Which Should We Try First? Ranking Information Resources through Query Classification

Joshua Church and Amihai Motro

Department of Computer Science
George Mason University
Fairfax, VA, USA
jchurch2@gmu.edu

Technical Report GMU-CS-TR-2010-17

Abstract. Users seeking information in distributed environments of large numbers of disparate information resources are often burdened with the task of repeating their queries for each and every resource. Invariably, some of the searched resources are more productive (yield more useful documents) than others, and it would be undoubtedly useful to try these resources first. If the environment is *federated* and a single search tool is used to process the query against all the disparate resources, then a similar issue arises: Which information resources should be searched first, to guarantee that useful answers are streamed to users in a timely fashion. In this paper we propose a solution that incorporates techniques from text classification, machine learning and information retrieval. Given a set of pre-classified information resources and a keyword query, our system suggests a relevance ordering of the resources. The approach has been implemented in prototype form, and initial experimentation has given promising results.

1 Introduction

Users seeking information in distributed environments of large numbers of disparate information resources are often burdened with the task of repeating their queries for each and every resource. Examples include searching for news items on a specific topic among hundreds of news feeds; searching for a job or an apartment in multiple classified ads repositories; or searching for technical advise in a multitude of support sites and discussion groups. Invariably, some of the searched resources are more productive (yield more useful documents) than others, and it would be undoubtedly useful to try these resources first.

If the environment is *federated* and a single search tool is used to process the query against all the disparate resources, then a similar issue arises: Which information resources should be searched first to guarantee that useful answers are streamed to users in a timely fashion.

This issue can be formalized abstractly as follows. Consider a collection $\{R_1, \ldots, R_n\}$ of information resources (e.g., document collections), assume a query Q is processed in the entire collection $R = \cup_1^n R_i$, and let A be its *ranked* answer. Let A_i be the subset of A that originates from R_i (the subsets A_i are not necessarily disjoint) and let ω_i denote the *contribution* of A_i to A. We define the *ranking* of the resource environment R for the given query Q as the ordering of the individual resources R_i according to their ω_i values. Sub-answer contribution could be measured in any of a number of ways, and should reflect both the total number of answer elements in A_i (quantity) and their relative ranking (quality).

The challenge we address in this paper is to design a methodology that *approximates* this order. That is, given a query Q against a collection of information resources R, rank the resources in R in terms of their expected contribution to the query Q. With such a ranking, users who seek answers to queries in a multi-source environment (or meta-searchers that process queries in such environments) can achieve their goals more effectively.

The solution we propose is based on techniques from information retrieval, text classification and machine learning, and it makes several simplifying assumptions. It assumes that each of the information resources in R is *homogeneous*; that is, its documents are on a single subject. Specifically, it assumes that the given information resources have been classified with a pre-determined set of $C_1, \ldots, C_p$ categories (labels). Given a query Q, we attempt to *classify* it by the same set of categories; but rather than settle on a single classification, each query Q results in a *ranked list of classifications*. This list, in turn, implies an order of the information resources, which we suggest as an approximation of the order defined above.

Thus, the main challenge is to classify a query; that is, to map Q to a permutation of $C_1, \ldots, C_p$. The main resource in our classification is a *semantic index*. This index is constructed from training documents, in a process that combines content acquisition, feature extraction, and latent semantic analysis (LSA) [5]. It provides the *background knowledge* necessary for classification. Essentially, this semantic index is an approximation of the traditional matrix of features (terms) by documents, in which the number of features has been reduced in a procedure called Single Value Decomposition (SVD). This new representation is known to mitigate the classical problems of synonymy (different terms have the same meaning) and polysemy (a term has multiple meanings). Into this space we also cast the query and compute the documents that are its k nearest neighbors using the traditional *cosine* similarity measure. Since each of the documents has an associated category C_i, the k nearest neighbors provide a multiset of categories. Using a voting approach, this multiset is used to infer a classifying order of the categories. If a final step, each category in the ordered classification of Q is replaced by the resources in R that are associated with this category.

The architecture of the system is described in detail in Section 3. This architecture was implemented in prototype form, weaving together readily available software components to perform the necessary content acquisition, feature extraction, learning and classification.

Our experiments are described and discussed in Section 4. Essentially, the experiments were designed to (1) validate the feasibility of the architecture and to measure its performance, and (2) to draw conclusions as to the optimal values of two important classification parameters: the number of dimensions (features) used in the semantic index and the number of closest neighbors used in the classification. Our experiments showed that even with moderate amount of training (3533 documents classified by 11 categories), effectiveness of 71% (as measured by the F-measure) can be achieved. These results were achieved with relatively small values for the number of features (200) and the number of closest neighbors (25).

Section 5 concludes the paper with a brief summary and directions for further research. To put our work problem in its appropriate context, we begin with a brief discussion of related work.

2 Background

We assume that readers are familiar with basic concepts of information retrieval [10] and we focus our attention on two active areas of research that relate strongly to this work: database ranking and query classification. We note that database ranking may refer to two different tasks: ranking answers (sets of rows) that are retrieved from a database in response to queries, listing "better rows" first (an early example of this may be found in [11]), or to ranking of a collection of databases with respect to a particular information need. Our work here, and therefore this review, concerns the latter.

Keyword-based selection of multiple structured data sources is the subject of [16]. The authors construct keyword summaries of the databases that participate in the ranking process. Given a query, they compare its keywords to each database summary and measure its proximity to the database's schema and content. The problem is original and the effectiveness of the approach has been demonstrated. Yet, keyword matching suffers intrinsically from poor precision and recall due to vocabulary mismatch [10]. For example, if users submit synonymous keywords that do not occur in database's keyword summary, they might miss relevant resources. Our work addresses this issue by using latent semantic analysis, a method known for its ability to handle synonymy and polysemy [7].

Text categorization is an intensively researched area, and query classification, a sub-area, has been very active recently [12, 15, 13, 9, 2, 3, 8]. Our task is essentially to learn a text categorizer that maps short, noisy, and ambiguous sets of keywords to relevant resources [9]. In general, query classification research varies by (1) the machine learning (ML) approach and (2) the type of training data.

A basic necessity to any query classification task is the selection and acquisition of training data that are representative of users' queries [9, 12]. Essentially, a large collection of labeled content that is both general and adaptive enough to categorize queries is hard to find. Training data are often obtained from search engines results [15, 3, 12, 13], click-through data in query logs [2, 1], or open directory services [15, 1]. Our solution utilizes document feeds that conform to

the RSS (Really Simple Syndication) protocol — a widely used format for disseminating content on the Web. Typically, each such feed contains articles on a particular topic. The assumption is that a sufficiently large set of documents obtained from each feed provide a reliable representation of the feed for the purpose of future classification.

We employ a classification technique called *transfer knowledge* or *background knowledge* which is recognized as an effective method for creating general purpose classifiers [4, 14, 17]. Background knowledge leverages training data from one classification task to apply to another related classification task [4, 14, 17]. Although previous research suggested the use of background knowledge [15, 3], it considered the query answer as the training material; that is, the search results are examined for patterns that explain the query. Other classifiers described in the literature include rule-based classifiers [2], pattern matchers [13, 12, 2], Support Vector Machines (SVM) [13, 12], and probabilistic classifiers [15, 3]. Our classifier's performance is linear in the number of documents, which makes it suitable for large-scale deployment. Moreover, our solution classifies queries without knowledge of the corresponding result set, a feature essential in the application of query classification to resource ranking.

3 System Architecture

Our system consists of two principal phases common to classification tasks: *preparatory indexing* and *request processing*, with the former providing the knowledge necessary for the latter. Figure 1 illustrates this architecture. Preparatory indexing consists of three stages: content acquisition, feature extraction, and semantic indexing. In two additional stages, request processing uses two system assets — the output of the semantic indexing and a classified catalog of the available resources — to assign a set of resources to each user query. A more detailed description of these five stages follows.

3.1 Preparatory Phase

In the first phase, training documents are processed and preserved using the well-known vector space model of information retrieval. The product of this phase is an LSI index to be used for query processing.

Content Acquisition. Initially, users of this system choose a set of *labels* (categories) to be used in classification. For each such label, one or more document collection is selected. Each collection should include documents that correspond primarily to the particular label. This collection provides relevant background to *interpret* the meaning of the label. In this work we chose to work with document collections that are RSS feeds. Custom Java software that incorporates the ROME Java software library is used for managing feeds. Each document is retrieved with an HTTP request, its content is parsed, and the result is saved to a local store.

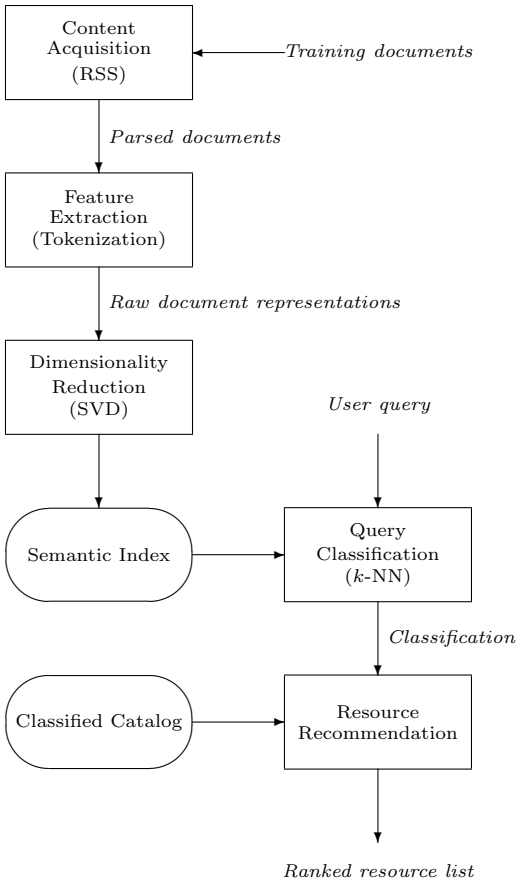

Fig. 1. System architecture

Feature Extraction. In the next stage, each parsed document is processed to obtain a list of features (terms). It is assumed that the features extracted are characteristic and descriptive of the label associated with the feed.

Semantic Indexing. Let the total number of acquired documents be m, and let the total number of extracted features be n. The documents are now represented in a matrix of n rows and m columns, with the value in position (i, j) denoting the *relevance* (significance) of the ith feature to the jth document. We measure this relevance with the well-known concept of *entropy*: Let $f_{i,j}$ denote the number of occurrences of the i'th feature in the j'th document, then $\Sigma_{j=1}^{m}(f_{i,j})$ is the total number of occurrences of this feature in all the documents, and $p_{i,j} = f_{i,j}/\Sigma_{j=1}^{m}(f_{i,j})$ is the relative frequency of occurrence. The value stored in

position (i, j) is $p_{i,j} \cdot \log p_{i,j}$. Typically, each document will contain only a small number of the possible features, resulting in a very sparse matrix. The row-dimensionality of this matrix is then reduced, first by eliminating features that correspond to common words ("stop-words"), and then by using Single Value Decomposition (SVD). A complete explanation of SVD is outside the scope of this paper and may be found in [7].

3.2 Request Processing

The second phase is the repetitive processing of user queries. It involves two stages: query classification and resource ranking.

Query Classification. The output of the preparatory phase is a *semantic index*: a set of vectors representing the documents by means of their features. Initially, each user query is transformed into a similar vector using the same SVD process that was applied to the original matrix. Next, this vector is compared to all the vectors in the semantic index, and, using the well-known *cosine* measure of similarity, its k nearest neighbors (k-NN) are determined [6]. Since each of the k documents originated from a particular collection (feed), it is associated with a particular label. The k labels thus obtained are tallied and *ranked* according to their rate of occurrence in the set. In other words, the documents closest to the query "vote" on its classification.

Resource Ranking. In the final stage we assume that the collection of available resources has been pre-classified using the same set of labels. (Indeed, it may be assumed that the set of labels has been derived from this classification.) This catalog of resources is now used to match the query with a ranked list of resources that correspond to its ranked classification.

It should be noted that, in essence, there are three classifications in this work, and they use the same set of categories: The training documents and the catalog resources are assumed to be pre-classified, whereas user queries are classified by the system.

4 Experimentation and Discussion

To validate the approach outlined in this paper we conducted an experiment of moderate size. Our objective was two fold. The first objective was to validate that the architecture that we proposed can indeed deliver good results. The second objective was to experiment with two important parameters of query classification, namely the number of dimensions with which a document is represented, and the number of closest neighbors that would be used to vote on the classification.

4.1 Datasets

Training Documents. Our system uses 11 different classification labels typical in the newspaper domain; for example, business, sports, health, education, and so on. For training the system, we used RSS feeds of the Washington Post newspaper. RSS feeds are increasingly the dissemination method of choice of on-line resources, and the advantage of using newspaper feeds and labels is that the documents been classified by human editors and thus provide authoritative interpretation for the labels. A total of 34 RSS feeds were sampled and a total of 3533 documents were extracted. Figure 2 shows the breakdown of the 3533 documents by the 11 categories. As can be seen, the distribution of the documents is relatively balanced. The resource catalog used in the final classification stage assigns each resource to one of the 11 categories. A small example of such a catalog is shown in Table 1.

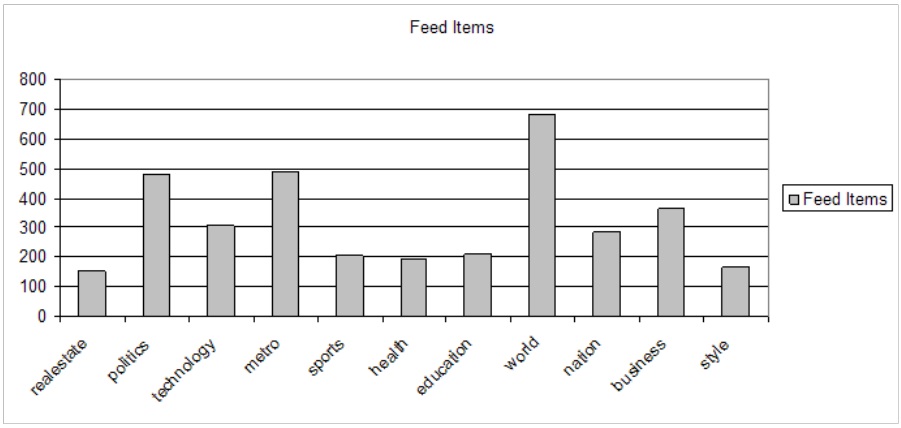

Fig. 2. Histogram of training documents

Test Queries. Our experiment used actual user queries submitted to the Google Web search engine. Specifically, the most frequent 100 queries for a given day were collected over a period of two weeks. From this set of 1,400 queries, a random sample of 66 queries was selected, providing for confidence level of 90%. These queries, typically a few keywords each, were classified "manually" using the same 11-label scheme. These authoritative classifications were later used to measure the accuracy of the classifications generated by the system.

4.2 Experimental Results

After the preparatory phase was completed, each of the 66 queries was classified 30 times, using 6 different values for the number of dimensions and 5 different

Category	Resources

Business	`http://www.forbes.com/fdc/rss.html`
	`http://feeds.fool.com/usmf/foolwatch`
	`http://online.wsj.com/xml/rss/3_7086.xml`

Sports	`http://sports.yahoo.com/top/rss`
	`http://sports.espn.go.com/espn/rss/nfl/news`
	`http://content.usatoday.com/marketing/rss/rsstrans.aspx?feedId=sports1`

Technology	`http://rss.news.yahoo.com/rss/tech`
	`http://feeds.wired.com/wired/index`
	`http://www.infoworld.com/rss/news.xml`

Table 1. A classified catalog (partial)

values for the number of closest neighbors. A classification was correct, if the manually-assigned category matched the *top* predicted category. The classification of the set of 66 queries with a specific number of dimensions and neighbors was considered a single experiment, whose success was measured with the F-measure (the harmonic mean of the precision and recall). Figure 3 summarizes the results of the 30 experiments.

	dimensions				
neighbors	50	100	200	300	400
1	0.538064	0.552026	0.552026	0.552026	0.509091
10	0.675138	0.675138	0.696651	0.696651	0.61708
25	0.686009	0.686009	0.707071	0.707071	0.629213
50	0.686009	0.686009	0.707071	0.707071	0.629213
100	0.686009	0.686009	0.707071	0.707071	0.629213
200	0.686009	0.686009	0.707071	0.707071	0.629213

Fig. 3. F-Measure at various levels

An analysis of the variance of these results (using two-way ANOVA without replication) concluded that there is no significant interaction between the number of dimensions and the number of neighbors chosen, suggesting that they could be optimized independently. Observing the impact of dimensionality on the F-measure (Figure 4), it is apparent that increasing the number of dimen-

sions improves performance through 200 dimensions, provides no improvement when the number is increased to 300, and worsens performance substantially thereafter. Observing the impact of the number of neighbors on the F-measure (Figure 5), it is apparent that increasing the number of closest neighbors (the number of documents that "vote" on the classification) improves performance through 25 neighbors. Once this number is reached, the quality of the classification remains unchanged. Combined, these three somewhat surprising conclusions suggest keeping the number of dimensions at 200 and the number of neighbors at 25.

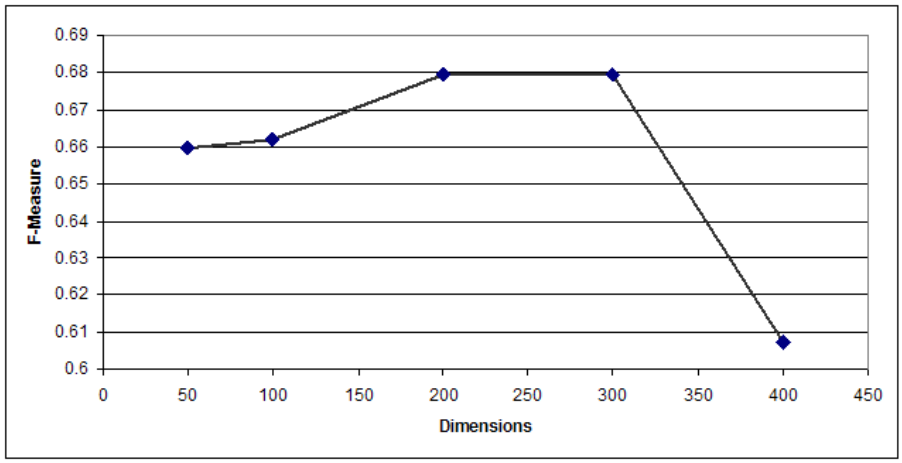

Fig. 4. F-measure *vs.* dimensions

4.3 Discussion

The overall results of this experiment are promising. Roughly speaking, the system can classify a query correctly (and recommend the appropriate resources) about 71% of time. And with various extensions and refinements (to be discussed later) we expect even further improvements.

The results of our experimentation with different classification parameters are also noteworthy. Intuitively, as the number of features used to describe a document increases, the semantics of documents are captured with more fidelity, and hence classification should be more accurate. Similarly, a larger number of close neighbors should be expected to minimize possible "noise" caused by the misclassification of some documents, resulting in a more robust classification. In practice, in both instances, increasing the corresponding parameters proved beneficial, but only to a certain level, beyond which there were no improvements (and in the case of the dimensionality of vectors, performance eventually started

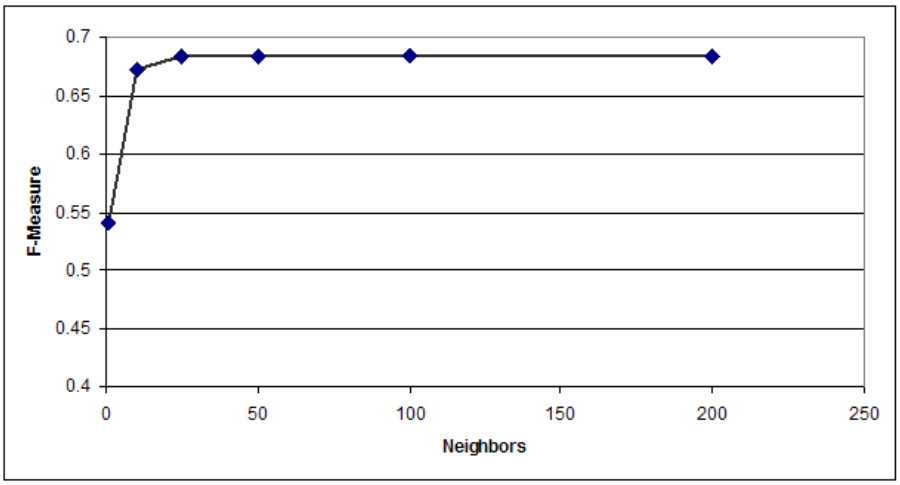

Fig. 5. F-measure *vs.* neighbors

to decline). These two results, combined with the discovery that these two parameters do not interact and could thus be optimized independently, suggest that "more is not always better". A conclusion that has positive impact on time performance.

5 Conclusions and Future Work

The availability of multiple information resources against which a query may be processed raises the issue of which information resources would prove to be more productive. In this paper we addressed this issue in the context of resources that are collections of documents and queries that are sets of keywords. Specifically, given a large number of document collections and a keyword query, rank the collections in the order of relevance; that is, resources that are more likely to yield documents relevant to the query should be listed earlier.

To address this issue, we proposed and tested a machine learning approach in which we assumed that the given resources have been pre-classified by a set of categories, and the challenge is to correctly classify a given query by the same set of categories. Our query classification method is based on the notion of similarity, and produces not a single category, but a ranked order of categories. These, in turn, suggest a ranking order of the corresponding resources.

Our prototype system combined off-the-shelf tools for RSS feed acquisitions, feature extraction, and latent semantic indexing. The information resources we used were RSS feeds of a major newspaper. Initial experimentation demonstrated that F-measures of 71% can be achieved with moderate size background knowledge.

For this approach to work well, it is important to use sufficient number of training documents and use RSS feeds that accurately characterize the topics that interest the user. Note that available collections of resources need not be static, as they can be updated periodically with "feed crawls".

There are many opportunities for further research and we mention here just four. First, when classifying a query we ranked the categories by their frequencies in the nearest neighbors set. This corresponds to a simple voting procedure in which all neighbors have equal votes. Another possibility here is to use a *weighted voting scheme*, in which the vote of each neighbor is weighted by its proximity to the query.

Second, we assumed that the information resources in the catalog have been pre-classified. A familiar argument is that this manual classification is laborious and inaccurate. An attractive proposition is to apply similar machine learning techniques to *automate the classification of the resources* as well.

Third, we assumed a pre-determined set of categories (business, sports, politics, and so on). Alternatively, we could *obtain the set of categories* from the classification of the resources (which was suggested above), and then use these in the other two classification processes (document training and user queries).

Finally, the classification of each query was by a ranked order of categories. With a small effort this classification could be converted into a *weighted vector* of categories. On the other hand, each of the given information resources have been assumed to fall into a single category (a somewhat restrictive assumption). An attractive approach is to classify each resource with a similar weighted vector of categories, and then use a similarity measure (such as the cosine) to find the resources that are the *closest neighbors* of the query classification vector.

References

1. J. Arguello, J. Callan, and F. Diaz. Classification-based resource selection. In *Proceedings of CIKM-09, 18th ACM Conference on Information and Knowledge Management*, pages 1277–1286. ACM, 2009.
2. S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury, and O. Frieder. Automatic classification of web queries using very large unlabeled query logs. *ACM Transactions on Information Systems*, 25(2):Article 9, April 2007.
3. A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of SIGIR-07, 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 231–238. ACM, 2007.
4. C. B. Do and A. Y. Ng. Transfer learning for text classification. In *Advances in Neural Information Processing Systems 18 (NIPS)*, 2005.
5. S. T. Dumais. Latent semantic indexing (LSI) and TREC-2. In *Proceedings of the Second Text Retrieval Conference*, pages 105–116. National Institute of Standards and Technology (NIST), Special Publication 500-215, 1993.
6. R. Feldman and J. Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.
7. G. W. Furnas, S. C. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using a singular value

decomposition model of latent semantic structure. In *Proceedings of SIGIR-88, 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 465–480. ACM, 1988.

8. X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query dependent ranking using k-nearest neighbor. In *Proceedings of SIGIR-08, 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–122. ACM, 2008.
9. Y. Li, Z. Zheng, and H. K. Dai. KDD CUP 2005 Report: facing a great challenge. *SIGKDD Explorations Newsletter*, 7(2):91–99, December 2005.
10. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
11. A. Motro. VAGUE: A user interface to relational databases that permits vague queries. *ACM Transactions on Information Systems*, 6(3):187–214, July 1988.
12. D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Q2C@UST: Our winning solution to query classification in KDD CUP 2005. *SIGKDD Explorations Newsletter*, 7(2):100–110, December 2005.
13. D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *Proceedings of SIGIR-06, 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 131–138. ACM, 2006.
14. M. E. Taylor and P. Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of ICML-07, 24th International Conference on Machine Learning*, pages 879–886. ACM, 2007.
15. D. Vogel, S. Bickel, P. Haider, R. Schimpfky, P. Siemen, S. Bridges, and T. Scheffer. Classifying search engine queries using the web as background knowledge. *ACM SIGKDD Explorations Newsletter*, 7(2):117–122, December 2005.
16. B. Yu, G. Li, K. Sollins, and A. K. H. Tung. Effective keyword-based selection of relational databases. In *Proceedings of SIGMOD-07, the 2007 ACM SIGMOD International Conference on Management of Data*, pages 139–150. ACM, 2007.
17. S. Zelikovitz and H. Hirsh. Using LSI for text classification in the presence of background text. In *Proceedings of CIKM-01, 10th ACM International Conference on Information and Knowledge Management*, pages 113–118. ACM, 2001.