyword, 32.5% two, 10% three, and 5% four and above. Also most users clicked on one article for their queries: only 0.0619% of user session clicked on two or more articles for a query. We simply convert each of those queries into several queries, each of which contains the same keyword(s), and one article that the user clicked. This would introduce an inherit error rate, since the same query is linked to different articles as answers.

From two-day log files, we obtained 271,803 queries, each of which is a list of keyword(s) and one article clicked. Our data mining starts from this query list, and produces high-level templates, which summarize similar queries answered to the same article. That is, templates are production rules associating similar queries with articles, functioning like a user-constructed index system of the search engine.

One important assumption in our work is that from the limited information returned by the search engine, there is a larger number of users clicked the relevant and correct article(s) as the answers. Even though clicks on other articles are possible due to various user intention and reasons, there should be a biggest group of users who clicked on the same, relevant article(s). Our data-mining algorithm should be able to deal with noise in the data, and to find rules associating queries and relevant clicks. Further, we assume that future users posting similar queries would have similar intention as previous users in the log files. Of course our mining algorithm should be applied to recent logs, reflecting recent interests and the shift of such in users.

2.1 A Simple Bottom-up Generalization Algorithm

A template represents a set of user queries with the same or similar intention, and thus is associated with an article as the answer. It is difficult to capture the "similar intention" as semantics of natural language are still an open problem in natural language understanding. Therefore, we use both syntactic constraints (such as same keywords; see below) as well as semantic constraints (such as generalized concepts and synonyms; see later) in our definition of queries with "similar intention".

The simplest form of templates consists of some keyword(s), and possibly, a "don't care" keyword, in the format of:

article ID ← $keyword_1$, …, $keyword_i$, [any]; accuracy; coverage

where $i \geq 1$ and "any" represents a list of any keyword(s). It is placed in […] to distinguish it from user-inputted keywords.

If a template contains a generalized term placed in [ ], it is called a *generalized template*; otherwise (if it contains user-inputted keywords only) it is called a *simple template*. Accuracy and coverage are two numbers for evaluating such a template. Coverage is the number of original user queries that are correctly covered (or predicted) by this template, and accuracy is the percentage of queries that correctly covered. If error is the number of queries incorrectly covered by a template (when users clicked a different article), then accuracy = coverage/(coverage + error).

A bottom-up generalization algorithm is designed to find generalized templates with [any]. First, it groups the queries clicked by the same article together, and sorts them by the article frequency, so queries of the most clicked articles are mined first. From the queries of the same article, two are chosen according to a greedy heuristic (see later) if they have some keyword(s) in common. Then a tentative template is created with those common keyword(s), and the [any] keyword inserted. Its accuracy and coverage can then be calculated. If the accuracy or the coverage of this template is below some thresholds, it is discarded; else the template replaces all queries correctly covered by it, and the process repeats, until no new templates can be formed. Then generalized templates (with the generalized concept [any]) and simple templates (the remaining user queries) are returned. All those templates are associated with the article directly.

To choose which two queries for generalization, a greedy heuristic is used. Every pair of queries is tried, and the pair with the maximum coverage while the accuracy is above some pre-set threshold is chosen at each step for generalization.

Experiments show that this algorithm does not produce satisfactory results: only a few useful templates are produced. The first reason is that the generalization step of introducing [any] into templates is often too bold: such templates cover too many queries incorrectly, and thus have a low accuracy. Second, many keywords have slightly different spellings (sometimes with a minor spelling error) and thus are not regarded as the same, preventing possible generalization. Third, many keywords are synonyms; but since they are spelled differently, they cannot be treated as the same for possible generalization. Fourth, one-keyword queries are not paired for generalization, since it would inevitably produce an overly generalized template "article ID ← [any]". However, most (52.5%) of the queries in the user logs are one-word queries. We wish generalization happen more gradually. This would also alleviate the first problem mentioned above.

2.2 Improvements Over the Simple Generalization Algorithm

In the following four subsections we provide solutions to the four problems mentioned above.

2.2.1 A Hierarchy Over Keywords

The first problem mentioned earlier is over generalization. That is, any two different keywords from two queries will be replaced by [any]. To provide more graded generalization, a hierarchical structure with the "is-a" relation would be useful. With this hierarchy, the generalization of two keywords would be the lowest concept that is an ancestor of both keywords in the hierarchy.

It is a tedious job to construct such a hierarchy over tens of thousands of keywords in the user queries. We used WordNet [Miller, 1990] to generate such a hierarchy automatically. A problem occurred is that most keywords have different senses or meanings, which in turn, have different parents in the hierarchy. We adopt the first, or the most frequently used meaning of each keyword in the WordNet. Previous research [Ng et al, 1996; Lin, 1997] found that the most frequently used meanings are accurate enough compared with more sophisticated methods.

For example, if we use "<" to represent the is-a relation, then WordNet would generalize "alphabet" as: alphabet

< character set < list < database < information < message < communication < social relation < relation < abstraction. Similarly, "symbol" would be generalized as: symbol < signal < communication < social relation < relation < abstraction. Then "communication" would be the least general concept of "alphabet" and "symbol" in the hierarchy.

Allowing gradual generalization using WordNet introduces too many possibilities of generalization for two queries. A keyword in one query may generalize with any other keyword in the other query, introducing an explosive number of combinations. A heuristic is further introduced: *two queries can be generalized if they have the same number of keywords, and only one keyword in the two queries is different*. That is, the two queries must be in the format of:

$keyword_1$, …, $keyword_i$, $keyword_j$

$keyword_1$, …, $keyword_i$, $keyword_k$

where $keyword_j \neq keyword_k$. A potential template is then produced:

$keyword_1$, …, $keyword_i$, [$concept_1$]

where $concept_1$ is the lowest concept which is an ancestor of $keyword_j$ and $keyword_k$ in the hierarchy. The same evaluation criteria (the accuracy and coverage of templates) are applied to the generalized templates.

With the introduction of the concept hierarchy, more interesting templates can be discovered. For example, from

Greek, alphabet

Greek, symbol

A template "article ID ← Greek, [communication]" is produced. "[communication]" would also cover other keywords such as "document", "letter", and "myth", which could be contained in future user queries.

We note that WordNet is only a useful tool to produce such a hierarchy over keywords. One can create or modify concepts and "is-a" links in such a hierarchy, or use other thesaurus to produce such a hierarchy. The more levels we have in the hierarchy, the more gradual or conservative the generalization would be.

2.2.2 Query Pre-processing

The second reason preventing useful generalization of the two queries is that minor differences in the spelling of the same keyword are regarded as different keywords. For example, "book" and "books" are regarded as completely different keywords. Misspelled words are also treated as completely different words.

A simple morphology analysis tool in WordNet is used, which converts words into their "original" forms. For example, "books" to "book", "studied" to "study". We apply the conversion in our program before generalization is taken place. Currently we are working on spelling correction on the keywords in user queries.

2.2.3 Synonym Conversion

Another major problem in this simple generalization algorithm is that many keywords are synonyms: they have very similar meanings but with different spellings. For example, "Internet" and "WWW", "penalty" and "punishment", and "computer" and "data-processor" are all synonyms. Since any generalization of two queries requires common components in the queries, such synonyms should be regarded as the same.

WordNet is again used for the synonym conversion. A dictionary of keywords appearing in user queries is dynamically created when the user logs are scanned. Before a new keyword is inserted into the dictionary, its synonyms are checked to see if any of them is already in the dictionary. If it is, the synonym in the dictionary replaces this keyword. This would reduce the size of the dictionary by about 27%. Even though the reduction is not huge, we found that the reduction happens on keywords that are very frequently used in the queries.

2.2.4 More Flexible Generalizations

Due to the introduction of the hierarchy, pairs of one-keyword queries may now be generalized. As to which two queries are paired for generalization, the same greedy heuristic is used: *the pair that produces a template with the maximum coverage while the accuracy is above a threshold is chosen*. All queries covered by the new template are removed, and the process repeats until no new templates can be generated.

The pseudo-code of the bottom-up generalization algorithm with all improvements is presented in Figure 1.

Several further possible generalization methods are currently under study. For example, queries with different number of keywords, queries with two different keywords in the pair, and so on.

```
QuerySet = IIS log files
QuerySet = Filtering(QuerySet)      /* producing list of user keywords and click */
QuerySet = Pre-process(QuerySet)    /* word stemming, synonym conversion */
QuerySet = sort(QuerySet)           /* cluster queries by article ID */
For each article ID do
Cluster together all one-keyword queries
Repeat
Use greedy strategy to choose two queries
Produce a template                  /* Use WordNet to generalize */
Insert the template into QuerySet; remove all queries covered from QuerySet
Until no new templates can be found
Cluster together all multi-keyword queries
Repeat
Use greedy strategy to choose two queries with one keyword being different
Produce a template
Insert the template into QuerySet; remove all queries covered from QuerySet
Until no new templates can be found
Output all generalized templates and simple templates
```

Figure 1: Pseudo-code of the template mining algorithm

3. Prototype and Test Results

We have constructed an off-line mining system as described in the previous section. From the two-day raw log files of a total size 850 megabytes, it takes a few hours to mine all the templates. Most of the time was spent on the pre-processing, as well as calculating errors of each potential template, since it requires scanning all of the rest of the queries. No attempt is made yet to optimize the efficiency of the code. In the end, about 400 templates are found whose accuracy is above 0.75 (a threshold we chose arbitrarily).

A graphic user interface for easy viewing of the templates is built, and is illustrated in Figure 2. It allows users to view all templates (both simple and generalized) or generalized templates only, view templates sorted by different ways, and display templates in different level of details. In Figure 2, detail display is chosen, so queries that were used to produce generalized templates are also listed under the templates. One can see that some generalized templates might be too general (for example, “ice” and “paw” become [entity something]), while others make sense (for example, “bird” and “fish” become [vertebrate craniate], “robin” and “goldfinch” become [oscine oscine_bird], and so on).

It should be noted that these templates were produced from only two days’ user logs over thousands of articles. With logs of more days or months, more reliable and interesting templates will be found, since more queries will be asked for each article, and over-general templates would be rejected by “negative examples” in the logs.

To fully test how effectively those templates help to improve the performance of the search engine, we would

*This work was done while those authors were visiting Microsoft Research China.

deploy the template matching mechanism into the actual search engine. At this point, our research effort has not yet been transferred to the product group, therefore, we can only run a simulation to see roughly how much improvement one might get from templates mined from the two-day logs.

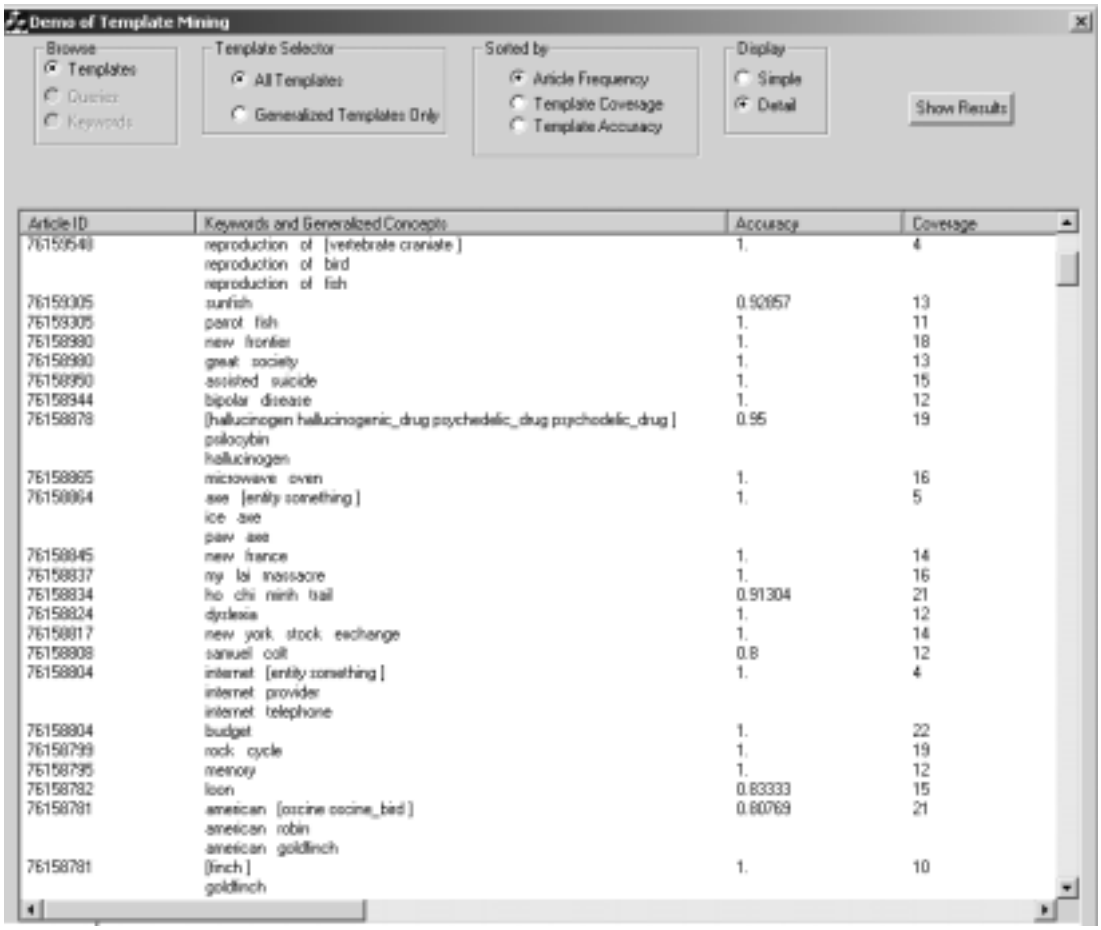

Figure 2: A user interface for viewing the templates mined.

To run our simulation more realistically, we partition the two-day log files into two batches: a training batch which is about 80% of all logs from the beginning, and a testing batch which is the last 20% of the logs. We then run our mining algorithm on the training batch, and test the templates produced on the testing batch (last 20% of the queries), simulating the actual deploy of the templates after data mining has been conducted.

Three criteria are used to see the effectiveness of the templates mined from the training logs. The first is the recall; it measures the percent of the testing queries that are matched with the templates. The higher the recall, the more queries are benefited from using templates. The second is the precision, which measures, among queries matched with the templates, the percent of the testing queries that are correctly predicted by the templates (that is, user clicks of those queries are the same as what templates predict). The third criterion is the speed improvement. Assuming template matching is k times faster than the regular search, the time needed for returning the answer from templates is, on average, only

$$\text{Recall}/k + (1-\text{recall}) \times 1$$

percent of time needed by the regular search engine. The smaller this number, the more saving in time is achieved from using the templates.

We calculate these three criteria by running our mining algorithm in three versions with an increasing power: the first version comes with the concept hierarchy only, the second with hierarchy and word morphology analysis (as described in Section 2.2.2), and the third with hierarchy, word morphology analysis, and the synonym conversion. We expect to see better results with later versions. This

also allows us to find out which component is the "source of power" that produces good results.

Table 1 presents our experiment results. As we can see, some performance measure (i.e., recall) of the search engine does improve dramatically when more improvements are incorporated in our mining algorithm. However, as shown in table 1, the first version has the highest precision, which does not match our intuition. However, we note that the recall of the first version is much lower than the other two, therefore the reliability of the precision is low. In addition, we are seeking the balance between precision and recall. So the later versions still have better overall performance, as they have much higher recall. Considering that we have relatively small log files (i.e. 260,000 queries over 20,000 articles) of short periods of time (i.e. two days), the results in table 1 are nevertheless promising. We believe that when more log files of longer periods of time are available, we will obtain better results.

	Recall	Precision
I: with hierarchy	6.5%	51.2%
II: I + morphology analysis	15.3%	41.7%
III: II + synonyms	15.5%	42.0%

Table 1: Template mining results

4. Relation to Previous Work

Template learning discussed in this paper is similar to rule learning from labeled examples, a well studied area in machine learning. Many methods have been developed, such as C4.5 rule, the AQ family, and RIPPER. C4.5 rule [Quinlan 1993] is a state-of-art method for learning production rule. It first builds a decision tree, and then extracts rules from the decision tree. Rules are then pruned by deleting conditions that do not affect the predictive accuracy. However, C4.5rule is limited to produce rules using only the original features (that is, only "dropping variables" is used in generalization), while our method also introduces generalized concepts in the hierarchy.

The AQ family [Michalski, 1983] generates production rules directly from data. It uses the "generate and remove" strategy, which is adopted in our algorithm. However, AQ is a top-down induction algorithm, starting from a most general rule and making it more specific by adding more conditions. AQ does not utilize concept hierarchy as we use in our generalization process. It seems difficult to incorporate concept hierarchy in the top-down learning strategy, as a large number of concepts in the hierarchy must be tried for making a rule more specific. The difference between our mining algorithm and RIPPER [Cohen, 1995] are also similar.

Concept hierarchy has been used in various machine learning and data mining systems. One usage is as background knowledge, as in [Han, Cai and Cercone, 1993]. Concept hierarchies have also been used in various algorithms for characteristic rule mining [Han, 1995, 1996; Srikant, 1995], multiple-level association mining [Han 1995], and classification [Kamber 1997]. What makes our work different is that our concept hierarchy is much larger and more general; it is generated automatically from WordNet over thousands of keywords. Another difference is that it is used directly in guiding the rule induction process.

The improvements over the simple generalization algorithm, presented in section 2.2, are similar to approaches to the term mismatch problem in information retrieval. These approaches, called dimensionality reduction in [Fabio, 2000], aim to increase the chance that a query and a document refer to the same concept using different terms. This can be achieved by reducing the number of possible ways in which a concept is expressed; in other word, reducing the "vocabulary" used to represent the concepts. A number of techniques have been proposed. The most important ones are manual thesauri [Aitchison and Gilchrist, 1987], stemming and conflation [Frakes, 1992], clustering or automatic thesauri [Rasmussen 1992, Srinivadsan 1992], and Latent Semantic Indexing [Deerwester et al., 1990]. These techniques propose different strategies of term replacement. The strategies can be characterized by (1) semantic considerations (manual thesauri), (2) morphological rules (stemming and conflation), and (3) term similarity or co-occurrence (clustering or Latent Semantic Indexing). In section 2.2, we used strategies (1) and (2). The manual thesauri we used is WordNet. We also dealt with synonymous, but unlike strategy (3) which clusters similar terms based on co-occurrence, we used clusters of synonymous provided by WordNet directly. Our training data are user logs, which do not contain full text information of processed documents, which are necessary for co-occurrence estimation.

5. Conclusions and Future Work

In this paper we describe a new data-mining algorithm that discovers generalized query patterns or templates from large, raw user logs of a commercial knowledge-based search engine. Those templates can provide insights to the web editors as to what topics users are mostly interested in. When incorporated with the regular search engine, those templates can improve the search speed, as well as recall and precision of the search engine, as our simulation shows.

In our future work, more semantic information will be introduced into our mining system so queries of similar meanings can be clustered and generalized. In addition, more log files of longer periods of time (such as months) are needed to produce more reliable and more useful templates, which will improve further the performance of the search engine.

References

[Aitchison and Gilchrist, 1987] J. Aitchison, and A. Gilchrist, Thesaurus Construction. A practical manual. ASLIB, London, 2nd edition, 1987.

[Boyan, 1996] J. Boyan, D. Freitag, and T. Joachims, A Machine Learning Architecture for Optimizing Web Search Engines, Proc. of AAAI Workshop on Internet-Based Information Systems, Portland, Oregon, 1996.

[Cohen, 1995] W. W. Cohen, Fast Effective Rule Induction, Proc. of International Conference on Machine Learning, 1995.

[Craven 1999] M. Craven, D. DiPasquo, et al., Learning to Construct Knowledge Bases from the World Wide Web, Artificial Intelligence, August, 1999.

[Deerwester et al., 1990] S. Deerwester, S.T. Dumais, T. Landauer, and Harshman, Indexing by Latent Semantic Analysis, Journal of the American Society for Information Science, 41(6):391-407, 1990.

[Fabio, 2000] C. Fabio, Exploiting the Similarity of Non-Matching Terms at Retrieval Time. Information Retrieval, 2, 25-45 (2000).

[Fayyad, et al, 1996] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996.

[Frakes, 1992] W. B. Frakes, Stemming Algorithm, In: W.B. Frakes and R. Baeza-Yates, Eds., Information Retrieval: data structures and algorithms, Prentice Hall, Englewood Cliffs, New Jersey, USA, ch.8, 1992.

[Google] http://www.google.com.

[Han, 1995, 1996] J. Han, Mining Knowledge at Multiple Concept Levels, Proc. of 4th Int. Conf. on Information and Knowledge Management, Maryland, 1995.

[Han, Cai and Cercone, 1993] J. Han, Y. Cai and N. Cercone, Data-Driven Discovery of Quantitative Rules in Relational Databases, IEEE Tran. On Knowledge and Data Engineering, 5(1), 1993.

[Kamber 1997] M. Kamber, et al, Generalization and Decision Tree Induction: Efficient Classification in Data Mining, Proc. of 1997 Int. Workshop on Research Issues on Data Engineering, Birmingham, England, 1997.

[Lin, 1997] D. Lin, Using Syntactic Dependency as Local Context to Resolve Word Sense Ambiguity. Proceedings of ACL/EACL-97, pp. 64-71, 1997.

[Ng et al, 1996] H. T. Ng and H. B. Lee, Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach. Proceedings of 34th Annual Meeting of the Association for Computational Linguistics, pp. 44-47, 1996.

[Michalski, 1983] R. S. Michalski, A Thoery and Methodology of Inductive Learning, Artificial Intelligence, 20(2), 1983.

[Miller, 1990] G. Miller, WordNet: an online lexicon database, International Journal of Lexicography, 1990.

[Pei, et al, 2000] J. Pei, J. Han, B. Mortazavi-asl, and H. Zhu, Mining Access Patterns effiently from Web Logs, Proc. 2000 Pacific-Asia Conf. on Knowledge Discovery and Data Mining, Kyoto, Japan, 2000.

[Perkowitz, 1997] M. Perkowitz and O. Etzion, Adaptive Sites: Automatically Learning from User Access Patterns, The Sixth International WWW Conference, Santa Clara, California, USA, 1997.

[Quinlan 1993] J. R. Quinlan, C4.5: Programs for Machine Learning, San Mateo: Morgan Kaufmann, 1993.

[Rasmussen, 1992] Clustering Algorithm, In: W.B. Frakes and R. Baeza-Yates, Eds., Information Retrieval: data structures and algorithms, Prentice Hall, Englewood Cliffs, New Jersey, USA, ch.16, 1992.

[Srikant, 1995] R. Srukant and R. Agrawal, Mining Generalized Association Rules, Proc. 1995 Int. Conf. Very Large Data Bases, Zurich, Switzerland, 1995.

[Srinivadsan, 1992] Thesaurus Construction, In: W.B. Frakes and R. Baeza-Yates, Eds., Information Retrieval: data structures and algorithms, Prentice Hall, Englewood Cliffs, New Jersey, USA, ch.9, 1992.

[Sundaresan and Yi, 2000] N. Sundaresan and J. Yi, Mining the Web for Relations, The Ninth International WWW Conference, Amsterdam, The Netherlands, 2000.

[Zaine et al, 1998] O. R. Zaine, Man Xin, and Jianwei Han, Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs, Proc. Advances in Digital Libraries Conf. (ADL98), Santa Barbara, CA, April 1998, pp. 19-29.