Towards A Next-Generation Search Engine

Qiang Yang[1], Hai-Feng Wang, Ji-Rong Wen, Gao Zhang, Ye Lu[1], Kai-Fu Lee, Hong-Jiang Zhang

Microsoft Research China

5F, Beijing Sigma Center
No. 49 Zhichun Road, Haidian District
Beijing 100080 P.R. China
(qyang,yel)@cs.sfu.ca, (i-haiwan, i-jrwen, i-gzhang, kfl, hjzhang)@microsoft.com

Abstract. As more information becomes available on the World Wide Web, it has become an acute problem to provide effective search tools for information access. Previous generations of search engines are mainly keyword-based and cannot satisfy many informational needs of their users. Search based on simple keywords returns many irrelevant documents that can easily swamp the user. In this paper, we describe the system architecture of a next-generation search engine that we have built with a goal to provide accurate search result on frequently asked concepts. Our key differentiating factors from other search engines are natural language user interface, accurate search results, and interactive user interface and multimedia content retrieval. We describe the architecture, design goals and experience in developing the search engine.

1 Introduction

With the explosive growth of information on the World Wide Web, there is an acute need for search engine technology to keep pace with the users' need for searching speed and precision. Today's popular search engines such as Yahoo! and MSN.com are used by millions of users each day to find information, but the main method of search has been kept the same as when the first search engine appeared years ago, relying mainly on keyword search. This has resulted in unsatisfactory search results, as a simple keyword may not be able to convey complex search semantics a user wishes to express, returning many irrelevant documents and eventually, disappointed users. The purpose of this paper is to sketch a next-generation

[1] The research was conducted while the author was visiting Microsoft Research China, on leave from School of Computing Science, Simon Fraser University, Burnaby BC Canada V5A 1S6

search engine, which offers several key features that make it more natural and efficient for users to search the web.

Search has so far experienced two main evolutions. The first is keyword based search engines [4, 11], as is currently the case with the majority of search engines on the web (e.g., Yahoo! and MSN.com). These engines accept a keyword-based query from a user and search in one or more index databases. Despite its simplicity, these engines typically return millions of documents in response to a simple keyword query, which often makes it impossible for a user to find the needed information. In response to this problem, a second generation of search engines is aimed at extracting frequently asked questions (FAQ's) and manually indexes these questions and their answers; the result is adding one level of indirection whereby users are asked to confirm one or more rephrased questions in order to find their answer. A prime example of this style of search engines is Askjeeves.com. An advantage of this style of interaction and cataloging is much higher precision: whereas the keyword based search engines return thousands of results, Askjeeves often gives a few very precise results as answers. It is plausible that this style of FAQ-based search engines will enjoy remarkable success in limited domain applications such as web-based technical support.

While the Askjeeves search engine generates impressive results, its design and architecture are not available for other researchers to experiment with due to its proprietary nature. We felt, however, a pressing need in the search-engine research community to exchange ideas in order to move towards newer-generation search engines. We observe that although FAQ-based second-generation search engines have improved search precision, much remains to be desired. A third-generation search engine will be able to deal with concepts that the user intends to query about, by parsing an input natural language query and extracting syntactic as well as semantic information. The parser should be *robust* in the sense that it will be able to return partial parse results whenever possible, and the results will be more accurate when more information is available. It will have the capability to deal with languages other than English, as, for example, the Chinese language poses additional difficulty in word segmentation and semantic processing. When facing ambiguity, it will interact with the user for confirmation in terms of the concept the user is asking. The query logs are recorded processed repeatedly, for providing a powerful language model for the natural language parser as well as for indexing the frequently asked questions and providing relevance-feedback learning capability.

The basis for our approach is that an important hypothesis we call the ***concept-space coverage hypothesis***: A small subset of concepts can cover most user queries. If this hypothesis is true, then we can simply track this small subset and use semi-automated method to index the concepts precisely – this results in a search engine that satisfies most users most of the times. To support our hypothesis, we took a one-day log from MSN.com query log and manually mapped queries to pre-defined concept categories. In Fig. 1, the horizontal axis represents the 27 predefined categories of concepts or keywords, and the vertical axis is the coverage of all queries

under consideration by the corresponding subset of concepts or keywords. Examples of the concepts are: “Finding computer and internet related products and services ”, “Finding movies and toys on the Internet” and so on. We took the top 3000 distinct user queries that represent 418,248 queries on Sept 4, 1999 (which we chose arbitrarily), and then classified these queries. The keywords are sorted by frequency, such that the *ith* value on the horizontal axis corresponds to the subset of all *i* previous keywords in the sorted list. As can be seen, both the keyword and the concept distribution obey the pattern that the first few popular categories will cover most of the queries. Furthermore, the concept distribution converges much faster than the keyword distribution as we expect. As an example, 30% of the concepts in fact cover about 80% of all queries in the selected query pool. This preliminary result shows that our hypothesis stands at least for MSN.com query log data. We are currently conducting more experiments to further confirm this hypothesis.

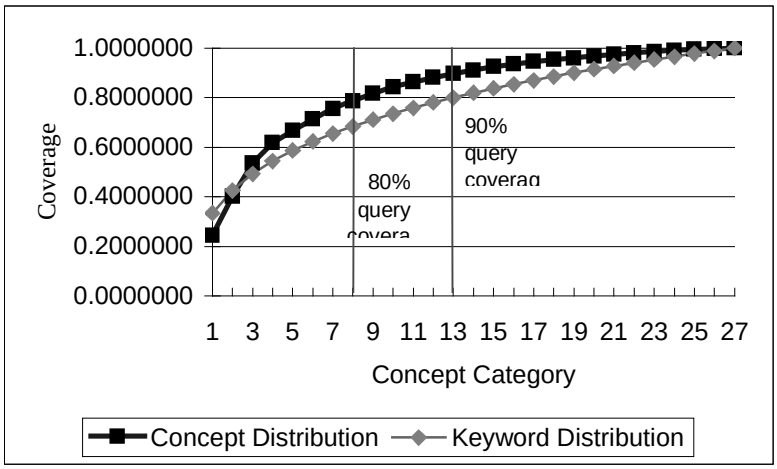

Fig 1. Plotting the query distribution against concept categories

In this paper, we describe our design of a prototype search engine based on FAQ analyses. We show that it is possible to build such search engines based on data mining and natural language processing, as well as well-designed user interfaces. This prototype system, shown in Fig. 2, demonstrates many exciting new opportunities for research in next-generation intelligent computing. We only focus on two main components of the system, the natural language subsystem and the log data mining subsystem, leaving the evaluation of the system to future discussions.

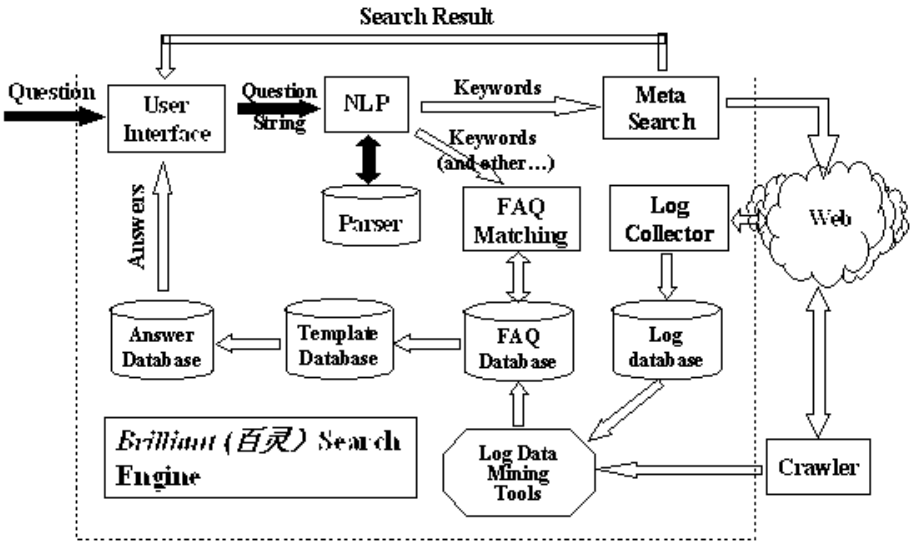

Fig. 2 Brilliant system architecture

Code-named "***Brilliant***", the search engine has the ability to robustly parse natural languages based on grammatical knowledge obtained through analysis of query log data. In order to assemble answers to questions, it has a methodology to process query logs for the purpose of obtaining new question templates with indexed answers. It also has relevance-feedback capability for improving its indexing and ranking functions. This capability allows the system to record users' actions in browsing and in selecting the search result, so that the ranking of these results and the importance of each selection can be learned over time.

2 Natural Language Processing with LEAP

In this section, we first introduce the concept of robust parsing and how to apply it to natural language understanding in search engines. We then describe several challenges of using robust parsing in our search engine design.

2.1 Using Robust Parsing in Search Engine

Spoken language understanding (SLU) researchers have been working on *robust parsing* to handle ill-formed inputs. Usually, a robust parser attempts to overcome the extra-grammaticality by ignoring the un-parsable words and fragments and by conducting a search for the maximal subset of the original input that is covered by the grammar [9, 13]. Natural language understanding share many features with

SLU: input sentences are almost always ungrammatical or incomplete. It is difficult to parse such sentences using a traditional natural language parser. One advantage of search engine queries is that they are short – again a feature shared with spoken languages. Therefore we choose a robust parser as our natural language parser in the search engine.

There are several main advantages of robust parsers. First, if an input sentence contains words that are not parsable, a robust parser can skip these words or phrases and still output an incomplete result. In contrast, a traditional parser will either completely break down, or need revision of the grammar rules. Second, if a given sentence is incomplete such that a traditional parser cannot find a suitable rule to match it exactly, robust parsers can provide multiple interpretation of the parsing result and associate with each output a confidence level. In Brilliant™ search engine, this confidence level is built based on statistical training.

LEAP (**L**anguage **E**nabled **AP**plications) is an effort in the speech technology group in Microsoft Research that aims at spoken language understanding [14]. Leap's robust parsing algorithm is an extension of the bottom-up chart-parsing algorithm [1]. The framework of the algorithm is similar to the general chart-parsing algorithm. The differences include: 1) traditional parsers require that a hypothesis *h* and a partial parse *p* have to cover adjacent words in the input; in robust parsers this is relaxed. This makes it possible for the parser to omit noisy words in the input. 2) Different hypotheses can result from partial parses by skipping some symbols in parse rules. This allows non-exact matching of a rule.

A LEAP grammar defines *semantic classes*. Each semantic class is defined by a set of rules and productions. For example, we can define a semantic class <Route> for the travel path from one place to another. This class is represented as follows:

```
<Route> TravelPath {
        =>   @from   <PlaceName:place1>   @to   <PlaceName:place2>
     @route;
        @from => from | ...;
     ......
}
<PlaceName> Place {
        Beijing | Shanghai | ...;
}
```

In the semantic classes above, <Route> defines a return class type, and TravelPath is a semantic class that contains a number of rules (the first line) and productions (the second line). In this class, @from must parse a piece of the input sentence according to a production as shown in the second line. The input item after the @from object must match according to <PlaceName> semantic class. If there are input tokens that are not parsable by any parts of the rule, it will be ignored by the parser. In this case, the scoring of the parse result will be correspondingly discounted to reflect a lower level of confidence in the parse result.

Suppose the input query is:

How to go from Beijing to Shanghai?

The LEAP parser will return the following result:

<VOID> How to go from *place* to *place*
<Route> How to go from *place* to *place*
<PlaceName:place1> *place*
<PlaceName:place2> *place*

Here <VOID> represents the root semantic class. Note that this input query cannot be parsed using the first rule in the semantic class TravelPath completely if we used a traditional parser. In our implementation, we calculated the score of the parsing result by discounting the number of input items and rule items that are skipped during the parsing operation. This value is normalized to give a percentage confidence value.

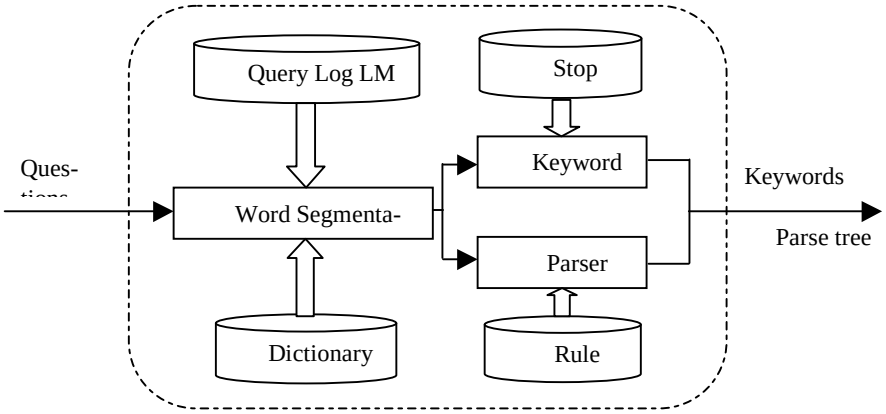

Fig. 3. The flowchart of natural language processing

We have adapted the LEAP system in our search engine using grammar rules trained through query logs. The system architecture is shown in Fig. 3. In this figure, we first perform Chinese word segmentation. The segmented sentence is passed to LEAP next. If LEAP parses the sentence successfully, the module outputs a parse tree. Otherwise keywords will be simply extracted and outputted. The template-matching module of the search engine will be able to use both types of output.

2.2 Evaluate the parsing result

In our search engine, we adapt Leap so that it evaluates the output based on the coverage of a rule against the input query. A parsed result will be selected if it covers the most words in the query and the most parts of rules. In order to improve the scoring strategy, we learn probabilities from query logs to include:

- · probabilities of the rules;
- · penalty for robust rule matching (insertion, deletion, substitution);
- · probabilities of "non-matching" words;
- · term probability according to their frequency in query log.

Considering the rule in the semantic class <Route> TravelPath:

⇨ @from <PlaceName:place1> @to <PlaceName:place2> @route;

We illustrate how to train the probability values associated with this rule. A rule having a high probability value means that using this rule to parse a query is more reliable. This is similar to PCFG [8]. We can also train the penalty values for robust matching: For an item in either a rule or the query sentence, if the item is skipped during parsing, a penalty will be exacted. In the above rule, if @from is skipped, the penalty can be set relatively low. However, if @route is skipped, the penalty should be high. In the sentence "How to get from Beijing to Shanghai" if "how to go" is skipped, the penalty should be high. In our search engine, penalty and score statistics are gathered using the query log files.

3 Question Matching and Query Log Mining

In this section, we describe the question-matching module. We first describe the question matching process used in the module. We then discuss issues related to obtaining the template and answer databases from the query log data.

3.1 Question Matching Process

The purpose of the question-matching module is to find a set of relevant concepts and their related answers from a given user query. To accomplish this, we set up a collection of FAQ concepts and a mapping from these concepts to answers. These answers can be further compressed by grouping together their similar parameters – the result is what we call templates.

To illustrate this mapping process, we consider a realistic example. Suppose that the user asked "How to go from Beijing to Shanghai?". We store in our FAQ database a collection of concepts indexed on "Route" and "Travel", and possibly

other related concepts. We include synonyms to allow a broader coverage of the concepts in the FAQ database in order to increase the recall of matching operation (see the "Concept-FAQ table"). We can achieve a further improvement by including a concept hierarchy to map user queries into a collection of related concepts. The rest of the processing is decomposed into three steps.

Step 1. Mapping from the question space to the concept space

The natural language processing module returns a parse tree that contains a semantic class and its parameters. The LEAP parser returns a concept "Route" as a semantic class. We can use the semantic class "Route" to search the Concept-FAQ table, which can then be used to search the concept-FAQ table. The concept-FAQ table is the core data structure for the whole database. Every FAQ is assigned a FAQ-ID, which is uniquely distinguished from the others. An FAQ is made up of a few concepts that are in fact represented by certain terms such as "Route". Every (*Concept, FAQ ID, Weight*) record denotes that the FAQ is related to the concept with a correlation *Weight* factor, which is learned from a later analysis of the query log file. Every FAQ is related to one or more concepts and every concept is related to one or more FAQ's. Thus there is a many-to-many relationship between FAQ's and concepts. Using the concept-FAQ table, we can compute a correlation factor between a concept set $\Phi(\text{concept}_1, \text{concept}_2, \ldots \text{concept}_n)$ and a FAQ with ID of x as

$$\sum_{i=1}^{n} Weight(concept_i, \quad x).$$

Hence, given a concept set, it is straightforward to obtain the top n best-matched FAQ's.

Step 2. Mapping from the FAQ space to the template space

A template represents a class of *standard* questions. It corresponds to a semantic class in the LEAP module. Every template has one or more parameters with values. Once all the parameters in a template are assigned a value, a standard question is derived from this template. For example, "air flights to *" is a template representing a class of questions about the flight from or to a certain location. Here the wild card "*" denotes that there is a parameter in the template that can be assigned an arbitrary place name. If "Shanghai" is chosen, this template is transformed into a standard question "what are the air flights to Shanghai?". We cannot construct a template for every question since there are many similar questions. We solve this problem by choosing all similar questions and prepare a single template for them. This effectively compresses the FAQ set

Step 3. Mapping from the template space to the answer space

All answers for a template are previously stored in a separate answer table. This answer table is indexed by the parameter values of the template. When a matching is done, the best parameter is calculated and passed to the GUI component to be shown to the user. Every answer is made up of two parts: a URL and its description.

3.2 Query log mining

Our system is purely data-driven, where the most important information is derived from the query logs and the World Wide Web. In our current work, we address the following critical questions

a. How to find the frequently asked questions from a large amount of user questions? The challenge here is to address the time-variant nature of the questions, because some questions important today may become unimportant tomorrow.

b. How to find answers for a template automatically or semi-automatically?

c. How to determine the weights between concepts and FAQ's, and between FAQ's and templates?

Our system for query log mining consists of statistical query co-occurrence analysis, clustering and classification analyses. In our experience, we have found these tools to offer the search engine index maintainer very effective help in keeping the FAQ and template databases up to date.

4 User Interface

Fig. 4 and 5 show the process in which users ask natural language queries. Much research has shown that natural language is more powerful in expressing user's intention than keywords. Users usually have difficulty in formulating good searching keywords even when they have clear ideas about what they need [12].

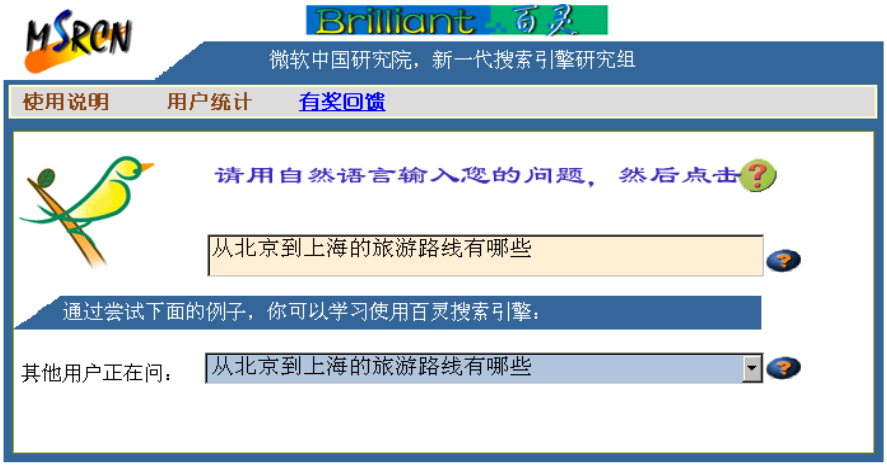

Fig. 4. Using natural language to ask queries in Brilliant ™ search engine.

Research has established that natural language is a powerful tool for expressing the user intention [15], where the most important parts of a query are known as *core phrases*. For example, two typical queries in the traveling domain search are:

How many routes are there from (Beijing) to (Shanghai)?

Please tell me about famous sights to visit in (Beijing)?

In these queries, the underlined words are core phrases, the parenthesized words are keywords, and the remaining words are redundant words. However, sometimes it is impossible to identify users' intention from the original query alone. In this case we can select all possibly relevant concept templates and ask the user to confirm.

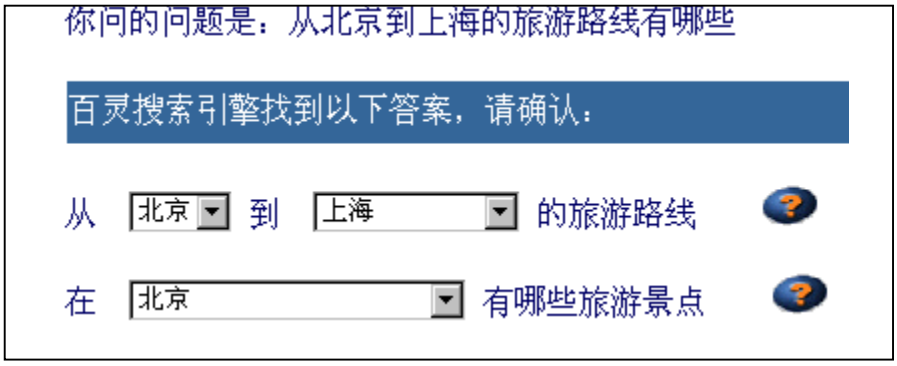

Fig. 5. Use Parameterized search results to asked back

In our user interface design, we clustered related concepts according to their similarity and treated the different parts of the result as parameters. For example, as

shown in Fig. 5, the two similar search results ("famous sites in Beijing", and "famous sites in Shanghai") can be combined into one group, where "Beijing" and "Shanghai" can be treated as parameters. The parameterized result can help focus users' attention on the core phrases, which in this case corresponds to famous sites.

5 Conclusions

Search engine technology has gone through several evolutions and has finally reached the point where Artificial Intelligence can offer tremendous help. We have outlined a new architecture of a search engine based on robust parsing of input queries and data mining of query log data. The advantage of the search engine include high precision and efficiency without compromising coverage. In the future, we plan to carry out empirical studies of this type of search engine with data from realistic Internet searches.

References

1. J. Allen, "Natural language understanding". The Benjamin-Cummings Publishing Company, Inc. 1995
2. John B. Best, Cognitive psychology. West Publishing Company, 1995, 4th Edition.
3. Eric Brewer, "Delivering high availability for Inktomi search engines". *Proceedings of ACM SIGMOD international conference on Management of data*, 1998, pp. 538.
4. Cho J., Garcia-Molina H., Page L., "Efficient Crawling Through URL Ordering", *The 7th International WWW Conference* (WWW 98). Brisbane, Australia, April 14-18, 1998.
5. P. R. Cohen, H. J. Levesque. "Intention is choice with commitment". *Artificial Intelligence*, 42, pp. 213-261, 1990.
6. Susan Dumais, John Platt, David Heckerman et al., "Inductive learning algorithms and representations for text categorization". *Proceedings of the 1998 ACM 7th international conference on Information and knowledge management* , 1998, pp. 148-55.
7. Eric Horvitz, "Principles of mixed-initiative user interfaces". *Proceeding of the CHI 99 conference on Human factors in computing systems: the CHI is the limit*, 1999, pp. 159-166.
8. F. Jelinek, J.D. Lafferty, and R.L. Mercer, "Basic Methods of Probabilistic Context Free Grammars", In *Proc. Laface and R. De Mori (eds). Speech Recognition and Understanding--Recent Advances*, Springer-Verlag Berlin Herdelberg, NATO ASI Series, Vol. F75, 1992, 345-360

9. A. Lavie, “GLR*: A Robust Grammar-Focused Parser for Spontaneously Spoken Language”. Carnegie Mellon University Ph.D.'s dissertation. 1996.
10. Jakob Nielsen. “User interface directions for the Web”. *Commun. ACM* 42(1), pp. 65-72, 1999
11. Page L. and Brin S., “The Anatomy of a Search Engine.” *The 7th International WWW Conference* (WWW 98). Brisbane, Australia, April 14-18, 1998.
12. A. Pollock and A. Hockley, “What’s wrong with Internet Searching”. D-Lib Magazine. http://www.dlib.org/dlib/march97/bt/03pollock.html
13. W. Ward, “Understanding Spontaneous Speech: The Phoenix System”. In *Proceedings of IEEE International Conference on Acousitcs, Speech and Signal Processing (ICASSP’91)*, 1991, pp 365-367.
14. Y. Wang. "A robust parser for spoken language understanding". In *Proc. of 6th European conference on speech communication and technology (Eurospeech99)*. Budapest, Hungary, Sept. 1999, pp. Vol.5, 2055-2058.
15. Yen-ju Yang, Lee-feng Chien, Lin-shan Lee. “Speaker intention modeling for large vocabulary mandarin spoken dialogues”. *Proc. of Fourth International Conference on Spoken Language*, 1996.
16. Oren Zamir and Oren Etzioni. “Web document clustering: a feasibility demonstration”. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998, pp46-54.