Categorizing Web Queries According to Geographical Locality

Luis Gravano
Columbia University
gravano@cs.columbia.edu

Vasileios Hatzivassiloglou
Columbia University
vh@cs.columbia.edu

Richard Lichtenstein
Harvard University
lichtens@fas.harvard.edu

ABSTRACT

Web pages (and resources, in general) can be characterized according to their *geographical locality*. For example, a web page with general information about wildflowers could be considered a *global* page, likely to be of interest to a geographically broad audience. In contrast, a web page with listings on houses for sale in a specific city could be regarded as a *local* page, likely to be of interest only to an audience in a relatively narrow region. Similarly, some search engine queries (implicitly) target global pages, while other queries are after local pages. For example, the best results for query [wildflowers] are probably *global* pages about wildflowers such as the one discussed above. However, *local* pages that are relevant to, say, San Francisco are likely to be good matches for a query [houses for sale] that was issued by a San Francisco resident or by somebody moving to that city. Unfortunately, search engines do not analyze the geographical locality of queries and users, and hence often produce sub-optimal results. Thus query [wildflowers] might return pages that discuss wildflowers in specific U.S. states (and not general information about wildflowers), while query [houses for sale] might return pages with real estate listings for locations other than that of interest to the person who issued the query. Deciding whether an unseen query should produce mostly local or global pages—without placing this burden on the search engine users—is an important and challenging problem, because queries are often ambiguous or underspecify the information they are after. In this paper, we address this problem by first defining how to categorize queries according to their (often implicit) geographical locality. We then introduce several alternatives for automatically and efficiently categorizing queries in our scheme, using a variety of state-of-the-art machine learning tools. We report a thorough evaluation of our classifiers using a large sample of queries from a real web search engine, and conclude by discussing how our query categorization approach can help improve query result quality.

CIKM'03, November 3–8, 2003, New Orleans, Louisiana, USA.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*query formulation, search process*

General Terms

Algorithms, Experimentation, Human Factors

Keywords

web search; information retrieval; search engines; query classification; query modification

1. INTRODUCTION

Web pages (and resources, in general) can be characterized according to their *geographical locality*. For example, a web page with general information about wildflowers could be considered a *global* page, likely to be of interest to a geographically broad audience. In contrast, a web page with listings on houses for sale in a specific city could be regarded as a *local* page, likely to be of interest only to an audience in a relatively narrow region. Earlier research [9] has addressed the problem of automatically computing the "geographical scope" of web resources.

Often search engine queries (implicitly) target global web pages, while other queries are after local pages. For example, the best results for query [wildflowers] are probably *global* pages about wildflowers discussing what types of climates wildflowers grow in, where wildflowers can be purchased, or what types of wildflower species exist. In contrast, *local* pages that are relevant to, say, San Francisco are likely to be good matches for a query [houses for sale] that was issued by a San Francisco resident, or by somebody moving to San Francisco, even if "San Francisco" is not mentioned in the query. The user's intent when submitting a query may not be always easy to determine, but if underspecified queries such as [houses for sale] can be detected, they can be subsequently modified by adding the most likely target geographical location or by getting further user input to customize the results.

Unfortunately, search engines do not analyze the geographical locality of queries and users, and hence often produce sub-optimal results, even if these results are on-topic and reasonably "popular" or "authoritative." Thus query [wildflowers] might return pages that discuss wildflowers in specific U.S. states (and not general information about wildflowers). In fact, as of the writing of this paper, the first 10 results that Google provides for this query include 5 pages each of which discusses wildflowers in only one U.S.

state (e.g., "Texas Wildflowers"). Similarly, the top 10 results that Google returns for query [houses for sale] include real estate pages for Tuscany, United Kingdom, and New Zealand. These pages are likely to be irrelevant to, say, somebody interested in San Francisco real estate who types such an underspecified query.

Deciding whether a query posed by a regular search engine user should produce mostly local or global pages is an important and challenging problem, because queries are often ambiguous or underspecify the information they are after, as in the examples above. By identifying that, say, query [wildflowers] is likely after "global" information, a search engine could rank the results for this query so that state-specific pages do not appear among the top matches. By identifying that, say, query [houses for sale] is likely after "local" information, a search engine could filter out pages whose geographical locality is not appropriate for the user who issued the query. Note that deciding *which location* is of interest to a user who wrote an underspecified query such as [houses for sale] is an orthogonal, important issue that we do not address in this paper. Our focus is on identifying that such a query is after "local" pages in nature, and should therefore be treated differently by a search engine than queries that are after "global" pages. By knowing that a user query is after local information, a search engine might choose to privilege pages whose geographical locality coincides with that of the user's or, alternatively, attempt to obtain further input from the user on what location is of interest.

In this paper, we first define how to categorize user queries according to their (often implicit) geographical locality. We then introduce several alternatives for automatically and efficiently classifying queries according to their locality, using a variety of state-of-the-art machine learning tools. We report a thorough evaluation of our classifiers using a large sample of queries from a real web search engine query log. Finally, we discuss how our query categorization approach can help improve query result quality. The specific contributions of this paper are as follows:

- A discussion on how to categorize user queries according to their geographical locality, based on a careful analysis of a large query log from the Excite web site (Section 3).
- A feature representation for queries; we derive the feature representation of a query from the results produced for the query by a web search engine such as Google (Section 4.1).
- A variety of automatic query classification strategies that use our feature representation for queries (Section 4.2).
- A large-scale experimental evaluation of our strategies over real search engine queries (Section 5).
- Preliminary query reformulation and page re-ranking strategies that exploit our query classification techniques to improve query result quality (Section 6).

2. RELATED WORK

Traditional information-retrieval research has studied how to best answer keyword-based queries over collections of text documents [18]. These collections are typically assumed to be relatively uniform in terms of, say, their quality and scope. With the advent of the web, researchers are studying other "dimensions" to the data that help separate useful resources from less-useful ones in an extremely heterogeneous environment like the web. Notably, the Google search engine [4] and the HITS algorithm [7, 13] estimate the "importance" of web pages by analyzing the hyperlinks that point to them, thus capturing an additional dimension to the web data, namely how important or authoritative the pages are.

Ding et al. [9] extract yet another crucial dimension of the web data, namely the geographical scope of web pages. For example, the Stanford Daily newspaper has a geographical scope that consists of the city of Palo Alto (where Stanford University is located), while the New York Times newspaper has a geographical scope that includes the entire U.S. To compute the geographical scope of a web page, Ding et al. propose two complementary strategies: a technique based on the geographical distribution of HTML links to the page, and a technique based on the distribution of geographical references in the text of the page. Ding et al. report on a search-engine prototype that simply filters out from the results for a user query any pages not in the geographical scope of the user. This technique does not attempt to determine whether a query is best answered with "global" or "local" pages, which is the focus of our paper. Ding et al. built on the work by Buyukkokten et al. [6], who discussed how to map a web site (e.g., `http://www-db.stanford.edu`) to a geographical location (e.g., Palo Alto) and presented a tool to display the geographical origin of the HTML links to a given web page. This tool then helps visualize the geographical scope of web pages [6].

A few commercial web sites *manually* classify web resources by their location, or keep directory information that lists where each company or web site is located. The NorthernLight search engine[1] extracts addresses from web pages, letting users narrow their searches to specific geographical regions (e.g., to pages "originated" within a five-mile radius of a given zip code). Users benefit from this information because they can further filter their query results. McCurley [14] presented a variety of approaches for recognizing geographical references on web pages, together with a navigational tool to browse pages by geographical proximity and their spatial context. (Please refer to [16] for additional references.) None of these techniques addresses our focus problem in this paper: automatically determining the geographical locality associated with a given, unmodified search engine query.

3. DEFINING GEOGRAPHICAL LOCALITY

As discussed above, queries posed to a web search engine can be regarded as *local*, if their best matches are likely to be "local" pages, or as *global*, if their best matches are likely to be "global" pages. In an attempt to make this distinction more concrete, we now discuss several examples of local and global queries.

Global queries often do not include a location name, as is the case for query [Perl scripting]. A user issuing this query is probably after tutorials about the Perl language, and hence pages on the topic with a restricted geographi-

[1] `http://www.northernlight.com/`

cal scope are less desirable than global pages. Other global queries do not mention a location explicitly either, but are topically associated with one particular location. An example of such a query is [Elgin marbles], which is topically associated with the city of Athens. We consider these queries as global, since their best matches are broad, global pages, not localized pages with a limited geographical scope. Interestingly, global queries sometimes do include a location name. For example, a query might be just a location name (e.g., [Galapagos Islands]) or a request for concrete information about a location (e.g., [Boston area codes]). General resources about the location (e.g., tourist guides) are arguably to be preferred for such queries, which are hence regarded as global. Other global queries include locations that are strongly associated topic-wise with the rest of the query. Query [Woody Allen NYC] is an example of such a query. The location mentioned in this query (i.e., "NYC," for "New York City") is not used to restrict query results to pages of interest to New York residents, but rather expresses a topic specification. Query [Ansel Adams Yosemite] is another example: photographer Ansel Adams took a famous series of photographs in Yosemite.

Local queries often include a location name, as is the case for query [Wisconsin Christmas tree producers association]. The location mentioned in this query (i.e., "Wisconsin") is used to "localize" the query results. Query [houses for sale New York City] is a related example. Other local queries do not include a location name, but still implicitly request "localized" results. Query [houses for sale] is an example of such a query. These queries tend to be underspecified, but are still asked by (presumably naïve) search engine users.

We conducted a thorough examination of a large number (over 1,200) of real search engine queries. Most queries that we encountered can be cleanly categorized as being either global or local. However, other queries are inherently ambiguous, and their correct category is impossible to determine without further information on the user intent behind them. For example, query [New York pizza] could be construed as a local query if it is, say, after pizza delivery web sites for the New York area. In contrast, the same query could be regarded as a global query if the user who issues it wants to learn about the characteristics of New York-style pizza.

4. USING CLASSIFIERS TO DETERMINE LOCALITY

We earlier established that queries are associated with local or global status, which influences the kind of results that are desirable. Since current search engines do not directly take into account geographical information, for certain types of queries they produce a large number of on-topic but unwanted results, as in the [houses for sale] example discussed earlier. In this section, we discuss automatic methods that can determine, given a query, whether the query is a local or global one. To build the two-class classifier, we experimented with several state-of-the-art classification techniques, using widely available implementations for each. We describe below the features used in the classification, how we extract them from web pages, and the classifiers with which we experimented.

4.1 Classification Features

Web queries, which we treat in this paper as ordered bags of words with no other structure, are typically fairly short. In the collection of 2,477,283 real queries that we used in our experiments (Section 5.1), 84.9% were five words long or shorter. Because few words are available per query, basing the classification directly on the words in the query may lead to severe sparse data problems. Even more importantly, some of the characteristics that make a query local or global are not directly observable in the query itself, but rather in the results returned. For example, a query that returns results that contain few references to geographical locations is likely to be global, while a query that returns results spread uniformly over many locations without including a significant percentage of results with no locations is likely to be local.

For these reasons, we base our classification on a sample of results actually returned for a given query rather than the words in the query itself. By observing distributional characteristics in the unmodified results, the classifier can infer the type of the query (global or local) so that the results can be appropriately filtered or re-ordered, or the query modified. In a way the approach is similar in spirit to query expansion techniques that rely on *pseudo-relevance feedback* [5]. In our experiments, we use Google (via the Google API[2]) to obtain the top 50 web pages that match the query. For simplicity, we limited our search to HTML pages, skipping over non-HTML documents. We chose Google because it represents state-of-the-art web search technology and offers a published interface for submitting large numbers of queries.

We represent the web pages returned by Google as text documents. This conversion is achieved by using the LYNX HTML browser with the `-dump` option. We base our classification features on measures of frequency and dispersion of location names in these text files. For this purpose, we have constructed a database of 1,605 location names by concatenating lists of all country names[3], of the capitals of these countries[4], of the fifty U.S. states, and of all cities in the United States with more than 25,000 people[5]. We then compare the words in each text document with the database of location names, and output any matching entries and their count per document. This matching is case insensitive, because we found capitalization of location names in web pages to be erratic. Note that we do not attempt to disambiguate words that match location names but also have other senses (e.g., "China"), as this is a hard problem in natural language analysis; instead, we count such words as locations. An alternative approach that would detect and disambiguate location names would be to use a named-entity tagger. We experimented with a well-known third-party named-entity tagger, but we encountered a very high error rate because of the noise often introduced in web pages.

Our classification features combine these location counts in various ways. For each query, we measure the average

[2] `http://www.google.com/apis`

[3] Obtained from the United Nations, `http://www.un.org/Overview/unmember.html`.

[4] Obtained from the CIA World Factbook, `http://www.capitals.com/`.

[5] Obtained from the U.S. Census Bureau (2000 census figures), `http://www.census.gov/prod/2002pubs/00ccdb/cc00_tabC1.pdf`.

(per returned web page) number of location words in the retrieved results. We count the average frequency of location words at different levels of detail (country, state, city), as well as the average of the aggregate total for all locations. We obtain these frequencies for both the total count (*tokens*) and the unique location words in each page (*types*), as it is possible that a few locations would be repeated many times across the results, indicating a global query, or that many locations would be repeated few times each, indicating a local query. We also consider the total number of unique locations across all the returned documents taken together, divided by the number of retrieved documents. For the average token frequencies of city, state, and country locations we also calculate the minimum and maximum across the set of returned web pages. To account for the hierarchical nature of location information, we calculate an alternative frequency for states where we include in the count for each state the counts for all cities from that state that were found in that text; this allows us to group together location information for cities in the same state. We also include some distributional measures, namely the fraction of the pages that include at least one location of any kind, the percentage of words that match locations across all pages, and the standard deviation of the total per page location count. Finally, we add to our list of features the total number of words in all of the returned documents, to explore any effect the local/global distinction may have on the size of the returned documents. These calculations provide for 20 distinct features that are passed on to the classifier.[6] The core data needed to produce these 20 query features (i.e., the locations mentioned in each web page) could be efficiently computed by a search engine such as Google at page-indexing time. Then, the final feature computation could be quickly performed at query time using this core data.

4.2 Classification Methods

We initially trained a classifier using RIPPER [8], which constructs a rule-based classifier in an incremental manner. The algorithm creates an initial set of very specific rules based on the data, similar to the way in which decision trees are generated. The rules are then pruned iteratively to eliminate the ones that do not seem to be valid for a large enough subset of the training data, so as to prevent overfitting.

Although RIPPER provides a robust classifier with high accuracy and transparency (a human can easily examine the produced rules), it outputs binary "local"–"global" decisions. In many cases, it is preferable to obtain a measure of confidence in the result or an estimate of the probability that the assigned class is the correct one. To add this capability to our classifier, we experimented with *logistic regression* [19]. Logistic, or *log-linear*, regression models a binary output variable (the class) as a function of a weighted sum of several input variables (the classification features). Conceptually, a linear predictor η is first fitted over the training data in a manner identical to regular regression analysis, i.e.,

$$\eta = w_0 + \sum_{i=1}^{k} w_i \cdot F_i$$

where F_i is the i-th feature and w_i is the weight assigned to that feature during training. Subsequently, η is transformed to the final response, C, via the *logistic transformation*

$$C = \frac{e^{\eta}}{1 + e^{\eta}}$$

which guarantees that C is between 0 and 1. Each of the endpoints of the interval $(0, 1)$ is associated with one of the classes, and C gives the probability that the correct class is the one associated with "1". In practice, the calculations are not performed as a separate regression and transformation, but rather as a series of successive regressions of transformed variables via the *iterative reweighted least squares* algorithm [1].[7] We used the implementation of log-linear regression provided in the R statistical package.[8]

Another desideratum for our classifier is its ability to support different costs for the two possible kinds of errors (misclassifying local queries versus misclassifying global queries). Which kind of error is the most important may vary for different settings; for our search modification application, we consider the misclassification of global queries as local ones a more serious error. This is because during our subsequent modification of the returned results (Section 6), we reorder the results for some of the queries that we consider global, but we modify the original queries for some of the queries classified as local, returning potentially very different results. Consequently, the results can change more significantly for a query classified as local, and the potential for error is higher when a global query is labeled local than the other way around.

Both RIPPER and log-linear regression can incorporate different costs for each type of error. We experimented with a third classification approach that also supports this feature, *Support Vector Machines* (SVMs) [2], which have been found quite effective for text matching problems [11]. SVM classifiers conceptually convert the original measurements of the features in the data to points in a high-dimensional space that facilitates the separation between the two classes more than the original representation. While the transformation between the original and the high-dimensional space may be complex, it needs not to be carried out explicitly. Instead, it is sufficient to calculate a *kernel* function that only involves dot products between the transformed data points, and can be calculated directly in the original feature space. We report experiments with two of the most common kernel functions: a linear kernel,

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y} + 1$$

and a Gaussian (radial basis function) kernel,

$$K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$$

where σ is a parameter (representing the standard deviation of the underlying distribution). This latter kernel has been recommended for text matching tasks [10]. Regardless of the choice of kernel, determining the optimal classifier is equivalent to determining the hyperplane that maximizes the total distance between itself and representative transformed data points (the *support vectors*). Finding the optimal classifier therefore becomes a constrained quadratic optimization

[6] Studying the effect on classification accuracy of a richer feature set (e.g., including as well all words on the result pages) is the subject of interesting future work.

[7] This is because the modeled distribution is binomial rather than normal, and hence the variance depends on the mean—see [19] for the technical details.

[8] `http://www.r-project.org/`

Set	Original number of queries	Number of appropriate queries	Global		Local	
Training	595	439	368	(83.8%)	71	(16.2%)
Development	199	148	125	(84.5%)	23	(15.5%)
Test	497	379	334	(88.1%)	45	(11.9%)

Table 1: Distribution of global and local queries in our training, development, and test sets.

problem. In our experiments, we use the SVM-Light implementation[9] [12].

In many binary classification tasks, one of the two classes predominates, and thus trained classifiers tend to favor that class in the absence of strong evidence to the contrary. This certainly applies to our task; as we show in Section 5.1, 83–89% of web queries are global. Weiss and Provost [21] showed that this imbalance can lead to inferior classifier performance on the test set, and that the problem can be addressed through oversampling of the rarer class in the training data. Their method examines different oversampling rates by constructing artificial training sets where the smaller class is randomly oversampled to achieve a specific ratio between samples from the two classes. For each such sampling ratio, a classifier is trained, which assigns a score to each object indicating strength of evidence for one of the classes. By fixing a specific strength threshold, we divide the classifier output into the two classes. Further, by varying this threshold[10] we can obtain an error-rate curve for each class as a function of the threshold. The entire process results in a *Receiver-Operator Characteristic* (ROC) curve [3] for each sampling ratio. Specific points on the curve that optimize the desired combination of error rates can then be selected, and the performance of the classification method across the different thresholds can be measured from the area between the curve and the x-axis. Weiss and Provost use the C4.5 classifier [17], a decision tree classifier with additional pruning of nodes to avoid overfitting. We use a software package provided by them (and consequently also the C4.5 algorithm) to explore the effect that different ratios of global to local queries during training have on classifier performance.

5. EXPERIMENTAL RESULTS

We now describe the data (Section 5.1) and metrics (Section 5.2) that we use for the experimental evaluation of the query classifiers (Section 5.3).

5.1 Data

For the experiments reported in this paper, we used a sample of real queries submitted to the Excite search engine.[11] We had access to a portion of the December 1999 query log of Excite, containing 2,477,283 queries. We randomly selected initial sets of queries for training, development (tuning the parameters of the classifiers we train), and testing purposes by selecting each of these queries for inclusion in each set with a constant (very small) probability. These probabilities were set to 400/2,477,283, 400/2,477,283, and 500/2,477,283 for the three sets, respectively. Subsequently we combined the training and development set, and reassigned the queries in the combined set so that three-fourths were placed in the training set and one-fourth in the development; we kept the test set separate. This process generated 595, 199, and 497 queries in the initial versions of the training, development, and test sets. We further eliminated queries that passed any of the following tests:

- Upon examination, they appeared likely to produce results with explicit sexual content.
- When supplied to Google—and after filtering out any non-HTML results and any broken links—the queries produced fewer than 40 files. This constraint is meant to ensure that we are not including in our experimental data queries that contain misspellings or deal with extremely esoteric subjects, for which not enough material for determining locality would be available.
- They had already been included in an earlier set (we constructed first the training set, then the development set, and finally the test set). Since multiple people may issue the same query, duplicates can be found in the log. Although our algorithms take no special advantage of duplicates, we eliminated them to avoid any bias. Taking into account variations of upper/lower case and spacing between queries (but not word order), this constraint removed 6 queries from the test set.

These filtering steps left us with 439 queries in the training set, 148 queries in the development set, and 379 queries in the test set.

We then classified the queries using the criteria laid out in Section 3. Table 1 shows the size of the three sets before and after filtering, and the distribution of local and global queries in each set. We observe that, in general, most queries (83–89%) tend to be global.

5.2 Evaluation Metrics

We consider a number of evaluation metrics to rate the performance of the various classifiers and their configurations. Since a large majority of the queries are global (85.6% in the training, development, and test sets combined), overall classification accuracy (i.e., the percentage of correct classification decisions) may not be the most appropriate measure. This is because a baseline method that always suggests the most populous class ("global") will have an accuracy equal to the proportion of global queries in the evaluated set. Yet such a classifier will offer no improvement during search since it provides no new information. The situation is analogous to applications in information retrieval or medicine where very few of the samples should be labeled positive (e.g., in a test for a disease that affects only 0.1%

[9] Available from `http://svmlight.joachims.org/`.

[10] Setting the threshold to each extreme assigns *all* or *none* of the data points to that category.

[11] `http://www.excite.com/`

of patients). While we do not want overall accuracy to decrease from the baseline (at least not significantly), we will utilize measures that capture the classifier's improved ability to detect the rarer class relative to the baseline method.

Two standard such metrics are *precision* and *recall* for the local queries. Precision is the ratio of the number of items correctly assigned to the class divided by the total number of items assigned to the class. Recall is the ratio of the number of items correctly assigned to a class as compared with the total number of items in the class. Note that the baseline method achieves precision of 100% but recall of 0%. For a given classifier with adjustable parameters, often precision can be increased at the expense of recall, and vice versa; therefore we also compute the *F-measure* [20] (with equal weights) to combine precision and recall into a single number,

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Finally, we argued earlier that one kind of misclassification errors may be assigned a higher cost. We can then calculate the *average cost* [15],

$$\text{Average cost} = \sum_{X \in \{\text{Global,Local}\}} \text{Cost}(X) \times \text{Rate}(X)$$

where $\text{Cost}(X)$ is the cost of wrong X classifications and $\text{Rate}(X)$ is the rate of wrong X classifications. Average cost is the measure to minimize from a decision theory perspective. The rate of wrong classifications for a class is equal to the number of data points that have been misclassified into that class divided by the total number of classification decisions, and the costs for each misclassification error are predetermined parameters. If both costs are set to 1, then the average cost becomes equal to the total error rate, i.e., one minus accuracy. In our experiments, we report the average cost considering the mislabeling of global queries as local twice as important as the mislabeling of local queries, for the reasons explained in the previous section.

5.3 Results

We trained the classifiers of Section 4.2 on the 439 queries in our training set. RIPPER and the regression model were trained on that training set without modification. For C4.5 and SVMs, we explored the effect that different proportions of local queries in the training set have on overall performance. For that purpose, we used our development set to evaluate the performance effects of different local query proportions, and select the optimal classifier within each family.

For the C4.5-based classifier, we used the C4.4 software provided by Foster Provost and Claudia Perlich to explore the effect of different proportions of local and global queries. We created training sets by randomly oversampling or undersampling the minority (local) class as needed, in increments of 10%. For any given proportion of local queries between 10% and 90%, we started from our training set, modified it according to the above sampling method to have the desired proportion of local queries, trained the corresponding C4.5 classifier, and evaluated its performance on our development set. The natural proportion of the local class in the unmodified training data is also included as one of the proportions used to build and evaluate a classifier. In this manner, we obtain curves for the various metrics as the proportion of local queries varies (Figure 1). We observe

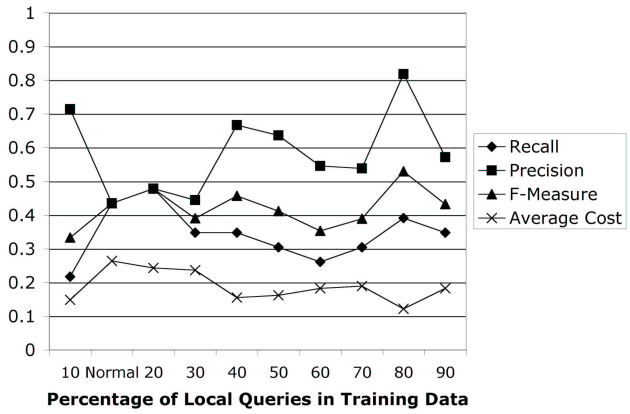

Figure 1: Evaluation metrics for C4.5 classifiers trained on different proportions of local queries.

that the highest value for precision and F-measure, and the lowest value for the average cost, are obtained when the classifier is trained with a significantly amplified proportion of local queries (80%). Further, running C4.5 with 80% local queries also produced the largest area under the ROC curve obtained when different precision/recall tradeoffs in the development set are explored. On the basis of this information, we selected the proportion of 80% local queries as the optimal configuration for C4.5. We refer to that configuration as C4.5(80), and this is the version of C4.5 that we evaluated on the test set.

Using our own implementation for constructing extended training sets with a given proportion of local queries, we performed similar experiments for Support Vector Machines with linear and Gaussian kernels. For these classifiers, we also experimented with versions trained with equal error costs for the two kinds of classification errors, and with versions where, during training, a false local assignment counts for twice as much as a false global assignment. We found that the optimal proportion of local queries is closer to the natural proportion with SVMs compared to C4.5 classifiers; the proportions chosen from our development set were 50% for the linear SVM classifier with equal error costs, 30% for the linear SVM classifier with unequal error costs, 30% for the Gaussian SVM classifier with equal error costs, and 20% for the Gaussian SVM classifier with unequal error costs. We denote the optimal classifiers from these four families as SVM-Linear-E(50), SVM-Linear-U(30), SVM-Gaussian-E(30), and SVM-Gaussian-U(20), respectively. Figure 2 shows the curve obtained for the SVM-Gaussian-U family of classifiers.

Having determined the best value for the proportion of local queries for C4.5 and SVM-based classifiers, we evaluate these classifiers, as well as the classifiers obtained from RIPPER and log-linear regression, on our test set.[12] Table 2 shows the values of the evaluation metrics obtained on the unseen test set. The classifier using a linear kernel SVM with unequal error costs achieves the highest F-measure,

[12]We also experimented with variable error costs for the RIPPER classifier, using the same 2:1 error cost correspondence, but the resulting classifier was identical to the RIPPER classifier obtained with equal error costs.

Classifier	Recall	Precision	F-Measure	Average Cost	Accuracy
RIPPER	53.33%	47.06%	50.00%	0.1979	87.34%
Log-linear Regression	37.78%	58.62%	45.95%	0.1372	89.45%
C4.5(80)	40.00%	32.73%	36.00%	0.2665	83.11%
SVM-Linear-E(50)	48.89%	48.89%	48.89%	0.1821	87.86%
SVM-Linear-U(30)	48.89%	53.66%	51.16%	0.1609	88.92%
SVM-Gaussian-E(30)	37.78%	53.13%	44.16%	0.1530	88.65%
SVM-Gaussian-U(20)	37.78%	53.13%	44.16%	0.1530	88.65%
Baseline (always global)	0.00%	100.00%	0.00%	0.1187	88.13%

Table 2: Evaluation metrics on the test set of selected classifiers optimized over the development set.

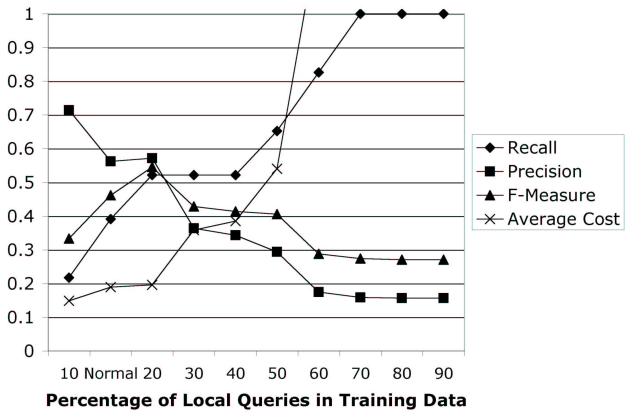

Figure 2: Evaluation metrics for Support Vector Machines with Gaussian kernel and false local assignments weighted twice as much as false global assignments, trained on different proportions of local queries.

while the log-linear classifier achieves the lowest average classification cost. As expected, the SVM classifiers that were trained with unequal error costs achieve the same or lower average cost (which also utilizes the same unequal error costs) compared to their counterparts trained with equal error costs. Overall, RIPPER, log-linear regression, and the two SVM classifiers with linear kernels achieve the highest performance, with small differences between them. They are followed by the two SVM classifiers with a Gaussian kernel function, while C4.5 trails significantly behind the other classifiers.

The features used for classification vary considerably from classifier to classifier.[13] RIPPER achieves one of the best classification performances using only one simple rule, based only on the average number of city locations per returned web page: if that number exceeds a threshold, the query is classified as local, otherwise as global. On the other hand, the C4.5 and SVM classifiers utilize all or almost all the features. The log-linear regression classifier falls in-between these two extremes, and primarily utilizes the average numbers of unique city, state, and country names per retrieved page, as well as the total number of unique locations per page (4 features).

For concreteness, and to conclude our discussion, Table 3 shows the performance of our classifiers on a few representative examples of local and global queries.

[13]Most classifiers automatically ignore some of the provided features, to avoid overfitting.

6. IMPROVING SEARCH RESULTS

The core of this paper is on classifying queries as either local or global. In this section, we present preliminary ideas on how to exploit this classification to improve the quality of the query results. Further exploration of these and other directions is the subject of interesting future work.

Consider a query that has been classified as local using the techniques of Section 4. By definition, this query is best answered with "localized" pages. We can easily determine if the query includes any location name by using the dictionary-based approach of Section 4.1. If no locations are present in the query (e.g., as in query [houses for sale]), in the absence of further information we can attempt to "localize" the query results to the geographical area of the user issuing the query, for which we can rely on registration information provided by the user, for example. Consequently, we can simply expand the query by appending the user's location to it, to turn, say, the query [houses for sale] into [houses for sale San Francisco] for a San Francisco resident. Alternatively, a search engine might attempt to obtain additional information from the user to further localize the query as appropriate. For example, the query [houses for sale] can then be transformed into [houses for sale New York City] for a San Francisco resident who is moving to New York City. In either case, the expanded query will tend to produce much more focused and localized results than the original query does. As of the writing of this paper, all of the top-10 results returned by Google for query [houses for sale San Francisco] are results of relevance to a person interested in Bay Area real estate. In contrast, most of the results for the original query, [houses for sale], are irrelevant to such a person, as discussed in the Introduction. An alternative, more expensive strategy for handling these queries is to compute and exploit the geographical scope of web pages as defined in [9]. Then, pages with a geographical scope that includes the location of the user issuing the query would be preferred over other pages. In contrast, a local query in which locations are mentioned is likely to return pages with the right locality, making any further modification of the query or reranking of the results unnecessary.

Consider now a query that has been classified as global using the techniques of Section 4. By definition, this query is best answered with "broad" pages. Rather than attempting to modify a global query so that it returns "broad" pages, we can follow a result reranking strategy to privilege these pages over more localized ones. One possible reranking strategy is to reorder the results from, say, Google for

Class	Query	Classifier						
		RIPPER	Regression	C4.5(80)	SVM-LE	SVM-LU	SVM-GE	SVM-GU
Global	[Perl scripting]	Global	−0.9381	Global	−1.9163	−1.7882	−1.0627	−1.0590
	[world news]	Global	−0.8306	Local	−0.5183	−0.3114	−0.4166	−0.1440
	[wildflowers]	Global	−0.5421	Global	−0.7267	−0.8082	−0.8931	−0.8144
	[Elgin marbles]	Local	0.4690	Local	0.6426	0.6654	0.0378	0.1016
	[Galapagos Islands]	Global	−0.7941	Global	−1.2834	−1.1998	−0.9826	−0.8575
	[Boston zip code]	Local	−0.0243	Local	0.6874	0.6152	0.0408	0.0797
	[Woody Allen NYC]	Global	−0.2226	Global	−0.3253	−0.3541	−0.6182	−0.5272
Local	[houses for sale]	Global	−0.6759	Global	−1.0769	−1.0962	−0.9242	−0.8516
	[Volkswagen clubs]	Local	−0.0933	Global	1.0844	0.7917	0.0562	0.0837
	[Wisconsin Christmas tree producers association]	Global	0.1927	Local	−0.1667	−0.4421	−0.4461	−0.3582
	[New York style pizza delivery]	Global	−0.0938	Global	−0.5945	−0.6809	−0.5857	−0.4824

Table 3: Classification assignments made by different classifiers on several example queries. SVM-LE, SVM-LU, SVM-GE, and SVM-GU stand for classifiers SVM-Linear-E(50), SVM-Linear-U(30), SVM-Gaussian-E(30), and SVM-Gaussian-U(20), respectively. For regression and SVM classifiers, positive numbers indicate assignment to the local class, and negative numbers indicate assignment to the global class; the absolute magnitude of the numbers increases as the classifier's confidence in its decision increases. (We linearly transformed the regression output from the (0, 1) to the (−1, 1) range.) The scale of the numbers is consistent across queries and between all SVM classifiers, but not directly comparable between regression classifiers (bound between −1 and 1) and SVM classifiers (unbounded).

the unmodified query based on the geographical scope of the pages as defined in [9]. Thus pages with a broad geographical scope (e.g., covering the entire United States) would prevail over other pages with a narrower scope. A less expensive alternative is to classify the result pages as local or global following a procedure similar to that of Section 4 for queries. Specifically, we implemented this alternative by training C4.5RULES, a rule-based version of the C4.5 decision-tree classifier, with a collection of 140 web pages categorized in the Yahoo! directory. Pages classified under individual states in the "Regional" portion of the directory were regarded as local, while pages under general categories were regarded as global. The feature representation for the pages was analogous to that for the queries in Section 4.1 but restricted to features that are meaningful over individual pages (e.g., total number of locations on a page), as opposed to over a collection of pages (e.g., minimum number of locations per page in the top-50 result pages for a query). At query time, we reorder the results so as to privilege global pages over local ones. This is based on the locality classification of the pages, which can be precomputed off-line since it is query-independent or performed on the fly as we do in our prototype implementation. This procedure is efficient, and produced promising initial results for a handful of global queries (e.g., [wildflowers]) that we tried.

Our preliminary approach to query modification is therefore as follows: Given a query specified by the user, we supply first the unmodified query to the search engine and collect the top 50 results. We extract location names from these results[14], and calculate the features of Section 4.1. Using one of the best performing classifiers of Section 4, we determine if the query is global or local. If it is local and contains at least one location name, nothing is done—the results returned from the unmodified query are presented to the user. If the query is local and contains no location, we add the user's location (or, alternatively, request further information from the user, as discussed), reissue the query and present the results. Finally, if the query is global, we calculate the scope of each retrieved web page using part of the location features computed earlier and the C4.5RULES classifier, and rerank the results so that more global pages are higher in the list shown to the user. We have built a prototype implementation of this algorithm, using the classifier obtained from RIPPER (because of the relative simplicity of its rules) for *query* classification, and Google as the search engine.

[14]As noted earlier, these names could be cached along with each web page at the time of indexing, to increase efficiency.

7. CONCLUSION

We have described an attribute of queries, locality, that—to the best of our knowledge—has not been explored before either in theoretical work or in practical search engines but can significantly affect the appropriateness of the results returned to the user. We defined a categorization scheme for queries based on their geographical locality, and showed how queries can be represented for purposes of determining locality by features based on location names found in the results they produce. Using these features, automatic classifiers for determining locality can be built. We explored several state-of-the-art classification approaches, and evaluated their performance on a large set of actual queries. The empirical results indicated that for many queries locality can be determined effectively.

The bulk of the paper discussed methods for classifying queries according to locality, and empirically established that this is desirable and feasible for many queries. We also presented some first thoughts on possible query refor-

mulation and result reranking strategies that utilize locality information to actually improve the results the user sees. Although our strategies for query modification and result reranking are preliminary, they illustrate a promising family of approaches that we plan to investigate in the future so that we can exploit the classification of queries based on their geographical locality in order to improve search result quality.

Acknowledgments

This material is based upon work supported in part by the National Science Foundation under Grants No. IIS-97-33880 and IIS-98-17434. We are grateful to Claudia Perlich and Foster Provost for providing us with their adaptation of the C4.5 classifier that we used in our experiments. Also, we would like to thank Thorsten Joachims for answering our questions on SVM-Light, and David Parkes for his helpful comments and insight.

8. REFERENCES

[1] D. M. Bates and D. G. Watts. *Nonlinear Regression Analysis and its Applications*. Wiley, New York, 1988.

[2] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh, 1992.

[3] A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, **30**(7):1145–1159, 1998.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference (WWW7)*, Apr. 1998.

[5] C. Buckley, J. Allan, G. Salton, and A. Singhal. Automatic query expansion using SMART: TREC 3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 69–80, April 1995. NIST Special Publication 500-225.

[6] O. Buyukkokten, J. Cho, H. García-Molina, L. Gravano, and N. Shivakumar. Exploiting geographical location information of web pages. In *Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB'99)*, June 1999.

[7] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the Seventh International World Wide Web Conference (WWW7)*, Apr. 1998.

[8] W. W. Cohen. Learning trees and rules with set-valued functions. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1996.

[9] J. Ding, L. Gravano, and N. Shivakumar. Computing geographical scopes of web resources. In *Proceedings of the Twenty-sixth International Conference on Very Large Databases (VLDB'00)*, 2000.

[10] G. W. Flake, E. J. Glover, S. Lawrence, and C. L. Giles. Extracting query modifications from nonlinear SVMs. In *Proceedings of the Eleventh International World-Wide Web Conference*, Dec. 2002.

[11] M. A. Hearst. Trends and controversies: Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, July 1998.

[12] T. Joachims. Estimating the generalization of performance of an SVM efficiently. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 2000.

[13] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, Jan. 1998.

[14] K. S. McCurley. Geospatial mapping and navigation of the web. In *Proceedings of the Tenth International World Wide Web Conference (WWW10)*, May 2001.

[15] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning*, Sept. 1997.

[16] R. Purves, A. Ruas, M. Sanderson, M. Sester, M. van Kreveld, and R. Weibel. Spatial information retrieval and geographical ontologies: An overview of the SPIRIT project. In *Proceedings of the 25th ACM International Conference on Research and Development in Information Retrieval (SIGIR'02)*, 2002.

[17] R. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.

[18] G. Salton. *Automatic Text Processing: The transformation, analysis, and retrieval of information by computer*. Addison-Wesley, 1989.

[19] T. J. Santner and D. E. Duffy. *The Statistical Analysis of Discrete Data*. Springer-Verlag, New York, 1989.

[20] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979.

[21] G. M. Weiss and F. Provost. The effect of class distribution on classifier learning: An empirical study. Technical Report ML-TR-44, Computer Science Department, Rutgers University, Aug. 2001.