To appear: International Journal of Pattern Recognition and Artificial Intelligence. Vol. 16, No. 8 (2002)

Improving Encarta Search Engine Performance by Mining User Logs

Charles X. Ling*
Dept. of Computer Science, Univ. of Western Ontario, Ontario N6A 5B7, Canada
ling@csd.uwo.ca

Jianfeng Gao
Microsoft Research Asia, China
jfgao@microsoft.com

Huajie Zhang*
Dept. of Computer Sciences, Univ. of New Brunswick, Canada
hzhang@unb.ca

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

Hongjiang Zhang
Microsoft Research Asia, China
hjzhang@microsoft.com

Abstract

We propose a data-mining approach that produces generalized query patterns (with generalized keywords) from the raw user logs of the Microsoft Encarta search engine (http://encarta.msn.com). Those query patterns can act as cache of the search engine, improving its performance. The cache of the generalized query patterns is more advantageous than the cache of the most frequent user queries since our patterns are generalized, covering more queries and future queries - even those not previously asked. Our method is unique since query patterns discovered reflect the actual dynamic usage and user feedbacks of the search engine, rather than the syntactic linkage structure of web pages (as Google does). Simulation shows that such generalized query patterns improve search engine's overall speed considerably. The generalized query patterns, when viewed with a graphical user interface, are also helpful to web editors, who can easily discover topics in which users are mostly interested.

***Keywords**: web log mining, web mining, data mining on the Internet, search engine improvement.*

1. Introduction

We study data mining applied to the Internet search engine in this paper. In particular, we apply data mining to discover useful and implicit knowledge from user logs of the Microsoft Encarta search engine (http://encarta.msn.com) to improve its performance. The user logs keep traces of users behaviors when they use the Encarta search engine. The purpose of the web-log mining is to improve Encarta search engine's performance (which is defined precisely later in the paper) by utilizing the mined knowledge as cache. Indeed, data mining is a promising approach since popular search engines like Encarta get hundreds of thousands of hits each single day, and therefore, it would be infeasible for traditional methods or human to analyze such logs.

Encarta accepts users' submission as keyword queries, and returns a list of articles for users to choose from. There are over 40,000 articles in Encarta. Users can also browse articles organized under a hierarchical structure of several levels.

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

The log keeps track of users' queries and their clicks on the returned articles, as well as their browsing activities. The log file is quite large: a one-day log is over several hundreds megabytes in size.

Our log-mining system is an integration of previous techniques (mostly on smaller problems though), but when they are carefully integrated and tuned, they are effective to solve this large-scale real-world problem. The goal is to find useful "patterns" - rules associating keywords of the queries with the target articles. If many users keying in similar queries clicked on the same article, then generalized query patterns summarizing the queries will provide a direct answer (the article) to the similar queries in the future. When a set of such patterns is used as cache of the Encarta search engine, its overall performance can be dramatically improved.

Let us assume that cache is *k* times faster than a regular search by the search engine. To obtain overall improvement with the cache, we must consider how likely a future user query is hit by the cache, and how accurately the cache predicts the article (so user does not have to revoke the regular search engine for the correct answer). The first part is measured by the **recall** (*R*) (or the hit ratio) of the cache; it measures the percent of the testing queries that are matched with the queries patterns in the cache. The second part is the **precision** (*P*), which measures, among queries hit by the cache, the percent of the testing queries that are correctly predicted (that is, user's click on the article returned from the query is the same as what query patterns predict). Thus, assuming a unit time is needed by a regular search engine, the overall time when the search engine is equipped with the cache is:

$$P\,R / k \;+\; R\,(1 - P)(1+1/k) + (1 - R)(1+1/k)$$

The first term is for the user query correctly matched with a pattern (with the probability *PR* the cache returns the correct result within time *1/k*), the second term for an incorrectly matched query (so the user requests another search from the regular search engine), and the third term for no match (so the regular search engine is invoked). The overall time is equal to:

$$1 + 1/k - P\,R$$

Clearly, the larger the *k*, and the larger the product of *P* and *R*, the less the overall time of the search engine equipped with cache. Thus, our first goal of mining user logs of Encarta is to improve the product of the precision and recall of the cache.

A major advantage of our caching technique over the previous ones is that our cache consists of generalized query patterns so that each pattern covers many different user queries. They can even match with new queries that were not asked previously. Most previous caching approaches keep the frequent user queries, or invert lists directly (See [Saraiva et al, 2001] and the references), limiting the recall of cache with a fix size dramatically. Some previous work on semantic caching (such as [Chidlovskii et al, 1999]) can compose answers of new queries by applying Boolean operations to the previous answers. However, they do not allow generalization of the previous queries as we do.

The second goal of the research is to discover comprehensible descriptions on the topics in which users are mostly interested. The web editors can study those topics, and add more articles on popular topics. Our generalized query patterns are production-rule based, and are easy to comprehend. Therefore, we need to build a user-friendly interface, providing different and clear views of the query patterns 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, 2000; 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 to improve search engine's performance. The most notable example is the Google search engine [Google], in which statistical information is mined from the linkage structure among web pages for weighting and ranking of the web pages. However, when users click on web pages returned, they access to the web pages directly. Therefore, Google does not collect users' feedback on the search results (unless when users click on the cached pages on the Google server). Thus, Google's improvement on the search engine is largely syntactic and static based on linkage relations among web pages. Our method is semantic and dynamic, since it mines from the users actual clicks (feedback) on the search results, and it dynamically reflects the actual usage of the search engine. 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 Patterns

In this Section, we describe in details how data mining is used to improve the Encarta search engine's performance. The Encarta website (http://encarta.msn.com) consists of a search engine, with 41,942 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 the search engine and the hierarchy interchangeably). The users' activities are recorded in the raw IIS (Internet Information Services) 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:

$$\text{keyword}_1, \text{keyword}_2, \ldots; \text{article}_1, \text{article}_2, \ldots$$

Each line indicates that a user query contains keyword_1, keyword_2, and so on, and the corresponding articles that the user clicks 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 only 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.

One inherent difficulty in our work is that user queries are mostly short, consisting one or two keywords. On the other hand, user clicks of the articles can vary due to various user intentions. Our data-mining algorithm should be able to deal with noise in the data, and to find rules associating queries with the most relevant articles.

From two-month Encarta's user log, we obtained 4.8 million queries, each of which is a list of keyword(s) and one article clicked. Our data mining starts from this query list, and produces generalized query patterns, which summarize similar queries answered to the same article. Again, those queries patterns will act as the cache of the Encarta search engine.

2.1 A Bottom-up Generalization Algorithm

A query pattern 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" in natural language, and therefore, we use both syntactic constraints (such as stemming of keywords; see later) as well as semantic constraints (such as generalized concepts and synonyms; see later) in our definition of queries with "similar intentions".

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

$$\text{article ID} \leftarrow \text{keyword}_1, \ldots, \text{keyword}_i, [\text{any}]; \text{accuracy}; \text{coverage}$$

where "any" represents any keyword(s), and is placed in […] to distinguish it from user-inputted keywords. If a pattern contains a generalized term placed in [ ], it is called a *generalized pattern*; otherwise (if it contains user-inputted keywords only) it is called a *simple pattern*.

Accuracy and coverage are two numbers for evaluating query patterns. (Note that the precision and recall mentioned earlier are measures for testing queries on the whole set of patterns, while accuracy and coverage here are for each individual query pattern.) *Coverage* is the number of original user queries that are correctly covered (or predicted) by this pattern, and *accuracy* is the percentage of queries that correctly covered. If *error* is the number of queries incorrectly covered by a pattern (when users clicked a different article), then *accuracy = coverage/(coverage + error)*. If the accuracy and coverage of all query patterns are known, and the distributions of testing set on those patterns are known, then the overall precision and recall of the pattern set (cache) on the test set can be easily derived.

A bottom-up generalization algorithm is designed to find generalized patterns with [any], and the overall process is described below. First, it groups the queries clicked by the same article together, and sorts them by the article frequency, so that 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 pattern 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 pattern is below some thresholds, it is discarded; else the pattern replaces all queries correctly covered by it, and the process repeats, until no new patterns can be formed. Then the generalized patterns (with the generalized concept [any]) and the simple patterns (the remaining user queries) are returned, and they are all associated with that article. This process is applied to all articles. In the end, all patterns of all articles are sorted according to their coverage, and the top *t* patterns with highest coverage from all articles are outputted as the cache. The cache size *t* was chosen as 5,000 in this paper, since a query distribution analysis shows that when *t* is overly larger than 5,000, the overall benefit of larger caching is reduced.

Note that the relation between query patterns and articles is a many-to-many one. On one hand, the same query pattern can associate with more than one article (since general keywords such as "animal" can be the keyword of several articles). Therefore, several articles can be returned if the user query matches with several query patterns with different articles. On the other hand, the same article can be associated with many query patterns (different keywords can associate with the same article).

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 (not shown here) indicate that this algorithm does not produce satisfactory results: only a few useful patterns are produced. The first reason is that the generalization step of introducing [any] into patterns is often too bold: such patterns 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 pattern "article ID ← [any]". However, most (52.5%) of the queries in the user logs are one-word queries. We want generalization to happen more gradually. This would also alleviate the first problem mentioned above.

In the following four subsections we provide solutions to the four problems mentioned above. Most improvements have been published previously (see Section on Relation to Previous Work), but when they are carefully integrated and tuned, they are shown to be effective to solve this large-scale real-world problem.

2.2 Improvement 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.

A problem that allows for gradual generalization using WordNet is that it introduces too many possibilities of generalization from two user queries. A keyword in one query may generalize with any other keyword in the other query using WordNet, introducing an explosive number of combinations. A heuristic is further introduced: *two queries can be generalized only when 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 pattern 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.

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

Greek, alphabet

Greek, symbol

A generalized query pattern:

article ID ← Greek, [communication]

is produced. "[communication]" would also cover other keywords such as "document", "letter", and "myth", which might be contained in future user queries.

Obviously, when the generalized query patterns are cached for the search engine Encarta, WordNet API calls must be available for testing whether or not a keyword in a new user query is covered by generalized concepts. This can be optimized to have little impact on the speed of the cache.

2.3 Improvement 2: 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 pattern with the maximum coverage while the accuracy is above a threshold is chosen*. All queries covered by the new pattern are removed, and the process repeats until no new patterns can be generated.

As discussed in the Introduction, the goal of producing generalized query patterns is to have the product of the recall and precision of patterns as large as possible. Therefore, a further generalization is applied here. After a least general keyword in the WordNet is obtained from a pair of two keywords, it is further generalized by "climbing" up in the hierarchy of the WordNet until just before the accuracy of the pattern falls under the fixed threshold. This would produce patterns with a maximum recall while maintaining the adequate accuracy.

This improvement produces many useful generalized queries, such as:

"Democracy, of, [American_state]" from

- Democracy, of, California
- Democracy, of, Texes
- Democracy, of, Pennsylvania;

"[weapon_system]" from

- Gun
- Pistol
- Bullet
- firearm

"[European], unification" from

- German, unification
- Italian, unification

"[rank], amendment" from

- first, amendment
- fourteenth, amendment
- fifth, amendment

2.4 Improvement 3: Morphology Conversion

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. This would prevent many possible generalizations since all other keywords except one in two queries must be exactly the same.

A simple morphology analysis using WordNet is designed to convert words into their "original" forms (i.e., stemming). 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.

With this improvement, we see user queries such as:

- populations, Glasgow
- population, Edinburgh

which cannot be generalized (because both keywords in the two queries are different) can now be generalized, and produce a generalized query pattern "population, [UK_region]"

2.5 Improvement 4: Synonym Conversion

A similar problem 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 all keywords except one must be the same, such synonyms should be recognized to allow more possible generalizations.

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.

With this improvement, more useful generalized queries can be produced, which would not be possible previously. For example:

"DNA, [investigation]" from

- DNA, testing
- gene, analysis
- genetic, inquiry

2.6 Implementations

The pseudo-code of our log mining algorithm for Encarta with all improvements is presented in Figure 1. The program is written in Visual C++.

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

```
QuerySet = IIS log files                    /* Original user log of Encarta */
QuerySet = Filtering(QuerySet)              /* Producing list of user keywords and click */
QuerySet = Stemming(QuerySet)               /* Improvement 3 (morphology conversion) */
QuerySet = Synonym(QuerySet)                /* Improvement 4 (synonym conversion) */
QuerySet = sort(QuerySet)                   /* Cluster queries by article ID */
For each article ID do
  Repeat
    Max_pattern = " "
    For each pair of queries with the same length and only one keyword different
      Temp_pattern = generalize(pair)       /* use WordNet to generalize the keyword */
      While accuracy(climb_up(Temp_pattern)) > threshold do        /* Improvement 2 */
        Temp_pattern = climb_up(Temp_pattern)     /* generalize one step up in WordNet */
      If cover(Temp_pattern) > cover(Max_pattern) then Max_pattern = Temp_pattern
    Remove all queries covered by Max_pattern from QuerySet
    Insert Max_pattern into QuerySet        /* Max_pattern can be paired for further generalization */
  Until no new patterns can be found
Output t generalized and simple patterns with top coverage for all articles
```

Figure 1: Pseudo-code of the mining top *t* user query patterns.

3. Simulation Experiments

We have constructed an off-line mining system as described in the previous section. From the two-month user log files of a total size about 22 GB, it takes a few hours to mine all the patterns. Most of the time was spent on error calculation of each generalized query pattern, since it requires scanning all of the rest of the queries to see if it covers any "negative examples". The efficiency of the code would be improved since no attempt is made yet to optimize it. In the end, 5,000 query patterns are chosen whose accuracy is above 0.75 (a threshold we chose for the experiments).

We run several realistic simulations to see how our generalized query patterns help to improve the Encarta search engine, compared to a baseline cache which consists of the same number of the most frequent (ungeneralized) user queries. We partition the two-month 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 patterns produced on the testing batch (last 20% of the queries unused in the training), simulating the actual deployment of the cache in the search engine after data mining has been conducted.

Again, recall and precision (see Introduction) of the cache are calculated to measure the overall speed of the search engine with cache of the generalized query patterns. The overall saving would be larger if the product of recall and precision is larger, and if the *k* (the speed difference between regular search and cache) is larger.

We also run our mining algorithm in three versions with an increasing power: the first version comes with the concept hierarchy and flexible generalization (Sections 2.2 and 2.3), the second with hierarchy and word morphology analysis (Section 2.4), and the third with hierarchy, word morphology analysis, and the synonym conversion (Section 2.5). 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.

4. Analyses of the Results

Table 1 presents our experiment results. As we can see, the recall (or hit ratio) of the cache does improve dramatically when more improvements in the mining algorithms are incorporated, and is much higher than the baseline cache, which consists of the same number of most frequent original user queries. On the other hand, the precision is virtually the same for different versions. This is actually within our expectation since patterns are chosen according to their coverage when their accuracy is above a fixed threshold (0.75). Overall, the best version produces the highest product of precision and recall. As we can see, the major improvement in recall comes from the generalization with hierarchy from the WordNet (improvements 1 and 2), and from synonym conversion (improvement 4).

If *k* is equal to 100, which is quite common as the speed up of using the cache vs the regular search engine, then the search engine with cache of the generalized query patterns would have an overall user search time of only 30% of the search engine without the patterns according to the formula $1 + 1/k - P\ R$ (see Introduction). That is, the search engine with the cache of the best generalized patterns is 3.3 times faster overall then the one without using the cache, and 3.1 times faster than the baseline cache of the most frequent user queries.

	Recall	Precision	Overall search time with cache, if no cache is 100%.
Baseline cache	7.31%	82.3%	95%
I: with hierarchy	41.7%	81.7%	67%
II: I + morphology analysis	52.8%	90.6%	53%
III: II + synonyms	80.4%	88.3%	30%

Table 1: Results of our log-mining algorithm for the Encarta search engine.

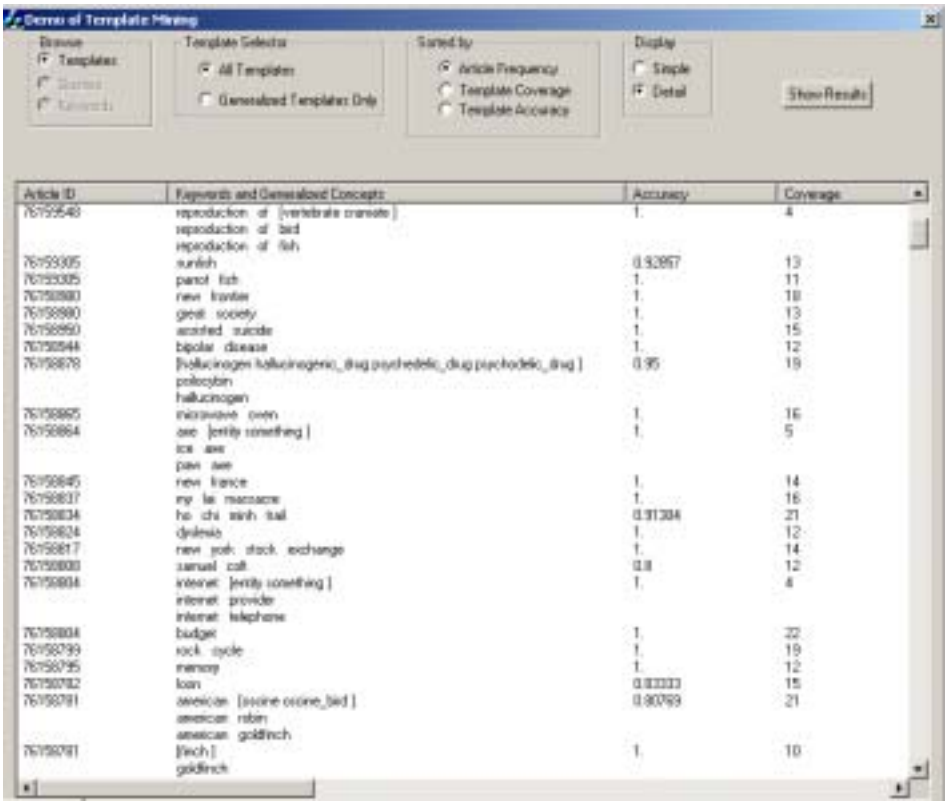

Figure 2: A user interface for viewing the patterns mined.

5. The Graphic User Interface

As discussed in the Introduction, the second goal of this research is to provide an intuitive graphic user interface for web editors to view the resulting generalized query patterns to discover overall users interests for maintaining and improving the Encarta search engine. A GUI for easy viewing of the resulting patterns is built, and is illustrated in Figure 2. It allows web designers to view all patterns (both simple and generalized) or generalized patterns only, sorted by different ways, and in different level of details. In Figure 2, detailed display is chosen, so queries that were used to produce generalized patterns are also listed under the patterns. One can see that some generalized patterns 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.

6. Relation to Previous Work

Many techniques used in this paper have been used (mostly on smaller problems though). We demonstrate that when those techniques are carefully integrated and tuned, they can solve this large-scale real-world problem.

Pattern 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 tens of thousands of keywords.

The improvements over the simple generalization algorithm 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, Srinivasan 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 our work, we used strategies similar to (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.

7. Conclusions and Future Work

We have described an integration of data-mining algorithms that discovers generalized query patterns from large, raw user logs of the Encarta search engine. Those patterns can be used as a cache technique of the search engine to improve its overall performance measured by the average time that the users spend to obtain the correct answers to the keyword search. Our approach is semantic (since it mines from the users actual feedbacks on the search results rather than a mere syntactic linkage relation), and dynamic (since it dynamically reflects the actual usage of the search engine and user interests). The query patterns discovered also provide insights to the web editors as to what topics users are mostly interested in.

In our future work, more semantic information will be introduced into our mining system so queries of similar meanings can be clustered and generalized further. In addition, efforts will be put into improving the code efficiency, and transferring our research and prototype system into the deployment of the Encarta search engine.

8. 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.

[Chidlovskii et al, 1999] B. Chidlovskii, C. Roncancio, and M.L. Schneider. Semantic cache mechanism for heterogeneous web querying. Computer Networks, 31(11-16): 1347-1360, 1999. Proc. of the 8th Int. World Wide Web Conf. 1999.

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

[Craven 2000] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to Construct Knowledge Bases from the World Wide Web, Artificial Intelligence, 118(1/2):69-113. 2000.

[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):29-40. 1993.

[Kamber 1997] M. Kamber, L. Winstone, W. Gong, S. Cheng, J. Han. 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 Theory and Methodology of Inductive Learning, Artificial Intelligence, 20(2):111-161, 1983.

[Miller, 1990] G. Miller, WordNet: an online lexicon database, International Journal of Lexicography, 3(4):235-312. 1990.

[Pei, et al, 2000] J. Pei, J. Han, B. Mortazavi-asl, and H. Zhu, Mining Access Patterns efficiently 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] E. Rasmussen. Clustering Algorithm. Chapter 16 in: W.B. Frakes and R. Baeza-Yates, editors, Information Retrieval: data structures and algorithms, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1992.

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

[Srinivasan, 1992] P. Srinivasan. 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, M. Xin, and J. 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.

[Saraiva et al, 2001] P.C. Saraiva, E.S. Moura, N. Ziviani, W. Meira, R. Fonseca, and B. Ribeiro-Neto. Rank-Preserving Two-Level Caching for Scalable Search Engines. ACM SIGIR 24th Int. Conference on Information Retrieval, 51-58, 2001

Authors' Brief Biography

Charles X. Ling obtained his MSc and Ph.D. from the Department of Computer and Information Science, University of Pennsylvania in 1989. Since then, he has been a faculty member in the Department of Computer Science, The University of Western Ontario. He is currently an Associate Professor, and director of Data Mining and E-Business Laboratory. He has done extensive research in machine learning, data mining, and cognitive modeling. He has helped major banks and insurance companies in Canada to apply data mining to direct marketing and E-Commerce. See his home page http://www.csd.uwo.ca/faculty/ling for updated information.

Jianfeng Gao is a researcher in natural language computing group at Microsoft Research Asia. His research interests include Statistical language modeling, information retrieval, machine translation, and Asian language processing. He obtained his PhD degree in computer science from Shanghai Jiaotong University in 1999.

Huajie Zhang is an assistant professor in University of New Brunswick, Canada. He received his PhD from the University of Western Ontario, Canada. His research interests include Bayesian networks, machine learning and AI.

Weining Qian is a Ph.D. candidate in Department of Computer Science and Engineering, Fudan University. His specialty is database and knowledge base. He is partially supported by Microsoft Research Fellowship. The work in this paper was done while he was an intern of Microsoft Research China. His current research interests include data mining, Web mining, and data management in peer-to-peer systems.

HongJiang Zhang received his Ph.D from the Technical University of Denmark and his BS from Zhengzhou University, China, both in Electrical Engineering, in 1982 and 1991, respectively.

From 1992 to 1995, he was with the Institute of Systems Science, National University of Singapore, where he led several projects in video and image content analysis and retrieval and computer vision. He also worked at MIT Media Lab in 1994 as a visiting researcher. From 1995 to 1999, he was a research manager at Hewlett-Packard Labs, where he was responsible for research and technology transfers in the areas of multimedia management; intelligent image processing and Internet media. In 1999, he joined Microsoft Research Asia, where he is currently a Senior Researcher and Assistant Managing Director in charge of media computing and information processing research.

Dr. Zhang is a member of ACM and a Senior Member of IEEE. He has authored 3 books, over 200 referred papers and book chapters, 7 special issues of international journals on image and video processing, content-based media retrieval, and computer vision, as well as over 30 patents or pending applications. He currently serves on the editorial boards of five IEEE/ACM journals and a dozen committees of international conferences.