Product Query Classification

Dou Shen, Ying Li, Xiao Li, Dengyong Zhou
Microsoft Corporation
One Microsoft Way, Redmond, WA, USA
{doushen, yingli, xiaol, denzho}@microsoft.com

ABSTRACT

Web query classification is an effective way to understand Web user intents, which can further improve Web search and online advertising relevance. However, Web queries are usually very short which cannot fully reflect their meanings. What is more, it is quite hard to obtain enough training data for training accurate classifiers. Therefore, previous work on query classification has focused on two issues. One is how to represent Web queries through query expansion. The other is how to increase the amount of training data. In this paper, we took product query classification as an example, which is to classify Web queries into a predefined product taxonomy, and systematically studied the impact of query expansion and the size of training data. We proposed two methods of enriching Web queries and three approaches of collecting training data. Thereafter, we conducted a series of experiments to compare the classification performance of using different combinations of training data and query representations over a real data set. The data set consists of hundreds of thousands queries collected from a popular commercial search engine. From the experiments, we found some interesting observations, which were not discussed before. Finally, we proposed an effective and efficient product query classification method based on our observations.

Categories and Subject Descriptors

H.3.m [**Information Storage and Retrieval**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

Product Query Classification, Query Enrichment

1. INTRODUCTION

More and more Web users take Web search engines as an indispensable tool for obtaining the desired information. In order to return the correct information to end users, in terms of either Web pages, or advertisements, search engines have to accurately understand the users' intents. Clearly, there are many aspects which can impact users' intents, including users' demographics, users' locations and so on. Among all of these aspects, Web queries issued by the users explicitly and directly reflect the users' intents. Therefore, many efforts have been conducted to estimate users' intents through Web queries [3, 12, 6, 21]. In this paper, we will focus on Web query classification, which can be used to infer users' intents by classifying the Web queries into some predefined taxonomies. For example, once we know the queries "sd450" and "lenovo x61" belong to "Camera" and "Computer" respectively, we can recommend some relevant information to users even if we do not have the exact matched information about "sd450" and "lenovo x61".

However, as noted in previous work [3, 12, 23], Web query classification is quite challenging due to two factors. One is that Web queries are very short, containing less than 3 terms in most cases, which cannot provide enough clues for classification. Another is that it is hard to collect enough training data since the queries may be too short for the labelers to understand without extra efforts such as resorting to search engines. To solve the first problem, researchers have exploited ways to enrich Web queries. For example, [23] uses Web search results to represent Web queries, which can help understand Web queries. For the second problem, some work has tried to automatically expand a small training data set using click-through log data [12], or take labeled Web pages as the training data [23].

Limited by the availability of large scale data sets, previous work did not exploit several important aspects of Web query classification. For example, some work adopts existing Web pages as training data to train classifiers for Web query classification. However, are the classifiers trained over Web pages applicable to Web query classification? Some work uses Web search snippets returned for queries to represent Web queries. Is this an effective way? Do we still need to enrich queries when we have enough training data? If query enrichment through Web search is helpful for query classification, can we find an approach which can effectively and efficiently classify Web queries without relying on Web search too much?

In this paper we take product query classification as an example and try to answer the above questions through a comprehensive study over a large data set collected from a commercial search engine. In this paper, we exploit two ways to enrich Web queries. One is the well studied method which enriches Web queries by search snippets. Another one

CIKM'09, November 2–6, 2009, Hong Kong, China.

is to expand queries by their similar queries which can be automatically discovered from click-through log data. Experimental results show that both methods can improve the classification accuracy significantly while the former performs better in most cases.

On the training data side, besides using labeled queries (as well as their enriched representations) to train classifiers, we also build classifiers using labeled product names. Product names are effective representation of products, based on which we can build accurate product classifiers. However, due to the difference of languages for Web queries and product names, classifiers trained over product names are much worse than those trained over queries. To remove the gap between product names and Web queries, we put forward a solution to translate product names to Web queries (which we call pseudo queries to differentiate them from real queries). Then we can train classifier using the pseudo queries. Although these classifiers cannot beat those trained over labeled queries, they are clearly better than the classifiers trained over product names directly. To study the impact of training data size, we vary the size of the training data and compare the different query representation methods. We found that the smaller the training data set, the larger the difference between different query representation methods. Using search snippets to represent Web queries achieves the best results compared to other methods. However, retrieving search snippets is time consuming and it may be infeasible for a large scale query set. Therefore, we try to minimize the dependency over search snippets. We build a classifier using original queries and classify Web queries directly. Only when we cannot classify the queries with high confidence, we use snippets. With this solution, we need to retrieve snippets for less than 30% queries and maintain the accuracy as using search snippets for all Web queries.

The rest of the paper is organized as follows. We discuss the related work in Section 2. Section 3 presents the query representation methods and training data selection. Section 4 describes the data sets used for experiments, evaluation methodology, experiments and results. Section 5 gives conclusion of the paper and some possible future research issues.

2. RELATED WORK

Web queries are currently the major bridge between Web users and search engines. Accurately understanding Web queries is of great importance. Web query classification provides such a way. Generally speaking, query classification can be split into two groups: (1) classification according to queries' types, such as informational or navigational or transactional [2, 9, 10]; (2) classification according to the queries' topic, such as "computers/hardware" or "computers/software" [4, 23].

For query type classification, Broder classifies Web queries according to their intents into three types: informational, navigational and transactional in [2]. Although Broder does not provide a way to distinguish different types of queries, he makes an informative survey which shows that the navigational queries takes up 24.5% while the informational queries and transactional queries take up 39% and 36% approximately. Following Broder's work, Lee et al. further study whether the types of a query is predictable and how to predict it [10]. They propose two types of features for the query type classification: user-click behavior and anchor-link distribution, which are proved to be quite effective empirically.

Besides working with the taxonomy defined by Broder, people conduct query type classification in other directions. For example, Gravano et al. discuss how to categorize queries according to their geographical locality [7]. Similar work on locality of Web queries is conducted in [16, 25]. Another work related to query type classification is presented in [6]. The authors focus on capturing commercial intention from search queries and Web pages.

In query topic classification, queries are classified into some predefined categories according to queries' topics or subjects. At the very beginning, some researchers manually classified Web queries for query analysis, especially on the query topic distribution in query logs [24]. Recently, automatic topic classification techniques have been exploited and they have been used for building query filters, study of user interests and enriching Web taxonomies [19].

In [19] and [4], the authors work on query topic classification independently with the similar techniques. Both of them use real-world search engines to obtain highly ranked Web documents based on each unknown query. These documents are used to extract cooccurring terms and to create a feature set. They can then use these features to judge the most appropriate categories of a query. Similarly, both [23] and [3] rely on Web search results to enrich Web queries for classification, except that [3] focuses on rare queries while [23] takes an existing taxonomy as the bridge to connect Web queries and the target categories. Instead of relying on Web search results, Beitzel et al. study the topic query classification in a different direction [1]. Given a set of labeled queries and unlabeled queries, Beitzel et al. exploit several classification approaches including exact-match using labeled data, N-Gram match using labeled data, classifiers based on perceptron and an approach adapted from computational linguistics named selectional preferences [13]. Different from previous work which emphasizes on feature representation of Web queries, Li et al. study how to drastically increase the amount of training data by semi-supervised learning with click-though log data [12]. They come to the conclusion that automatically increasing the training data can lead to significant improvement in classification performance.

In this paper, we will focus on query topic classification and take product query classification as an example. In previous work about query topic classification, as we mentioned above, researchers have focused on either how to represent Web queries (such as by using search results), or how to get more training data. Little work has been done to exploit the impact of Web query representation when varying the amount of training data. What is more, the data sets used in previous studies are relative small. In this paper, we are going to employ hundreds of thousands Web queries collected from a commercial search engines for comprehensive comparisons among different classification methods.

3. METHODOLOGY

3.1 Problem Definition

Query topic classification is to classify Web queries into a predefined taxonomy based on the queries' topic, or aboutness. It is closely related to conventional text classification, such as document or Web page classification. However, in query topic classification, the objects are Web queries, which are much shorter than documents/Web pages in text classification and thus become more ambiguous. For example, "ap-

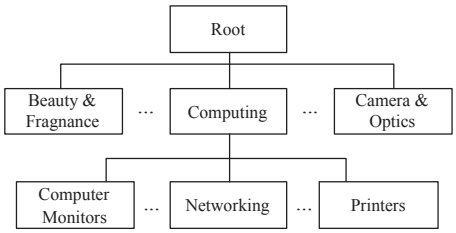


Figure 1: Illustration of product taxonomies

ple" can mean either a kind of fruit, or an IT company. This makes the query topic classification problem even harder.

We highlight "topic classification" since we want to differentiate this problem from query type or functionality classification. Query type classification is to group the queries according to their functionality. For example, when people submit a query "american airlines home", they probably have a particular site in mind and want to reach it directly through this query. However, for queries like "cars", people may want to find more information about "cars". Therefore, the former group of queries are called "navigational queries", while the latter is referred by "informational queries" [2]. It is clear that we need to consider different features for different classification problems (topic classification, or type classification). In this paper, we will focus on the topic classification problem.

More specifically, we will work in a special domain, that is, product query classification. The goal is to classify a query into an existing product taxonomy, which includes categories like "Camera & Optics", "Computing", and so on. The taxonomy may have a hierarchical structure, as it is in most cases. Figure 1 shows part of taxonomy used in this paper. We choose this domain because product queries are very important for both Web users and commercial search engines, especially when more and more people tend to purchase what they need through Internet or at least conduct some online research before the actual deals happen. Another reason is that we have plenty of data in this domain, over which we can conduct solid experiments and provide insights for classification problem in other domains.

3.2 Query Enrichment

As we mentioned above, Web queries are usually very short, consisting of less than 3 terms on average [23, 8]. Such short text may not be enough to reflect the aboutness of the queries well, which results in bad classification performance. Therefore, people have tried to enrich the Web queries as the first step. In the following, we will introduce two ways for query enrichment. One is using Web search, which is widely adopted in previous work [23, 3]; another one is to use similar queries where the similarity is estimated from click-through log data.

3.2.1 Using Search Snippets

Given a query, a straightforward way to catch its meaning is through its contexts. For example, even we do not know what "SD450" is, but we can make a safe prediction based on the text surrounding it, such as "...Canon PowerShot SD450 digital camera specifications...". We can use many approaches to obtain such contexts. One convenient and popular one is using search engines, which can provide a list of Web pages given a certain query. For most popular search engines, the returned Web pages are sorted according to their relevance to the query, along with the URL, title, a piece of text, which is often called "snippet". The snippet is a kind of query-dependent summary, which provide the context surrounding the query. Previous studies have validated the effectiveness of using Web search results to represent Web queries and found that titles and snippets of the returned pages are good to represent Web queries, while the text of the whole Web page may introduce noise. Therefore, in this paper, we will follow them and use titles and snippets to represent Web queries.

One common debate about using search snippets is that we expect search engines to return more relevant Web pages based on the query classification results, then how we can rely on search results for query classification. As we stated above, what we need is a method to get the contextual information around the queries. Web search engines provide such a way. Even if the search results are not strongly related to the Web queries, we can still use it to get the contextual information. Actually, the same idea has been exploited in pseudo relevance feedback, which can greatly improve search performance [15].

3.2.2 Using Query Similarity

Instead of using the search snippets returned by search engines, we can use the similar queries to enrich a target query. For example, even if we know nothing about "sd450", we can infer its meaning from its similar queries, such as "powershot sd450", "canon elph" and so on. Similarity among queries can be automatically estimated from the click-through log data collected by commercial search engines. As we know, after search engines return the pages according to a user's query, the user may click some pages. The user, the submitted query, as well as the clicked pages constitute the click-through data. Intuitively, the queries leading to the same clicked pages are similar in terms of their aboutness. Based on this observation, which is widely used in previous study, we put forward a way to calculate query similarities and further enrich queries based on the calculated similarity.

Specifically, we construct a bipartite graph over queries and URLs [17]. If a user clicks a URL given a query, we add a link from the query to the URL. The link is further weighted by the click number. With this graph, a query's similar queries can be found by a random walk starting from the given query. The random walk repeatedly travels from queries to URLs and then from URLs to queries. The transition probabilities are proportional to the weights. Finally, the similarity between two queries is then measured by the hitting times, which are the expected number of the random walk from one query to another.

3.3 Choose Training data

The quantity and quality of training data is critical for most classification problems. As we discussed above, it is hard to collect large scale manually labeled Web queries. Therefore, some previous work takes labeled Web pages as the training data [23], while others starts from a small labeled query set and automatically discover more queries for training [12]. Since we are working on product query classification, we take labeled products as training data accordingly. Considering the difference between the languages of queries and products, we propose to translate the labeled products to labeled queries, which can be used as training

data. Finally, we can also obtain training data from the click-through log data with some heuristic rules.

3.3.1 Product Data

With the development of online shopping, more and more products are available online with labels, as those in MSN Shopping (http://shopping.msn.com), Amazon (http://www.amazon.com) and eBay (http://www.ebay.com). In this paper, we collect the products from MSN Shopping. Most of the products have product names, product description and other attributes like color, size and weight. Clearly, product names and product descriptions are more proper to represent the products in the context of Web query classification since the queries usually do not contain information about detailed attributes. We further compared product names and product description empirically and found that classifiers trained over product names are much better than those trained over product descriptions. Therefore, we finally use the product names (associated with labels) as the training data in this paper.

3.3.2 Pseudo Query

As we will discuss in Section 4, the performance of the classifiers trained over product data is not very promising. One of the possible reasons is that the language of product names is different from that of product queries. In other word, the feature space (details about feature generation are presented in Section 4.2) of the training data is different from the feature space of test data. To overcome this problem, we put forward a method to translate the labeled product names to labeled queries. We call the translated queries as "Pseudo Queries" to distinguish them from the real queries.

Specifically, we generate pseudo queries based on the idea and method of statistical machine translation. We treat product names as sentences in the source language, and their corresponding query forms as those in the target language. Given a set of training sentence pairs (which will be discussed shortly), we learn a translation model based on tuple ngrams [5], while other translation models can also applied to our problem. At the highest level, a tuple-ngram-based translation model aims to learn the joint probability of a sentence pair, which is modeled by ngrams of phrase pairs. At translation time, the model finds the target sentence (query form) that maximizes the joint probability given a source sentence (product name). To deal with words/phrases that never occurred in training, we use the backoff strategy that a word/phrase can always be translated into itself or to *NULL* (*i.e.*, deletion). The technical details of this approach can be found in [11].

To obtain the training sentence pairs, we leverage a large amount of click-through data in the product search domain. In product search, each search result corresponds to a distinct product item in the data feed. Whenever a user issues a query and clicks on a product item, we create an association of that query and the corresponding product name as a sentence pair, and we select the most frequent such pairs to form our training data.

3.3.3 Original Query and the Expansions

With the labeled product data, we can obtain the labels of some queries from click-through log data with heuristic rules. Intuitively, among the returned search results from a commercial search engine for a query q, if most of the users click on the products with the same label l, we can claim that we should label q by l. Formally speaking, give a query q, we aggregate its associated clicked products according to their categories and get a set $click(q) = \{(l_1, n_1), \ldots, (l_i, n_i), \ldots, (l_m, n_m)\}$, where l_i is a category (or label), n_i is the number of clicked products belonging to l_i, m is the number of categories. Then we can label q by l_k if n_k meets the following condition. Note that a query may have more than one label as labeled in this way.

$$\frac{n_k}{\sum_{i=1..m} n_i} > t \tag{1}$$

This method is kind of voting from the Web users, which reflects the views of the majority of the users. However, we have to admit that this method may introduce some wrong labels due to noisy clicks. Therefore, we will use a large t to reduce the wrong labels. After collecting the labeled queries using the above method, we can adopt the two query enrichment methods to expand the queries and thus enrich the representation of the training data.

Three natural questions arise here: Firstly, how consistent is it between the automatically generated query labels and the manually assigned query labels? Secondly, if we can automatically label the queries using the click-through log data, why do we have to build classifiers? Thirdly, does the classifier trained over the automatically collected data work well for the queries which cannot be automatically labeled using click-through data? For the first question, we empirically compared the automatically generated labels with the labels provided by three human labelers over 6,000 randomly sampled queries. We can see the consistency is very high, which ranges from 94.9% to 98.3%. For the second one, the answer is clear since not all the queries have enough appearances in the click-through log, especially for the new emerging queries, and thus require a classifier to do the classification. For the third one, we randomly selected 2,000 queries, which cannot be automatically labeled using the click-through data, and invited three human labelers to label them. The experiments show that the trained classifiers work pretty well on these queries.

4. EXPERIMENTS

4.1 Data Preparation

The taxonomy we use in this paper is from MSN Shopping, which consists of thousands of categories spanning over several levels. For simplicity, we consider two flat classification tasks, one is to classify the queries into the direct children of the root node, and another is to classify queries into the direct children of "Computing" (See Figure 1). We refer to the former as "Level-1" classification and the latter as "Computing" classification. For "Level-1", we have 26 categories including "Computing" and for "Computing" we have 11 categories. We choose two classification tasks because we want to demonstrate the contribution of different training data and query enrichment methods for classification tasks with different granularity.

Since it is hard and expensive to collect a large manually labeled query data set, we use the method introduced in Section 3.3.3 to automatically obtain the labeled data for experiments. We randomly sampled 710,670 queries which lead to 40,676,169 clicks from the click-through log collected

from MSN Shopping. Clearly, when the threshold t in the inequality (1) is small, a query may have more than one labels. The smaller t, the more labels a query may have. As shown in table 1, when the threshold is 0.0, where we count all the categories associated with the clicked products for a query, the average number of labels assigned to the queries is 1.388. By increasing the threshold, we see that both the number of labeled queries and the average number of labels per query become smaller. The average number of labels for the product queries is quite different from that of the general Web queries as presented in [23], which means that the product queries are not so ambiguous. This is one of the reason why the classification performance reported in this paper is much higher than that in [23]. In this paper, in order to obtain high quality labeled data, we set the threshold as 0.5, which results in 662,692 queries. Similarly, we collect 24,056 labeled queries for "Computing".

To clarify the first question raised in Section 3.3.3, we randomly selected 6,000 queries from the 662,692 queries. Then we invited 3 human labelers to labels the queries manually. For simplicity, we let them label the queries with the 26 categories in "Level-1" classification task. Actually, we can do the same evaluation over the "Computing" classification task. Table 2 shows the consistency of the labeling results among human labelers and the automatic rule based method. Consistency between two labelers (including the rule-based method) is measured by the percentage of queries which are assigned the same labels. The column "Majority" means that we combine the labeling results from the three labelers by voting when the labelers disagree with each other for a certain query. For a few queries, the labelers come to three different labels, then we invite a fourth labeler to make the decision. From this table we can see that the consistency between the rule based method and the human labelers is quite high, ranging from 0.949 to 0.983, which is 0.963 on average. The consistency between the rule-based method and the combination of the three labelers is 0.971, which assures us to conduct experiments over the automatically labeled data. Related to the observation of smaller average number of labels a product query may have, we find that the consistency among human labelers in this paper is much higher than that in [23].

To train the translation model as described in 3.3.2, we extracted 2M click events from a the product search query log in Live Search. We selected 500K unique (query, product) pairs that occurred more than once in the click data, which served as training sentence pairs for our translation model.

To calculate the query similarity as presented in 3.2.2, we use a query log from Live Search, which contains hundreds of millions queries and URLs. Adult queries are removed from the query log. We also removed the query and URL pair if the click number is below than a specific threshold (We set the threshold as 10 empirically in this paper).

4.2 Classifier and Feature Extraction

There are several state-of-the-art classification models including Support Vector Machines (SVM) , Logistic Regression (LR) [18] and so on. In this paper we take LR in that it is easy to implement, fast in training and achieves comparable classification performance in most cases [18]. LR is to model the conditional probability of labels given a query,

Table 1: Statistics of Labeled Data when Changing Threshold t

Threshold	#Query	Average #Label
0.0	710,670	1.388
0.1	704,191	1.189
0.2	697,523	1.103
0.3	691,291	1.049
0.4	685,360	1.024
0.5	669,692	1.000
0.6	656,694	1.000
0.7	628,346	1.000
0.8	608,429	1.000
0.9	553,694	1.000

Table 2: Consistency among Human Labelers and Automatic Rule based Method

	Labeler1	Labeler2	Labeler3	Majority
Rule Based	0.949	0.956	0.983	0.971
Labeler1		0.950	0.952	0.964
Labeler2			0.973	0.964
Labeler3				0.988

as shown below:

$$P(l|q) = \frac{exp^{\sum_j \lambda_j \phi_j(q,l)}}{\sum_l exp^{\sum_j \lambda_j \phi_j(q,l)}} \quad (2)$$

where q denotes a query and l denotes the possible labels. $\phi_j(q,l)$ represent the features extracted for the pair q and l. λ_j is the parameter corresponding to $\phi_j(q,l)$ which indicates the importance or contribution of the feature $\phi_j(q,l)$ in determining the probability of l given q. It is clear that two components are important for LR. One is how to define the features and the other is how to estimate the parameters. Several methods have been exploited in the literatures for estimating the parameters using a maximum likelihood objectives [18, 22]. In this paper, we adopt L-BFGS, which is shown to converge significantly faster than others [22].

Feature extraction is very important for classification problems and the optimal feature spaces vary a lot for different tasks. In this paper, we use unigram and bigram features as well as several regular expressions to generalize terms with certain patterns. Unigram and bigram are frequently used in language models [14], which refer to the single terms and two consecutive terms in a query respectively. Note that for bigrams, we treat sentence start "<s>" and sentence end "</s>" as two special terms. Before we generate the unigrams and bigrams, we do some conventional preprocessing work including converting the queries to lower cases, removing stopwords and stemming. We adopt some regular expressions to generate features since we want to leverage some informative patterns which are prevail in product queries. For example, products with the same label from the same producer may have the same patterns in their models, such as "SD430" and "SD450". Several examples of the patterns used in this paper include "<NUM>" (such as "2007"), "<CHAR><NUM>" (such as "sd450") and so on.

4.3 Evaluation Metrics

We employ the standard measures to evaluate the performance of query classification, i.e. precision, recall and F1-measure [20]. To evaluate the average performance across multiple categories, there are two conventional methods:

Table 3: Comparison between Products and Pseudo Queries

	Computing		Level-1	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1
Product	0.591	0.549	0.539	0.682
PseudoQry	0.654	0.615	0.554	0.700

micro-average and macro-average. Micro-average gives equal weight to every query, while macro-average gives equal weight to every category, regardless of its frequency.

For some experiments in this paper, we allow the classifier to predict the label of a query only when the confidence of the classifier is beyond a threshold. In these scenarios, we introduce two more measurements, accuracy and coverage. Coverage means the percentage of the queries for which the classifier's confidence is beyond a predefined threshold. Accuracy means the percentage of queries among the covered queries which are correctly classified. Accuracy is actually micro-precision when each query has only one label.

In our experiments, we split the labeled queries into training and test data. In order to reduce the uncertainty of data split, a 3-fold cross validation procedure is applied . That is, we randomly split the queries into 3 folds and pick up one fold as the test data and the other two folds as the training data each time. Then we report the average results of the three runs. For some experiments, such as using product and pseudo queries as training data, we do not use queries as training data, but we still run the test over each fold and report the average results.

In the discussions of the experiments, when we claim one classifier is better than another, we use a pair-difference t-test over the results of the cross-validation to verify the significance of the difference. By convention, we say that a difference between means at the 95% level is "significant"; a difference at 99% level is "highly significant".

4.4 Results and Analysis

4.4.1 Comparison between Product and Pseudo Query as Training Data

We first compare two kinds of training data, one is product data as described in 3.3.1 and the other is "Pseudo Queries" (denoted by PseudoQry) given in 3.3.2. We randomly sampled 1,500,000 and 500,000 product names from the labeled product data in MSN Shopping, for the "Level-1" and "Computing" tasks, respectively. For each of these product names, we can get a set of pseudo queries using the translation model and keep those whose confidence is larger than 0.15 (detailed study of the threshold is given in Section 4.4.6). As we can see from Table 3, classifiers trained over pseudo queries can improve the classification performance, especially for "Computing" classification, where the relative improvement is more than 11%. Using the t-test, we find that the improvement is "highly significant".

4.4.2 Effect of Query Enrichment through Query Similarity when Changing the Size of Training Data

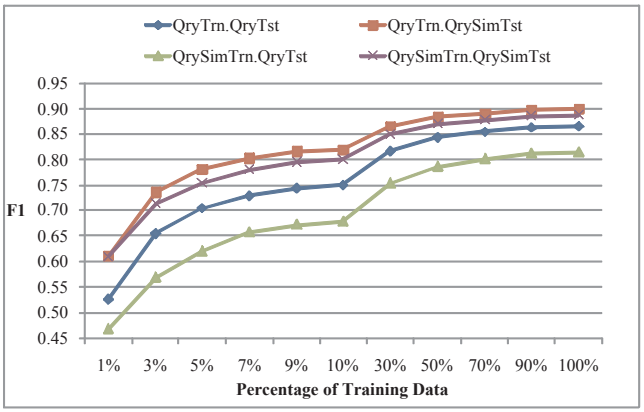
In this section we study the effect of enriching queries through their similar queries as introduced in Section 3.2.2 when changing the size of the training data. For each query, we collect all the queries with which the similarity is larger than 0.001. Detailed study of the threshold is given in Section 4.4.6. Here, we study four combinations, (1) using original queries for training and test (denoted by QryTrn.QryTst); (2) using original queries for training and enriched queries for test (denoted by QryTrn.QrySimTst); (3) using enriched queries for training and original queries for test (denoted by QrySimTrn.QryTst); (4) using enriched queries for training and test. Figures 2(a) and 2(b) shows the performance of the four combinations for "Computing" classification and "Level-1" classification respectively. In Figure 2(a) and 2(b), the vertical axis is micro-F1 while the horizontal axis is the percent of training data we randomly sampled from the two folders used for training in each run (refer to Section 4.3 for more information about splitting training and test data). As shown in the figures, the performance of all the combinations becomes better when we increase the size of training data, which becomes relatively stable when we use 70% training data or more. Another observation is that for both "Computing" and "Level-1", we can get the order of the four combinations based on their performance as follows: QryTrn.QrySimTst > QrySimTrn.QrySimTst > QryTrn.QryTst > QrySimTrn.QryTst. Using the t-test, we find that the difference between each pair of combinations is "highly significant". We can also find that the fewer the training data, the larger the gap between these combinations. Specifically, the relative improvement of "QryTrn.QrySimTst" against "QryTrn.QryTst" is 3.9% and 3.5% for "Computing" and "Level-1", respectively, when we use all the training data. These numbers become 9.2% and 5.8% when we use 10% of the training data. We draw the figures in terms of macro-F1 too and come to the same conclusions, so we did not show those figures for simplicity.

Queries enriched through similar queries have a much richer representation, which can reduce the data sparseness and increase the classification performance. Therefore, it is easy to understand why "QrySimTrn.QrySimTst" outperforms "QryTrn.QryTst". Similarly, we can explain why "QryTrn.QrySimTst" is better than "QryTrn.QryTst". However, why "QrySimTrn.QryTst" performs worst is not straightforward. One explanation is that the classifier trained over the enriched queries lies in a larger feature space, which cannot distinguish the queries represented by their original feature space.

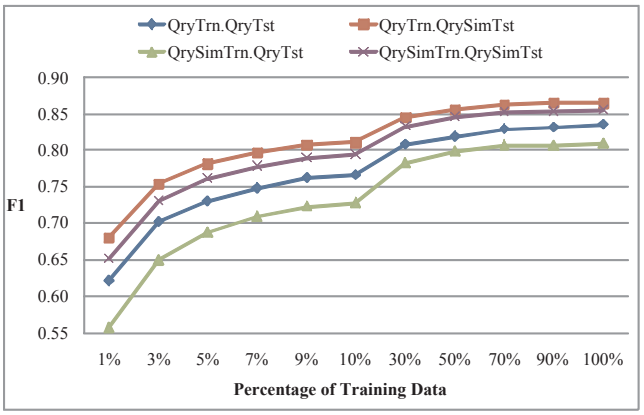
4.4.3 Effect of Query Enrichment through Search Snippets when Changing the Size of Training Data

Similar to Section 4.4.2, we study the effect of query enrichment through search snippets. Here, for each query, we enrich it by the top 10 returned snippets. The impact of the number of returned snippets is studied in Section 4.4.6. Figures 3(a) and 3(b) show the results for "Computing" and "Level-1" respectively. All the notations have the same meanings as in Section 4.4.2, except that "QrySnpt" refers to the enriched queries through search snippets.

Similar to the figures as shown in Section 4.4.2, we order the four combinations based on their performance as follows: QrySnptTrn.QrySnptTst > QryTrn.QrySnptTst > QryTrn.QryTst > QrySnptTrn.QryTst. Using the t-test, we find that the difference between each pair of combinations (except the pair of QryTrn.QryTst and QrySnptTrn.QryTst for "Computing") is at least "significant". We can see that the gaps between these combinations (except QrySnpTrn.QryTst) become larger when we use fewer training data. Specifically, the relative improvement of "QrySnptTrn.QrySnptTst" against "QryTrn.QryTst" is 6.6% and 7.5% for "Comput-

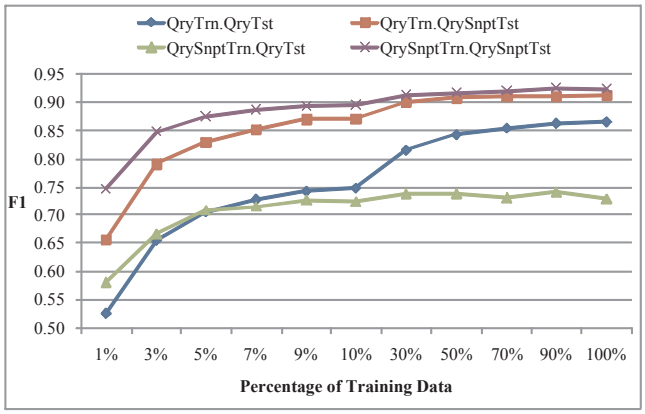


(a) "Computing"

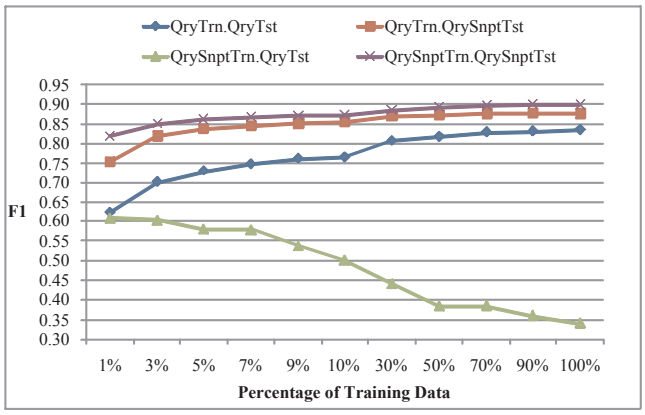


(b) "Level-1"

Figure 2: Comparison between Original Queries and Query Expansion through Similar Queries



(a) "Computing"



(b) "Level-1"

Figure 3: Comparison between Original Queries and Query Expansion through Search Snippets

ing" and "Level-1", respectively, when we use all the training data. These numbers become 19.4% and 13.9% when we use 10% of the training data.

With the similar reasons as given in Section 4.4.2, we can explain why QryTrn.QrySnptTst and QrySnptTrn.QrySnptTst are better than "QryTrn.QryTst" and why "QrySnptTrn.QryTst" performs worst. One obvious observation from Figure 3(b) is that "QrySnptTrn.QryTst" becomes significantly worse when we increase the size of training data. This is because the feature space of snippets-enriched queries may become more different from the feature space of the original queries when we use more training data. Therefore, the classifier trained over the enriched queries are tuned to fit the enriched feature space, which cannot accurately classify the queries represented in the original feature space. Comparing with the performance of "QrySimTrn.QryTst" in Section 4.4.2, we may see that the web search snippet based query enrichment is more ambitious, which can cause relatively large drift from the original queries.

As indicated in Figures 2(a) and 2(a), for both "Computing" and "Level-1" tasks, "QryTrn.QrySimTst" performs better than "QrySimTrn.QrySimTst". Contradictorily , in Figure 3(a) and 3(b), we find that "QrySnptTrn.QrySnptTst" performs better than "QryTrn.QrySnptTst" for both "Computing" and "Level-1". Thus, we cannot simply conclude which classifiers (those trained from the original queries, or those trained over enriched queries) are better when they are used to classify enriched queries. The reason we mentioned above, that is, search-snippet based query enrichment method is more ambitious than the similar-query based method, can partially explain the difference. However, the deep reason is under investigation and we will leave it to our future work. But whatever the reason is, we can clearly see that properly using query enrichment can significantly improve the classification performance, especially when we do not have enough training data.

4.4.4 *Comparison Among All Combinations of Training Data and Query Enrichment Methods*

In Sections 4.4.1, 4.4.2 and 4.4.3, we have studied the effect of different kinds of training data and query enrichment methods. We can see that although using pseudo queries as the training data can improve the classification performance compared to using product names, it is not as good as using automatically labeled queries, either in their original form, or in their enriched form. We also find that using the enriched queries in a proper way can significantly improve the classification performance, especially when the size of training data is small. In this section, we put all the possible combinations of training data and query enrichment methods together and further compare them in detail. In order to see a clear picture of the advantages of each combination, we randomly select 50% training data to train the classifier when the training data is the original query, and their enrichment, either by through similar queries, or search snippets.

Tables 4 and 5 show the comparison results over "Com-

puting" and "Level-1" respectively, which include the average micro-F1 and macro-F1 over three folds, as well as their standard deviation values. The rows in these tables indicate which kind of training data is used to training classifiers. The columns indicate which method is used to enrich the queries for test. Besides the observations we discussed in previous sections, we can see several more interesting ones. Firstly, the classifiers trained from product names achieve much better performance when classifying enriched queries compared to the original queries. They work especially well for the queries enriched through search snippets. Secondly, classifiers trained from pseudo queries outperform those trained from product names when classifying both the original queries, and those enriched by similar queries. However, their performance drops significantly when they are used to classify the queries enriched by search snippets. Thirdly, the classifiers trained over the enriched queries by similar queries can achieve better performance when they are used to classify the enriched queries through search snippets. However, on the other side, we see dramatic drops in performance when we use the classifiers trained over search-snippet based enriched queries to classify the queries which are enriched by similar queries. Notice that both similar-query based enrichment and pseudo queries share similar natures of the original queries, while they are all quite different from the search-snippet based enrichment, it is easy to explain the above observations using the reasons presented in Sections 4.4.1, 4.4.2 and 4.4.3.

4.4.5 Hybrid Classifiers

As discussed in Section 4.4.4, "QrySnptTrn.QrySnptTst" and "QryTrn.QrySnptTst" perform better than "QryTrn.QryTst" for both "Computing" and "Level-1". However, for "QrySnptTrn.QrySnptTst" and "QryTrn.QrySnptTst", we have to collect search snippets for all the queries as a first step for the classification, which requires lots of resources and take a long time. Therefore, we want to find a hybrid classifier, which can reduce the efforts of collecting search snippets while maintain high performance. Recall the Equation (2), given a query q, a LR classifier can calculate the probability of each possible label and then assign the label with highest probability to q. The probability can also be called as confidence of the classifier to make the classification decision. Intuitively, the classification accuracy is higher when the confidence is larger. However, if the classifier only classify the queries with high confidence, the number of the queries which can be classified (or called *coverage* as defined in Section 4.3) becomes smaller.

In this section, we first train a LR classifier using the original queries, and then use it to classify the original queries, or enriched queries by search snippets (top 10 search snippets are used as in the above experiments). The top four rows (corresponding to QryTst and QrySnptTst) in Tables 6 and 7 illustrate the trade-off between accuracy and coverage when we change the confidence threshold from 0.95 to 0.10. Compared to QrySnptTst, the classifier can achieve much higher accuracy while lower coverage on QryTst when the confidence threshold is higher. By decreasing the threshold, we can see that the accuracy on QryTst drops quickly while the drop on the QrySnptTst is relatively slow. Therefore, we should trust the classification results of using the original queries more when the confidence is high. Thus, a straightforward method to get a hybrid classifier is to find a threshold so that we output the classification results of using the original queries when the classifier's confidence is larger than that threshold; otherwise, we output the classification results of using the enriched queries through search snippets. With this method, it is clear that we can assign labels to all the queries, while we just need to retrieve the search snippets for some of the queries. The last two rows in Tables 6 and 7 validate the proposed hybrid classifier. We can see that by setting the threshold as 0.70, we can classify more than 70% queries directly without retrieving the search snippets and the final accuracy is even higher than adopting QrySnptTst, for which we have to collect the search snippets for all test queries.

4.4.6 Parameter Tuning

In the above experiments, we have empirically set the values of several parameters. In this section, we will study the impact of the parameters on the classification performance.

One parameter is for enriching queries through their similar queries. Clearly, we have to use a similarity threshold to control what queries can be used to enrich a target queries. If the threshold is too small, we may include some non-similar queries. On the other side, if the threshold is too large, the queries we can get for a query may be too few. Both cases will result in bad classification performance. For simplicity, we only study the impact of the threshold for the "QrySimTrn.QrySimTst" and "QryTrn.QrySimTst" on the "Level-1" task where we use 50% training data. The results are shown in Table 8. The first row shows the average number of queries we can get for each query given a certain threshold. It is easy to see that the larger the threshold, the smaller the average number. We can also find that when the threshold changes from 0.001 to 0.01, the performance changes slightly. After that, the performance drops quickly. In the above experiments, we set the threshold as 0.001.

Another parameter is for query enrichment through Web snippets, which decides how many returned snippets we use to enrich a query. Here, we show the impact of the threshold on "QrySnptTrn.QrySnptTst" and "QryTrn.QrySnptTst" for the "Computing" task, where we use 50% training data. The results are shown in Table 9. From the table, we can see that the classification performance will change but not too much by varying the number of snippets. Although using 40 or 50 snippets will lead to the best performance, we use only 10 snippets in the above experiments to speed them up.

The final parameter we want to study is the threshold used to generate pseudo queries. As shown in Section 3.3.2, given a product a name, we can generate a set of pseudo queries together with the confidences. We can use a threshold to keep the queries with high confidence and remove the noise. We conduct a group of experiments for "Computing" and "Level-1" by varying the threshold from 0.05 to 0.30. However, due to the space limit, we do not show the detailed results. From the experiments, we find that the classification performance becomes better first and then worse in most cases. Therefore, in the above experiments,we choose a middle threshold, which is 0.15.

4.4.7 Evaluation over Manually Labeled Data

All the above experiments are conducted over the automatically labeled queries. Although we have shown in 4.1 that the automatically obtained labels are quite consistent with manually generated labels, we still want to see how

Table 4: Compare All Combinations of Training Data and Query Enrichment Methods over "Computing"

	Macro-F1			Micro-F1		
	Qry	QrySim	QrySnpt	Qry	QrySim	QrySnpt
Product	0.591±0.026	0.750±0.017	0.825±0.013	0.549±0.007	0.749±0.005	0.845±0.004
PseudoQry	0.647±0.017	0.765±0.007	0.747±0.004	0.611±0.004	0.771±0.004	0.727±0.002
Qry	0.797±0.025	0.845±0.020	0.882±0.015	0.843±0.004	0.884±0.002	0.908±0.002
QrySim	0.745±0.024	0.834±0.019	0.860±0.025	0.786±0.006	0.872±0.002	0.890±0.008
QrySnpt	0.717±0.021	0.793±0.023	**0.901**±0.007	0.740±0.010	0.824±0.012	**0.917**±0.005

Table 5: Compare All Combinations of Training Data and Query Enrichment Methods over "Level-1"

	Macro-F1			Micro-F1		
	Qry	QrySim	QrySnpt	Qry	QrySim	QrySnpt
Product	0.539±0.004	0.620±0.001	0.679±0.000	0.682±0.001	0.742±0.001	0.747±0.001
PseudoQry	0.554±0.006	0.599±0.003	0.650±0.001	0.700±0.001	0.731±0.001	0.742±0.000
Qry	0.699±0.008	0.750±0.001	0.761±0.005	0.818±0.001	0.856±0.000	0.872±0.002
QrySim	0.657±0.002	0.739±0.001	0.751±0.005	0.798±0.002	0.846±0.001	0.859±0.003
QrySnpt	0.131±0.046	0.519±0.027	**0.802**±0.003	0.383±0.040	0.598±0.028	**0.892**±0.002

Table 6: Performance of Hybrid Classifier for "Computing"

	Confidence	0.95	0.90	0.85	0.80	0.75	0.70	0.60	0.50	0.40	0.30	0.20	0.10
QryTst	Accuracy	0.989	0.985	0.981	0.977	0.970	0.966	0.953	0.941	0.915	0.904	0.891	0.866
	Coverage	0.474	0.576	0.635	0.679	0.713	0.741	0.793	0.839	0.894	0.925	0.960	1.000
QrySnptTst	Accuracy	0.920	0.918	0.917	0.916	0.916	0.915	0.914	0.913	0.912	0.912	0.912	0.912
	Coverage	0.984	0.988	0.990	0.992	0.994	0.995	0.998	0.999	1.000	1.000	1.000	1.000
Hybrid	Accuracy	0.917	0.920	0.921	0.922	0.923	**0.924**	0.921	0.915	0.904	0.898	0.889	0.866
	Coverage	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 7: Performance of Hybrid Classifier for "Level-1"

	Confidence	0.95	0.90	0.85	0.80	0.75	0.70	0.60	0.50	0.40	0.30	0.20	0.10
QryTst	Accuracy	0.987	0.981	0.975	0.969	0.964	0.958	0.944	0.928	0.911	0.894	0.878	0.835
	Coverage	0.481	0.575	0.632	0.673	0.706	0.734	0.783	0.830	0.871	0.907	0.939	1.000
QrySnptTst	Accuracy	0.887	0.885	0.883	0.882	0.881	0.880	0.878	0.876	0.876	0.876	0.876	0.876
	Coverage	0.980	0.985	0.988	0.990	0.992	0.994	0.996	0.999	1.000	1.000	1.000	1.000
Hybrid	Accuracy	0.880	0.882	0.884	0.885	0.886	**0.886**	0.886	0.883	0.878	0.871	0.864	0.835
	Coverage	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 8: Impact of Threshold in Query Enrichment through Query Similarity for Level-1

		0.001	0.005	0.01	0.03	0.05	0.07	0.09
Average Number of Similar Queries		5.565	5.030	3.796	1.995	1.553	1.354	1.246
QrySimTrn.QrySimTst	Macro-F1	0.739	0.740	0.736	0.738	0.731	0.729	0.728
	Micro-F1	0.846	0.846	0.844	0.844	0.841	0.837	0.837
QryTrn.QrySimTst	Macro-F1	0.750	0.745	0.746	0.741	0.733	0.726	0.723
	Micro-F1	0.856	0.855	0.853	0.847	0.842	0.839	0.835

Table 9: Impact of Number of Snippets in Query Enrichment through Search Snippets for "Computing"

		10	20	30	40	50
QrySnptTrn.QrySnptTst	Macro-F1	0.901	0.891	0.894	0.896	0.903
	Micro-F1	0.917	0.920	0.923	0.925	0.927
QryTrn.QrySnptTst	Macro-F1	0.882	0.883	0.887	0.885	0.880
	Micro-F1	0.908	0.911	0.912	0.912	0.911

Table 10: Results on Labeled Data

	Labeler1	Labeler2	Labeler3	Majority
Labeler1		0.926	0.943	0.952
Labeler2			0.965	0.961
Labeler3				0.973
Predicted	0.935	0.939	0.956	0.948

the classifiers trained from the automatically labeled queries work over the queries which cannot be automatically labeled using the method given in Section 3.3.3. Therefore, we randomly select 2,000 queries which cannot be automatically labeled and let the three labelers label them manually with the 26 categories in the "Level-1" task. After that, we use the labeled queries to evaluate the hybrid classifier discussed in Section 4.4.5. Note that here, we use all the automatically labeled queries to train the hybrid classifier. The evaluation results are given in Table 10. Table 10 also shows the consistency among the three labelers and their consistency with the voted labels ("majority" as defined in Section 4.1). We can see the consistency among the labelers for the 2,000 queries is a little lower than their consistency on the 6,000 queries introduced in Section 4.1. One of the reasons for the difference is that the 2,000 may be very new, or not so frequent, or more ambiguous compared to the 6,000 queries, which is also the reason why they cannot be labeled automatically. The performance of the classifier(as denoted by "Predicted") over the 2,000 queries is pretty good, which is even better than those shown in Table 7. One of the reasons is that the hybrid classifier used in this experiments is trained over more training data. Another reason is that the labels provided by the labelers are a little more accurate compared to the automatically generated labels, which will not misjudge the classifier's correct predictions.

5. CONCLUSION

This paper presented a comprehensive study of query classification by taking product query classification as an example. In this paper, we proposed to use labeled product data (corresponding to Web pages in general query classification) to train classifiers and then further improve the classifier by introducing pseudo queries, which are translated from labeled product names. We also put forward two query enrichment methods to better represent the queries, either through similar queries estimated from click-though log data, or through Web search snippets. After that, we systematically compared different combinations of training data and query representation methods over a large query data set and found some interesting observations. Most of these observations are not investigated before due to the limit of large scale data sets. Based on these observations, we worked out a hybrid classifier, which can achieve high classification accuracy without collecting search snippets for most of the test queries.

In the future, we will manually label more test data to further validate the hybrid classifier. We will also build some hierarchical classifiers and study the contributions of different training data and different query enrichment methods. Finally, we will investigate how the product query classification methods can improve the product search relevance.

6. REFERENCES

[1] 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. *TOIS*, 25(2):9, 2007.

[2] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[3] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *SIGIR '07*, 2007.

[4] S.-L. Chuang, H.-T. Pu, and L.-F. Chien. Automatic subject categorization of query terms for filtering sensitive queries in web multimedia search. In *PCM '01*, pages 825–830, 2001.

[5] J. M. Crego, A. de Gispert, and J. B. Marino. The TALP ngram-based SMT system for IWSLT'05. In *Proc. International Workshop Spoken Language Translation*, Pittsburgh, 2005.

[6] H. K. Dai, L. Zhao, Z. Nie, J.-R. Wen, L. Wang, and Y. Li. Detecting online commercial intention (oci). In *WWW '06*, pages 829–837, 2006.

[7] L. Gravano, V. Hatzivassiloglou, and R. Lichtenstein. Categorizing web queries according to geographical locality. In *CIKM '03*, pages 325–333, 2003.

[8] B. J. Jansen. The effect of query complexity on web searching results. *Information Research*, 6(1), 2000.

[9] I.-H. Kang and G. Kim. Query type classification for web document retrieval. In *SIGIR '03*, pages 64–71, 2003.

[10] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *WWW '05*, pages 391–400, New York, NY, USA, 2005. ACM Press.

[11] X. Li, Y.-C. Ju, G. Zweig, and A. Acero. Language modeling for voice search: a machine translation approach. In *Proc. International Conf. on Audio, Speech and Signal Processing*, 2008.

[12] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *SIGIR '08*, 2008.

[13] R. Manmatha, A. Feng, and J. Allan. A critical examination of tdt's cost function. In *SIGIR '02*, pages 403–404, New York, NY, USA, 2002. ACM Press.

[14] C. Manning and H. SchÜTze. *Foundations of Statistical Natural Language Processing*. MIT Press., Cambridge, Ma, 1999.

[15] C. D. Manning, P. Raghavan, and H. SchÜZe. *Introduction To Information Retrieval*. Cambridge University Press, 2008.

[16] B. Martins, M. J. Silva, S. Freitas, and A. P. Afonso. Handling locations in search engine queries. In *GIR '06*, 2006.

[17] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM '08*, October 2008.

[18] K. Nigam, J. Lafferty, and A. K. Mccallum. Using maximum entropy for text classification. pages 61–67, 1999.

[19] H.-T. Pu, S.-L. Chuang, and C. Yang. Subject categorization of query terms for exploring web users' search interests. *JASIST*, 53(8):617–630, 2002.

[20] C. V. Rijsbergen. *Information Retrieval*. Butterworths, London, second edition edition, 1979.

[21] D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW '04*, pages 13–19, New York, NY, USA, 2004. ACM.

[22] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *NAACL '03*, pages 134–141, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[23] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Query enrichment for web-query classification. *TOIS*, 24(3):320–352, 2006.

[24] A. Spink, D. Wolfram, M. B. J. Jansen, and T. Saracevic. Searching the web: the public and their queries. *JASIST*, 52(3):226–234, 2001.

[25] L. Wang, C. Wang, X. Xie, J. Forman, Y. Lu, W.-Y. Ma, and Y. Li. Detecting dominant locations from search queries. In *SIGIR '05*, pages 424–431, 2005.