Query Routing for Web Search Engines: Architecture and Experiments

Atsushi Sugiura* and Oren Etzioni**

Human Media Research Laboratories, NEC Corporation*
Department of Computer Science and Engineering, University of Washington**

Abstract

General-purpose search engines such as AltaVista and Lycos are notorious for returning irrelevant results in response to user queries. Consequently, thousands of specialized, topic-specific search engines (from VacationSpot.com to KidsHealth.org) have proliferated on the Web. Typically, topic-specific engines return far better results for "on topic" queries as compared with standard Web search engines. However, it is difficult for the casual user to identify the appropriate specialized engine for any given search. It is more natural for a user to issue queries at a particular Web site, and have these queries automatically *routed* to the appropriate search engine(s).

This paper describes an automatic query routing system called *Q-Pilot*. Q-Pilot has an off-line component that creates an approximate model of each specialized search engine's topic. On line, Q-Pilot attempts to dynamically route each user query to the appropriate specialized search engines. In our experiments, Q-Pilot was able to identify the appropriate query category 70% of the time. In addition, Q-pilot picked the best search engine for the query, as one of the top three picks out of its repository of 144 engines, about 40% of the time. This paper reports on Q-pilot's architecture, the query expansion and clustering algorithms it relies on, and the results of our preliminary experiments.

Keywords: Web search, query routing, query expansion, search engines.

1 Introduction

Search engines, such as Yahoo! [21] and AltaVista [14], are useful for finding information on the World Wide Web. However, these *general-purpose* search engines are subject to low precision and/or low coverage. Manually-generated directories such as Yahoo! provide high-quality references, but cannot keep up with the Web's explosive growth. Although crawler-based search engines, like AltaVista, cover a larger fraction of the Web, their automatic indexing mechanisms often cause search results to be imprecise. It is thus difficult for a single search engine to offer both high coverage *and* high precision. This problem is exacerbated by the growth in Web size and by the increasing number of naive users of the Web who typically issues short (often, single word) queries to search engines.

The recent growth in both the number and variety of specialized *topic-specific* search engines, from VacationSpot.com [20] to KidsHealth.org [18] or epicurious.com [16], suggests a possible approach to this problem: search topic-specific engines. Topic-specific search engines often return higher-quality references than broad, general-purpose search engines for several reasons. First, specialized engines are often a front-end to a database of authoritative information that search engine spiders, which index the Web's HTML pages, cannot access. Second, specialized search engines often reflect the efforts of organizations, communities, or individual fanatics that are committed to providing and updating high-quality information. Third, because of their narrow focus and smaller size, word-sense ambiguities and other linguistic obstacles to high-precision search are ameliorated.

The main stumbling block for a user who wants to utilize topic-specific search engines is: how do I find the appropriate specialized engine for any given query? Search.com offers a directory of specialized search engines, but it is up to the user to navigate the directory and choose the appropriate engine. A search engine of search engines is required. To build such an engine two questions have to be addressed: How can we build an index of high-quality, specialized search engines? And, given a query and a set of engines, how do we find the best engine for that query? In this paper, we focus on the latter problem, which is often referred to as the *query routing* problem.

Although many query routing systems [1][3][6] have been developed, few of them are aimed at the topic-specific search engines provided on the Web. To automate the query routing process, conventional query-routing systems need to access the complete internal database associated with each engine. Yet most of the specialized search engines on the Web, do not permit such access to their internal databases.

This paper presents a novel query routing method, called *topic-centric* query routing, which compensates for lack of unfettered access to search engine databases by using two key techniques:

- **Neighborhood-based topic identification**: a technique for collecting the abstract topic terms relevant to a search engine from existing Web documents.
- **Query expansion**: a technique for obtaining the terms relevant to a query. For the purpose of topic-centric query routing, it is used mainly for evaluating the relevance of a query to the identified topic terms of search engines.

While conventional query routing techniques compare a user query with all the documents or terms contained in search engines' databases, our method compares a query with a relatively small number of abstract topic terms. In this sense, we call the proposed method *topic-centric* query routing. It is implemented in a query routing system called *Q-Pilot*.

The rest of this paper first describes related work to clarify the position of our research and then describe the topic-centric query routing method and Q-Pilot in detail. It also presents the results of experiments using Q-Pilot.

2 Related Work

Conventional query routing systems and services (some of them are currently available on the Web) can be classified into three groups.

Manual query routing services. Some query routing services has recently become available on the Web. However, each has some aspect of query routing performed manually by the user. AllSearchEngines.com [13], SEARCH.COM [19], InvisibleWeb.com [17] and The Search Broker [7] provide a categorized list of specialized search engines, but these services basically require the users themselves to choose engines from the list. Although they provide keyword search interfaces to find desired search engines, the terms that can be accepted as the keywords are limited to the abstract category names (such as "sports"). The users are required to map from their specific queries (such as "Michael Jordan") to the related categories in their mind.

Automated query routing systems based on centroids. Some systems perform automated query routing. A centroid-based technique is widely used by these kinds of systems. Namely, it generates "centroids" (summaries) of databases, each of which typically consists of a *complete* list of terms in that database and their frequencies, and decides which databases are relevant to a user query by comparing the query with each centroid. CORI [1] is a centroid-based system. GlOSS [3] is also based on the same kind of the idea, although it does not explicitly generate the centroid. STARTS [2] and WHOIS++ [10] propose standard architectures and protocols for the distributed information retrieval using centroids provided by information sources.

An advantage of the centroid-based technique is to be able to handle a wide variety of search keywords by using the large number of the terms obtained from databases. However, this technique cannot be applied to most of the topic-specific search engines provided on the Web because of the restricted access to their internal databases, as we mentioned in the Introduction section.

Automated query routing systems without centroids. There are some automated query routing systems that do not generate centroids. However, these systems have strict limitations on acceptable search keywords. Query routing in Discorver [8] relies on short texts, associated with WAIS databases, to explain the contents of databases given by service providers. Discover can operate only when some of the search keywords are contained in the short texts. Although Discover helps users refine their queries so that it can select topic-specific search engines, this effort is insufficient for handling a wide variety of search keywords. Profusion [4] routes queries in thirteen predefined categories to six search engines. It posts sample queries from each category to the search engines and examines which engine is good for that category by checking relevance of the returned documents. Profusion has a dictionary to determine which categories the given user queries are relevant to. Since, however, this dictionary is created by looking at newsgroups' names (a term "movie" can be categorized into a recreation category from "rec.movie.reviews"), as a result Profusion can accept only limited types of queries.

One exceptional system that cannot be classified into any of these three groups is Ask Jeeves [15], which performs automated routing of queries to a limited set of Web sites that contain "answers" to user "questions." Since Ask Jeeves is a proprietary commercial system, little is known about its internal routing algorithm, its degree of automation, or its ability to scale.

3 Q-Pilot: A Topic-centric Query Routing System

3.1 Overview

Q-Pilot is an automated query routing system, which does not generate centroids, composed of an off-line pre-processing component and an on-line interface (Figure 1). Off-line, Q-Pilot takes as input a set of search engines' URLs and creates, for each engine, an approximate textual model of that engine's content or scope. We experimented with several methods for approximating an engine's scope and found that the *Neighborhood-based topic identification* technique, which collects terms representing the scope from Web pages in the "neighborhood" of the search engine's home page, is surprisingly effective. Q-Pilot stores the collected terms and their frequency into the search engine selection index.

On-line, Q-Pilot takes a user query as input, applies a novel query expansion technique to the query and then clusters the output of query expansion to suggest multiple topics that the user may be interested in investigating. Each topic is associated with a set of search engines, for the query to be routed to, and a phrase that characterizes the topic (Figure 2b). For example, for the query "Python" Q-Pilot enables the user to choose between movie-related search engines under the heading "movie - monty python" and software-oriented resources under the headings "objected-oriented programming in python" and "jpython - python in Java".

An important key point in the Q-Pilot design is to use the Neighborhood-based identification of search engines' topics in combination with query expansion. The Neighborhood-based method does not collect terms relevant to search engine's topics from search engine's internal databases, but collects them from the limited "neighborhood" Web pages. Therefore, only a small number of abstract terms (some of them representing the topics of a search engine) can be obtained. On the other hand, user queries are likely to be short (only two or three search keywords, usually), and no topic term is specified in many cases. Query expansion bridges a gap between the short query and the small number of terms about search engines' topics. The query expansion technique used in Q-Pilot is specially tailored for the query-routing purpose to identify the topics implicit in the query. Thereby Q-Pilot can make the topic-level mapping from queries to search engines.

Another important benefit of the query expansion process is the ability to automatically obtain terms relevant to a query from the Web, which is an immense corpus. This allows Q-Pilot to identify topics of any kinds of queries without having to maintain a massive dictionary or database of terms in a wide range of fields.

Note that Q-Pilot obtains all the information necessary in query routing from the Web. That is, the topics of search engines are identified using the existing neighborhood Web documents, and the terms relevant to the query are also obtained from Web documents. In a sense, Q-Pilot is an intelligent agent that uses the Web as its knowledge base and autonomously learns what it does not know.

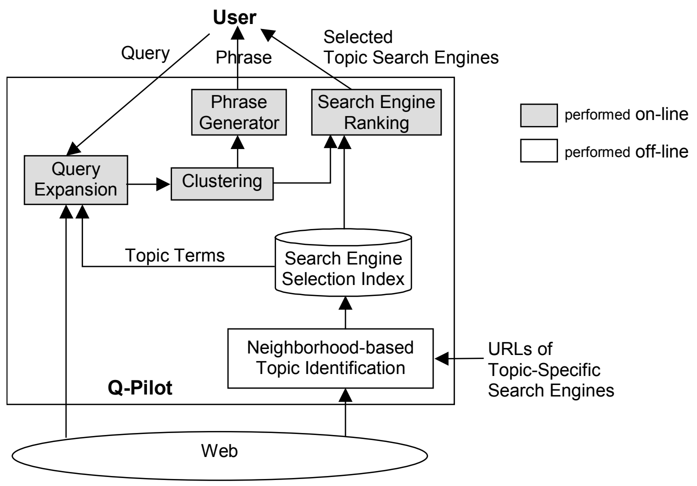
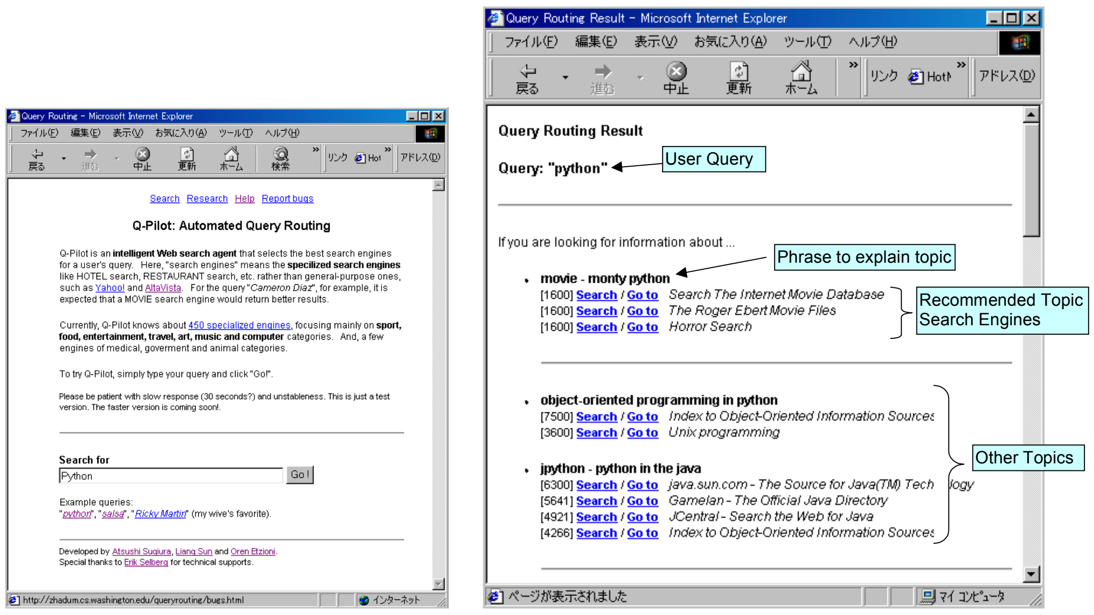


Figure 1: System architecture of Q-Pilot.

3.2 User Interface

Q-Pilot provides a simple keyword search interface (Figure 2a) and outputs the query routing result for the given query (Figure 2b). As shown in Figure 2b, when the query is related to multiple topics, Q-Pilot selects search engines for each different topic and gives phrases explaining the topics. The user can choose the search engine to be queried by clicking a “Search” link or a “Go to” link. With the “Search” link, the user's original query is forwarded to the corresponding topic-specific search engine and the search results from that engine are displayed. The “Go to” link leads the user to a search form page of the topic-specific search engine, where the user has to submit the query again.

Some query routing systems forward the query directly to the selected search engines, skipping the intermediate step shown in Figure 2b, and subsequently merge the search results into a unified format. The current version of Q-Pilot, however, does not perform such merging.


Query Routing - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 進む 中止 更新 ホーム 検索 リンク Hot アドレス(D)

Search Research Help Report bugs

Q-Pilot: Automated Query Routing

Q-Pilot is an **intelligent Web search agent** that selects the best search engines for a user's query. Here, "search engines" means the **specilized search engines** like HOTEL search, RESTAURANT search, etc. rather than general-purpose ones, such as Yahoo! and AltaVista. For the query "*Cameron Diaz*", for example, it is expected that a MOVIE search engine would return better results.

Currently, Q-Pilot knows about 450 specialized engines, focusing mainly on **sport, food, entertainment, travel, art, music and computer** categories. And, a few engines of medical, goverment and animal categories.

To try Q-Pilot, simply type your query and click "Go!".

Please be patient with slow response (30 seconds?) and unstableness. This is just a test version. The faster version is coming soon!.

Search for
Python
Go !

Example queries:
"*python*", "*salsa*", "*Ricky Martin*" (my wive's favorite).

Developed by Atsushi Sugiura, Liang Sun and Oren Etzioni.
Special thanks to Erik Selberg for technical supports.

http://zhadum.cs.washington.edu/queryrouting/bugs.html インターネット

Query Routing Result - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 進む 中止 更新 ホーム リンク Hot アドレス(D)
User Query
Phrase to explain topic
Recommended Topic Search Engines
Other Topics

Query Routing Result

Query: "python"

If you are looking for information about ...

- **movie - monty python**
  [1600] Search / Go to *Search The Internet Movie Database*
  [1600] Search / Go to *The Roger Ebert Movie Files*
  [1600] Search / Go to *Horror Search*

- **object-oriented programming in python**
  [7500] Search / Go to *Index to Object-Oriented Information Sources*
  [3600] Search / Go to *Unix programming*

- **jpython - python in the java**
  [6300] Search / Go to *java.sun.com - The Source for Java(TM) Technology*
  [5641] Search / Go to *Gamelan - The Official Java Directory*
  [4921] Search / Go to *JCentral - Search the Web for Java*
  [4266] Search / Go to *Index to Object-Oriented Information Sources*

ページが表示されました マイ コンピュータ

(a) A query form. (b) An example of query routing results.
Figure 2b: Screen snapshots of Q-Pilot.

3.3 Neighborhood-based identification of Search Engine’s Topics

We propose two methods for Neighborhood-based topic identification, which collect terms relevant to a search engine from existing, static Web documents:

- **The front-page method:** Every search engine has a page providing a query interface (we call this page a *front page*), and the front page usually contains terms explaining a topic of that search engine. In the front-page method, all terms in the front page[1] and their frequencies are registered to a search engine selection index.
- **The back-link method:** Web pages that have links pointing to a search engine's front page (we call these pages *back-link pages*) often contain good explanations of that search engine. The back-link method first finds multiple back-link pages for a target search engine e_i,[2] next extracts from all the back-link pages only the terms that are in the lines of the links to e_i, and stores into the search engine selection index all the extracted terms and their document frequencies.

We call high-frequency terms, which appear in the search engine selection index, *topic terms*. Specifically, a set of topic terms $TOPIC_i$ for the search engine e_i is defined as follows:

$$TOPIC_i = \{w_{ij} \mid f_{ij} > f_{max} * 0.8\}$$

where w_{ij} ($1<=j<=m$) is a term in the index for e_i, f_{ij} is its frequency, and f_{max} is the highest frequency observed in the index. Figure 3 shows examples of terms obtained by the back-link method. The abstract general terms would usually be topic terms.

Computer security search engine (www.securityfocus.com)		Hotel search engine (hotel search under www.travelweb.com)	
security	1.00	*hotel*	1.00
bug	0.89	travel	0.57
bugtraq	0.44	airline	0.57
unix	0.39	reservation	0.43
exploit	0.28	restaurants	0.29
attack	0.28	map	0.29
password	0.22	book	0.29
mail	0.17	flights	0.29
holes	0.17	availability	0.29
nt	0.17	points	0.29

Figure 3: Terms collected by the back-link method.
(Italic fonts designate topic terms and frequencies are normalized)

For comparison in the experiments described later, we also implemented a method that collects terms from the search engine's database

- **The database sampling method:** A database sampling method obtains a part of a database and generates a kind of an incomplete centroid. That is, it submits training queries to the search engine and stores in the search engine selection index all the terms in the returned documents and frequencies of those terms. A similar method has been proposed by Xu [11].

Due to the nature of the sources used for the term collection, most of the terms collected by the front-page and back-link methods are abstract general terms like "hotel" and "travel." Relatively few specific terms, such as the proper noun "Hilton," are obtained. In contract, the database sampling method is quite successful at collecting specific terms.

[1] Stop words are excluded.

[2] Q-Pilot finds back-link pages by using the link search function of AltaVista, querying "*link:URL_of_engine*".

3.4 Query Expansion

Query expansion is a technique, widely used in information retrieval, for obtaining additional terms relevant to a given query (search keywords). It is usually used to help information searchers express their intentions more accurately and increase the precision of search results. In Q-Pilot, however, its main purpose is to evaluate the relevance of search keywords to the topic terms stored in the search engine selection index. Figure 4 shows a query expansion algorithm in Q-Pilot. Its general framework is as follows:

- **Getting relevant terms from the Web dynamically**
  Q-Pilot does not use any special dictionaries for query expansion, but it uses the Web (the existing Web documents) as the source of relevant terms. As shown in step 1 of Figure 4, it finds the Web documents relevant to the user query dynamically by submitting that query to a general Web search engine[3]. The relevant terms are extracted from those documents. Since there is an immense corpus on the Web, terms relevant to any kind of search keywords can be obtained, even peculiar proper nouns, technical terms, etc. In thesaurus-based query expansion [9], covering any terms of any fields is difficult.
- **Co-occurrence-based evaluation of term relevance**
  The mutual relevance of terms is evaluated on the basis of their co-occurrence in the documents. In steps 2 and 3, the co-occurrences of the search keywords and other terms are counted in 30 documents retrieved by the general search engine in step 1. That is, the system lists all distinct terms contained in 30 documents, and counts for each term the number of documents that contain both the search keyword and that term. To reduce the computational time, Q-Pilot handles a pair of a page title and a snippet in the search result as a single document and does not download the actual documents.
- **Using a pseudo-feedback technique**

  It is difficult to determine the term relevance from only the results of a single document search on the general search engine. Even closely relevant terms often have few co-occurrences in the 30 documents of the first search. Q-Pilot, therefore, re-evaluates such low co-occurrence terms: selecting terms to be re-evaluated from the first search results (steps 4 and 6), formulating new queries by adding the selected terms to the original query (steps 5 and 7), and performing the co-occurrence-based evaluation again for each formulated query (steps 8 and 9). Such automatic query refinement is called pseudo-feedback [5].

The pseudo-feedback process treats topic terms as follows. First, as shown in step 4, low co-occurrence topic terms in the first search results are selected for re-evaluation prior to other non-topic terms.

Steps 6 and 7 are also important for topic terms. In these steps, non-topic terms are added to the original user query for the pseudo-feedback. However, the main purpose of this is to get new topic terms that were not obtained through the first search rather than to re-evaluate the added non-topic terms. As found in earlier query expansion experiments, search results can be improved in many cases by using additional terms. The improved search results are more likely to contain good terms like topic terms.

The clustering in step 6 (explained in the next subsection) is expected to contribute effectively to finding new topic terms. Suppose, for example, that the non-topic terms “Monty,” “scripting,” and “language” are obtained by the first search when the user query is “Python.” The topic terms like “comedy” and ”programming” are more likely to be obtained in the pseudo-feedback process by clustering non-topic terms and querying “Python Monty” and “Python language scripting” independently rather than by querying “Python Monty language scripting” without clustering.

Both the topic terms and the non-topic terms obtained through query expansion are used in the ranking of topic-specific search engines and the extraction of phrases to explain topics of the selected

[3] Q-Pilot currently uses MetaCrawler, a meta-search engine, in Step 1 so that it can collect a variety of terms from many information sources. Step 8 uses AltaVista because of its short response time.

search engines.

In the pseudo-feedback process, seven queries are posted to the general search engine in parallel and the total processing time for query expansion is about four seconds currently.

1. Get a document set D_0 relevant to a user query Q_0, where search keywords are $w_{01}, \ldots, w_{0n}$, by sending Q_0 to a general search engine.
2. Count co-occurrences of search keywords and other terms in the document set D_0.
3. Let WH_0 and WL_0 be a set of terms whose co-occurrences exceed a certain threshold and a set of the other terms, respectively. WH_0 is considered relevant to the query Q_0 and will be a part of the query expansion result.
4. Pick up at most four topic terms w_{t1}-w_{t4} from WL_0.
5. Formulate four queries QT_1-QT_4 by combining w_{t1}-w_{t4} with Q_0 (for example, QT_1="$w_{01} \ldots w_{0n}\ w_{t1}$").
6. Clustering all terms in D_0 to at most three clusters: $W_1=\{w_{11}, \ldots, w_{1m}\}$, $W_2=\{w_{21}, \ldots, w_{2k}\}$ and $W_3=\{w_{31}, \ldots, w_{3j}\}$.
7. Formulate three queries Q_1-Q_3 by combining W_1-W_3 with Q_0 (for example, Q_1="$w_{01} \ldots w_{0n}\ w_{11} \ldots w_{1m}$").
8. Get document sets DT_1-DT_4 and D_1-D_3 by sending QT_1-QT_4 and Q_1-Q_3 independently to a general search engine.
9. Count co-occurrences in DT_1-DT_4 and D_1-D_3. Sets of high co-occurrence terms WTH_1-WTH_4 and WH_1-WH_3, as well as WH_0 in step 3, are query expansion results.

Figure 4: Query expansion procedure.

3.5 Clustering

As shown in Figure 4, the terms in the document set D_0 retrieved in response to the original user query Q_0 are clustered for the preparation of the pseudo-feedback process (step 6). Since Q-Pilot spends a lot of time (about four seconds) on query expansion, it uses a simple, ad hoc method for the clustering in order to reduce the total computational time. The clustering algorithm in Q-Pilot generates at most only three clusters that are mutually exclusive, such as one about the comedy group Monty Python, one about the programming language Python, and one about the snake python. The algorithm is as follows:

1. Pick up the term w_{max1} with the highest co-occurrence in the document set D_0 obtained in step 1 of the query expansion algorithm in Figure 4. Let D_{01} be a set of documents containing w_{max1}.
2. Pick up the highest co-occurrence term w_{max2} in a set of documents not containing w_{max1}. Let D_{02} be a set of documents containing w_{max2} and not containing w_{max1}.
3. Let D_{03} be a set of the other documents.
4. Terms that appear in D_{01}, D_{02}, and D_{03} would be the clusters of terms W_1, W_2, and W_3 in step 6 of Figure 4, respectively.

For the user query "python", for example, "Monty" and "programming" would typically be w_{max1} and w_{max2}, respectively, and "snake" would be contained in D_{03}.

After the query expansion process, there are at most eight clusters of terms: WH_0 (step 3), WTH_1-WTH_4, and WH_1-WH_3 (step 9). Since, however, different clusters are often related to the same topic, Q-Pilot merges them to eliminate the duplicates. Basically, it merges clusters that contain the same topic term. It also merges clusters that have many common terms. In future work, we plan to compare the performance of this ad hoc method with that of standard, linear-time clustering algorithms such as buckshot or fractionation.

3.6 Ranking Topic-specific search engines

Q-Pilot calculates the *goodness* of each topic-specific search engine for the given query by comparing the terms obtained through query expansion with the terms stored in the search engine selection index. Using the calculated *goodness*, it generates a ranked list of the search engines and selects the top three as query routing results. If there are multiple clusters of query expansion terms, the search engines are ranked for each cluster. Therefore, *3*n* search engines are selected for *n* clusters.

The *goodness* of a search engine *e* for a given set $W=\{w1, w2, ...\}$ of query expansion terms is calculated as follows:

$$goodness(e,W) = \underset{w_i \in W}{\qquad} f_i * c_i$$

where c_i is the number of co-occurrences of w_i counted in query expansion process and f_i is the frequency of term w_i in the search engine selection index for *e* (f_i=0 if there is no w_i in the index). Note that not only topic terms but also non-topic terms are used in the calculation of *goodness*.

3.7 Extracting Phrases to Explain Topics

As shown in Figure 2b, Q-Pilot gives a phrase to explain a topic representing the content of each cluster. The phrase is extracted from the document sets *D0-D3* and *DT1-DT4* obtained in the query expansion process. In extracting a phrase for a cluster of terms $W=\{w1, w2, ...\}$, Q-Pilot first finds in those document sets all phrases (sequence of terms), each of which contains only w_i□W, prepositions, and articles. Using some heuristics, it then selects the best one. Basically, a phrase that contains topic terms and many high co-occurrence terms is selected.

For example, if *W*={‘python’, ‘object’, ‘programming’, ‘scripts’, ‘oriented’} is obtained by query expansion, “object oriented programming with python” and “scripts of python” would be extracted and the first phrase that has the topic term “programming” would be selected.

STC [12], a linear-time document clustering algorithm, which also extracts phrases that represent document clusters from the classified documents. In STC, cluster choice is a function of phrase length and the number of documents in a cluster. However, the phrase extraction process of Q-Pilot is more sensitive to the relevance of the phrases, search keywords, and topic terms, and it would be expected that Q-Pilot extract more appropriate phrases. A detailed comparison between STC and Q-Pilot’s clustering algorithm is a direction for future work.

4 Experiments

To evaluate the proposed topic-centric query routing method, we conducted several experiments. The questions here are:

- How do three methods for topic identification (the front-page, the back-link and database the sampling methods) affect the routing accuracy?
- To what extent do query expansion and clustering improve accuracy?
- How does the number of routing-target topic-specific search engines impact accuracy?
- Is the topic-centric method practical for routing queries to topic-specific search engines on the Web?

In the rest of this section, we will describe a data set used in the experiments, the accuracy measure, an experimental setup and results.

4.1 Experimental Data and Performance Measure

As the experimental data, we used a query log given by actual users to The Search Broker [7], which is a query routing system using about 400 topic-specific search engines classified into 25 categories, such as Computer, Travel, and Entertainment.

The Search Broker has a table, created manually, mapping a specific topic term t_i to a specific search engine e_i and it can select the engine only when the user describes the topic term at the top of the submitted list of search keywords. Therefore, every query in the Search Broker's query log $Q=\{q_1, q_2, ..., q_N\}$ has the following format:

q_i = "t_i w_{i1} w_{i2} w_{im}"

In the experiments, we fed to Q-Pilot the query qb_i = "w_{i1} w_{i2} w_{im}" (q_i minus the topic term t_i) and examined whether Q-Pilot could select the search engine e_i corresponding to t_i. More precisely, since Q-Pilot returns a ranked list of search engines, we examined whether e_i was contained in the top d elements in that ranking. Given a set of test queries Q, the accuracy of query routing is calculated as follows:

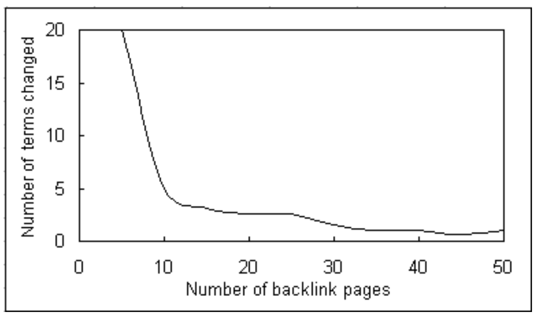
$$Accuracy(Q,d) = \underset{q_i \in Q}{\quad} F(R(d,qb_i),e_i)/|Q|$$

where $R(d, qb_i)$ is a set of search engines that Q-Pilot ranks within the top d for qb_i, and F returns 1 if e_i□R and otherwise returns 0.

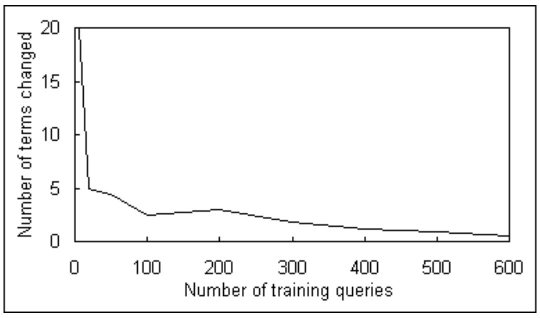
Note that this measure of accuracy is conservative for two reasons. First, human users of the Search Broker can make inappropriate engine choices, but when Q-pilot fails to match these, its accuracy measure is penalized. Second, there may be more than one appropriate engine choice per query. Again, when Q-pilot fails to match the choice made by the human --- its accuracy is penalized.

4.2 Experimental Setup

Before the measurement of accuracy, it is necessary to create the search engine selection indices for each topic identification method. To create the indices, we used 50 back-link pages for the back-link method and used 600 training queries for the database sampling method. This is because, as shown in Figures 5a and 5b, the most important parts of two indices (the 20 terms with the highest frequencies) are rarely changed after the learning by the 50 back-link pages and the learning of the 600 training queries. The 600 training queries were randomly selected from the Search Broker's query log and they were distinct from the set of test queries used in the later experiments.



(a) Back-link method



(b) Database sampling method

Figure 5: Changes of terms with the 20 highest frequencies when creating the search engine selection index.

4.3 Experimental Results

4.3.1 Comparison of three topic identification methods and evaluation of the effectiveness of query expansion

We first measured, for each of the three topic identification methods, the *Accuracy* in routing 150 test queries to 27 topic-specific search engines in the Computer and Travel categories of the Search Broker. There are originally 52 engines of those two categories in the Search Broker. However, only 27 of them (15 in the Computer category and 12 in the Travel category) allow a Web page collection robot to access their databases. So, we used the 27 engines to which the database sampling method is applicable.

Table 1 lists the results obtained when *Accuracy* was calculated with *d*=3 (the correct engine is within the top three). The Simple_QR denotes the results obtained without query expansion and clustering. The QR+QE denotes the results obtained with query expansion (but without clustering).

In the Simple_QR, the database sampling method performed best because of its ability to collect terms relevant to search engines. As we mentioned, the neighborhood-based methods (the front-page and the back-link methods) can collect only a small number of abstract terms. Therefore, in many cases, the search keywords did not match any terms in the search engine selection index and could not be mapped to any search engine.

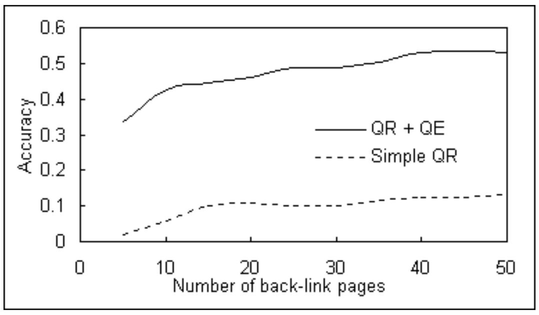
Query expansion improved the performance of all three topic identification methods, especially, that of the back-link method. Its accuracy was improved by about 40% to almost the same level as that of the database sampling method. This result shows that the proposed query expansion technique can find topics of given queries well enough to compensate for the lack of term collection capability of the back-link method.

The poor performance of the front-page method is due to the difficulty of identifying topic terms by using only the front page. That is, there can be no big difference in term frequencies when the terms are collected from only that page. When multiple back-link pages are used, however, the back-link method can identify the topic terms more precisely.

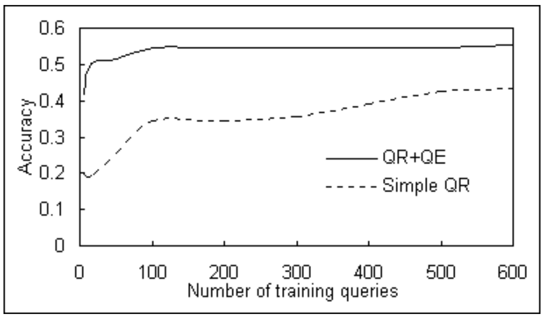
Table 1: *Accuracy* (%) by three topic identification methods (*d*=3, 27 topic-specific search engines, 150 test queries).

	Simple QR	QR + QE
Front-page method	9.2	32.2
Back-link method	13.5	54.2
Database sampling method	43.2	55.4

Learning curves for the back-link method and the database sampling method are shown in Figure 6. At 50 back-link pages and 600 training queries, the learning curves of both are saturated in terms of QR+QE. These results show that 50 back-link pages and 600 training queries are reasonable sizes as the learning data sets needed to create search engine selection indices.



(a) Back-link method



(b) Database sampling method

Figure 6: Learning curves
(*d*=3, 27 topic-specific search engines, 150 test queries).

4.3.2 Effectiveness of Clustering

When performing both query expansion and clustering on the 150 test queries, an average of 1.6

clusters of terms was generated. Since three engines are selected for each cluster, a total of 4.8 engines are selected. To be fair, we compared the accuracy in selecting the top five engines without clustering (*d*=5) with the accuracy in selecting the top three engines for each cluster with clustering (*d*=3). As shown in Table 2, the performances were not improved by clustering in the back-link and the database sampling methods.

Although the clustering did not greatly improve of the query routing performance, it is important from a point of view of the user interface. Suppose, for example, that the system displays the search engines about programming languages together with those about comedies without clustering when the user query is "Python". A user who is searching for the comedy group Monty Python and does not know about the programming language Python would be likely to mistrust the performance of the system. Clustering and giving phrases to explain the clusters would be expected to make the user more confident of the query routing results.

Table 2: *Accuracy* (%) when using clustering.
(27 topic-specific search engines, 150 test queries)

	QR + QE (*d*=5)	QR + QE + Clustering (*d*=3)
Front-page method	32.2	46.1
Back-link method	63.5	61.5
Database sampling method	66.9	63.8

4.3.3 Scalability against the number of topic-specific search engines

We measured the performance in routing 800 test queries to 144 topic-specific search engines belonging to nine categories: Animal, Entertainment, Computer, Food, Music, Medical, US Government, Travel and Sport. The database sampling method could not be applied to many of the 144 search engines in our experimental set because these engines forbid robots. In addition, we did not believe that the front-page method would perform well. As a result, our experiments below relied exclusively on the back-link method.

The results are shown in Figure 7. *Accuracy* naturally decreased as the number of search engines increased, but it was still high, about 40%, even on 144 engines. Also, It was almost constant from 90 engines. One of the reasons for this high accuracy is that the nine categories of the 144 engines are not closely related to each other conceptually. In such a set of routing-target search engines, it would be expected that the topic-centric query routing method could operate well even when routing to the larger number of topic-specific search engines.

This experiment, using 144 topic-specific search engines in nine categories, also demonstrated that the proposed query expansion technique that obtains from the Web terms relevant to queries could operate well for a wide variety of queries.

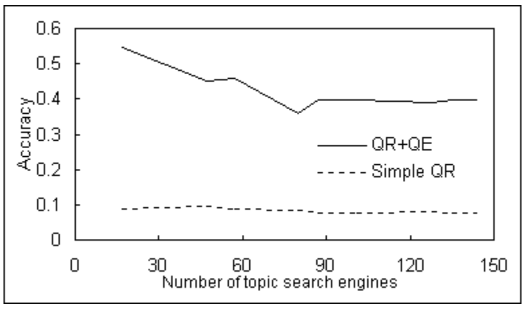


Figure 7: Performance for 144 engines in back-link method
(*d*=3, 144 topic-specific search engines in 9 categories, 800 test queries)

4.3.4 Possibility of Practical Use

The most important aspect with regard to practical use of the proposed topic-centric query routing method is of course its accuracy. We consider that, as an automated query routing method, the performance of Q-Pilot is good as demonstrated by the experiments conducted in the Section 4.3.1-4.3.3. However, about 40% accuracy on the 144 search engines might not be high enough for practical Web services. But, even when the wrong engines were selected, most of them were in the same category of the correct engines and were somewhat relevant to the queries. This is supported by other experimental results shown in Figure 8 and Table 3.

Figure 8 shows the category selection accuracy when routing the 800 queries to the 144 topic-specific search engines. The category selection accuracy was calculated for the search engines ranked within the top three by Q-Pilot and it denotes the percentage of engines that were in the correct engines' category. As shown in Figure 8, the category selection accuracy was very high: about 70% when using nine categories.

Table 3 shows the percentage of that topic-specific search engines ranked within the top three by Q-Pilot actually possess the data records and/or documents relevant to the query. We submitted *qbi* (excluded a topic term from the original query *qi*) to the selected search engines and examined whether those search engines returned at least one reference. This experiment was conducted on the 27 routing-target engines used in the experiments presented in Sections 4.3.1 and 4.3.2. As shown in Table 3, about 80% of the engines, selected when the back-link method was used, had information relevant to the queries.

It is important to note that "accuracy" in this paper is defined by the level to which variants of Q-pilot match the judgements of human users as captured in the Search Broker logs. However, in Table 3 we use a different measure of accuracy (did the engine return results in response to the query?) with higher accuracy figures. Probably, the measurement in Table 3 is overly liberal -- just because an engine returned some results doesn't mean that they are actually relevant to the query, so the 'true' accuracy is probably somewhere in the middle between Table 3 and Table 1.

Although these results were encouraging, the query routing accuracy depends on the set of topic-specific search engines. If the system would know higher-quality topic-specific search engines, the percentages listed in Table 3 could be better. As for the scalability, if we carefully choose routing-target topic-specific search engines that are independent of each other, both the engine selection accuracy and the category selection accuracy could be still high even when the number of the topic-specific search engines increases. So, in the practical use of the proposed method, how we choose a set of routing-target topic-specific search engines is critical.

Another important aspect is the response time. Current response time of Q-Pilot is 6-7 seconds, not short enough for practical use. The main reason of this long response time is its query expansion process referring the general search engine that can be used only through the Web. If we use a crawler and generate in local storage the index of the Web documents necessary for query expansion, the response time could be shorter. Further reduction of the response time would be possible by performing query expansion in advance for keywords frequently specified by users and making a cache of the query expansion results.

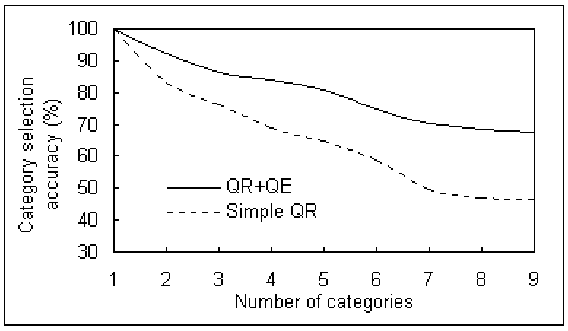


Figure 8: Category selection accuracy
(*d*=3, 144 topic-specific search engines in 9 categories, 800 test queries).

Table 3: Percentage of search engines actually having data records relevant to queries (d=3, 27 topic-specific search engines, 150 test queries).

	QR + QE
Front-page method	68.5
Back-link method	78.6
Database sampling method	83.5[4]

5 Conclusion and Future Work

This paper presented an automated query routing system Q-Pilot, which routes queries to topic-specific search engines available only through the World Wide Web. Q-Pilot learns topics of the search engines from the existing Web documents and identifies topics of given queries dynamically by query expansion.

Our preliminary experimental results show that the combination of the back-link method and the query expansion technique can yield performance that matches that of conventional query routing techniques, which are based on analyzing the contents of internal databases. We also found that Q-Pilot performs well even when routing queries to the large number (144) of topic-specific search engines. Finally, we showed that the proposed query expansion technique correctly identify the topics of a wide variety of queries. Our results offer a baseline for future research on query routing on the Web.

One direction for future research is the further improvement of the accuracy of query routing. One way to do this is to use user feedback given through the operations of the users who select their intended search engines on the query routing results like those in Figure 2b. It is expected that the search engine selection index can be optimized by adjusting the weights of the terms contained in the search keywords according to the engines that the user selected.

Another way to improve the accuracy is to rank topic-specific search engines by using collocations. In the case of a query "white paper of ...", for example, the current Q-Pilot might give a top rank to the White House search engine, although it should be ranked at the top only when both "white" and "house" are adjacent in a query. Such cases could be avoided by using collocation information when ranking search engines.

Several more experiments should be carried out. The most important one is to examine the performance of Q-Pilot on a substantially larger number of topic-specific search engines. Although the current version of Q-Pilot handles only 144 engines in 9 categories, far more engines are needed to cover all categories and topics. We need to verify that our query routing architecture scales to several thousands of engines, and identify methods of enhancing its accuracy. Also, as mentioned, we should compare the performance of our clustering and phrase extraction methods with that of the standard clustering algorithms and STC algorithm.

Acknowledgements

The authors express their appreciation to Erik Selberg and Liang Sun, who helped develop the system, and to Udi Manber, who kindly provided the experimental data from the Search Broker query log. Thanks are due to Dan Weld and Cody Kwok for discussions on related topics. This research was funded in part by Office of Naval Research grant 98-1-0177, and by National Science Foundation grants IRI-9357772 and DL-9874759.

References

[4] The index created by the database sampling method consists of terms that the search engine's database actually contains. Therefore, in the Simple_QR, this value will be 100%. In the QR+QE, however, terms obtained through query expansion match the terms in the index and it is not 100%.

[1] Callan J.P. , Lu Z., and Croft W.B., “Searching Distributed Collections with Inference Networks,” Proceedings of SIGIR’95, pp. 21-29, 1995.

[2] Gravano L., Chang K., Garcia-Molina H., Lagoze C. and Paepcke A., “Stanford Protocol Proposal for Internet Search and Retrieval,” http://www-db.stanford.edu/~gravano/starts.html

[3] Gravano L., Garcia-Molina H., and Tomasic A., “GlOSS: Text-Source Discovery over the Internet,” to appear in ACM Transactions on Database Systems, 1999.

[4] Gauch S., Wang G. and Gomez M., “ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines,” Journal of Universal Computing, Springer-Verlag, Vol. 2 (9), 1996.

[5] Kwok, K.L, and Chan M., “Improving two-stage ad-hoc retrieval for short queries,” Proceedings of SIGIR '98, pp. 250-256, 1998.

[6] Liu L. “Query Routing in Large-scale Digital Library Systems,” Proceedings of ICDE’99, 1999.

[7] Manber U. and Bigot P. A., “The Search Broker,” Proceedings of the First Usenix Symposium on Internet Technologies and Systems, 1997.

[8] Sheldon M. A., Duda A., Weiss R., and Gifford D. K., “Discover: A Resource Discovery System based on Content Routing,” Proceedings of the Third International World Wide Web Conference, Computer Networks and ISDN Systems, 1995.

[9] Voorhees and Ellen M., “Expanding Query Vectors with Lexically Related Words,” in:Harman, ed., pp.223-231, 1994.

[10] Weider C., Fullton J. and Spero S., “Architecture of the Whois++ Index Service,” RFC1913, 1996.

[11] Xu J. and Callan J., “Effective retrieval with distributed collections,” Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 112-120, 1998.

[12] Zamir O. and Etzioni O., “Grouper: A Dynamic Clustering Interface to Web Search Results,” Proceedings of the Eighth International World Wide Web Conference, Computer Networks and ISDN Systems, 1999.

[13] AllSearchEngines.com, http://www.allsearchengines.com/

[14] AltaVista, http://www.altavista.com/

[15] Ask Jeeves, http://www.ask.com/

[16] Epicurious.com, http://epicurious.com/

[17] InvisibleWeb, http://www.invisibleweb.com/

[18] KidsHealth.org, http://www.kidshealth.org/

[19] SEARCH.COM, http://www.search.com/

[20] VacationSpot.com, http://www.vacationspot.com/

[21] Yahoo!, http://www.yahoo.com/

Vitae

Atsushi Sugiura received B.E. and M.E. degrees in electric engineering from the University of Osaka in 1988and 1990. He then joined NEC Corporation, where he is currently an assistant research manager at the C&C Media Research Laboratories. From 1998 to 1999 he was a visiting scientist at Department of Computer Science and Engineering, the University of Washington. His research interests include human computer interactions, visual programming, programming by demonstration, and intelligent software agents.

Oren Etzioni is an Associate Professor in the Department of Computer Science and Engineering at the University of Washington. He received his Ph.D. from Carnegie Mellon University in 1991. After joining the University of Washington he launched the Internet Softbot project. He received an NSF Young

Investigator Award in 1993. His research interests include software agents, Web navigation and search technology, and human-computer interaction. See http://www.cs.washington.edu/homes/etzioni.