# Automatically Identifying Localizable Queries

Michael J. Welch
University of California, Los Angeles
Los Angeles, CA 90095
mjwelch@cs.ucla.edu

Junghoo Cho
University of California, Los Angeles
Los Angeles, CA 90095
cho@cs.ucla.edu

## ABSTRACT

Personalization of web search results as a technique for improving user satisfaction has received notable attention in the research community over the past decade. Much of this work focuses on modeling and establishing a profile for each user to aid in personalization. Our work takes a more query-centric approach. In this paper, we present a method for efficient, automatic identification of a class of queries we define as *localizable* from a web search engine query log. We determine a set of relevant features and use conventional machine learning techniques to classify queries. Our experiments find that our technique is able to identify localizable queries with 94% accuracy.

## Categories and Subject Descriptors

H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval—*Search process*; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

## General Terms

Experimentation, Human Factors, Measurement

## Keywords

Localizable query, web search, machine learning

## 1. INTRODUCTION

Typical queries submitted to web search engines contain very short keyword phrases [6]. These short text queries are generally insufficient to fully specify a user's information need, yet users still have an expectation of finding relevant content. With the global search market approaching 10 billion queries per month[1] and substantial incentives to increase market share, maximizing user satisfaction with query results is in constant focus. Researchers have investigated several techniques for personalizing search results by incorporating contextual metadata about the user during both document retrieval and result ranking. Others have attempted to classify queries into general categories based on perceived user intent.

We address the problem from a different angle, by studying which metadata is most applicable to a particular *query*. In this paper, we will present a technique for automatically identifying a class of queries we define as *localizable* from a web search engine query log. *Localizable* queries are those search strings for which the user would implicitly prefer to see results prioritized by their geographical proximity; "airport shuttle" or "Italian restaurant", for example, are likely submitted by a user with the goal of finding information or services relevant to their current whereabouts. Our analysis suggests a significant fraction of user queries would benefit from such localization. Users, however, explicitly add location constraints to less than one half of such queries. By automatically localizing the appropriate queries, we can not only improve a user's search experience, but also the sponsored advertisement matching for locally available goods and services.

In addition to traditional desktop and laptop PCs, mobile and handheld devices are rapidly becoming a major source for information retrieval from the web. Many such devices are already equipped with geocoding capabilities though GPS, triangulation, or other techniques, and market research estimates that nearly all will contain some form of location awareness within the next few years. Currently, providers typically offer location-aware services and mobile-specific content though special portals[2][3][4]. This creates a burden on both the user, who must remember the mobile-specific sites, and the content providers, who must maintain parallel versions of their sites.

We aim to eliminate the need for such redundancy and specialization by answering the question: How can "traditional" web search engines know *when* to incorporate a variety of rich per-user metadata in their results? Our approach is to identify the queries which contain locations as contextual modifiers, and extract the base portion of those queries. We define a set of distinguishing features and experiment with a variety of classifiers to automatically identify the set of localizable queries from a search engine query log [20].

Our objective is to ultimately improve overall user satisfaction with web search results by automatically localizing queries when appropriate. It is worth noting that, in this pa-

[1] http://searchenginewatch.com

[2] http://www.google.com/mobile/
[3] http://livesearchmobile.com/
[4] http://mobile.yahoo.com/


*SIGIR'08,* July 20–24, 2008, Singapore.

per, we are focusing on the task of *identifying* which queries would benefit from localization. The decision to localize is based on a classifier we construct, and thus we feel it is important to prioritize the accuracy of classification. In particular, we wish to avoid "false-positive" classifications, which would lead to erroneously localizing queries. In cross validation experiments, our technique correctly identifies approximately 46% of the "localizable" queries with 94% accuracy.

The rest of the paper is organized as follows. Section 2 discusses related work in query refinement and personalization. We discuss user studies providing the motivation behind our work in section 3 and outline our approach in section 4. We analyze the query log data and discuss localizable query identification in section 5, and describe the relevant features for classification in section 6. Our results with several supervised classifiers are presented in section 7. Section 8 discusses our conclusions and presents some ideas for future work.

## 2. RELATED WORK

Considerable work has been done in the area of personalized and per-user web search. Researchers have investigated several data sources for refining or expanding user queries with relevant contextual terms, including in-link anchor text [13], co-occurring terms within the result set [25, 2], co-occurring terms in query logs [5], and keywords from the user's desktop environment [12].

Personalization of ranking algorithms was proposed in [19] by adjusting the "random-surfer" model based on user preferences. Jeh and Widom[7] addressed some of the scalability concerns of this personalized PageRank approach.

Taxonomies and ontologies have been used to filter and rank search results using concept weights learned from user browsing behavior [22, 24]. Liu et. al [17] discuss personalizing and disambiguating queries by classifying them into per-user category profiles based on past browsing history.

Others have performed query-dependent studies, though the focus is typically not per-user personalization. Lau and Horvitz [14] use Bayesian networks to estimate user goals by classifying query refinement patterns found in search engine logs. Lee et. al [15] present a set of features for automatically classifying user intent as *navigational* or *informational* for a set of queries.

Many of these personalization techniques are effective when a user's query is ambiguous, such as those containing terms with homonyms or multiple senses. Our work differs from most prior research in personalized web search by addressing personalization from a *query-dependent* focus, rather than user-dependent. This allows us to personalize results by utilizing applicable metadata whenever it is deemed appropriate, not only for ambiguous queries.

## 3. MOTIVATIONAL STUDIES

Before we discuss the technical aspects of our work, it is worth spending a moment to focus on a few important preliminary questions. In particular, we wish to verify that the concept of "localizable" is *consistent* amongst users, and that automatic localization is *worthwhile*.

### 3.1 Query Coverage

Web search engines typically incorporate a complex mix of factors when ranking query results. Factoring in localization increases the complexity of the system, and so it is important to verify that its overall impact justifies that complexity. To that end, we conducted a user study to estimate the percentage of query instances which would benefit from localization. We gave 9 survey participants each a different list of 100 randomly sampled entries from an America Online search query log [20] and asked them to classify each query into one of three categories:

1. Query would likely not benefit from localization
2. Query would likely benefit from automatic localization
3. Query is already localized

Our survey participants said that, on average, 70% of queries would not benefit from localization, 16% of queries would benefit from automatic localization, and 14% of queries were already localized. Automatic localization may potentially improve the results for a significant fraction of user queries, as these results suggest that, while approximately 30% of the queries issued to search engines are localizable, users only explicitly localize about one-half of them.

### 3.2 User Agreement

Research involving user satisfaction and search result personalization typically must deal with some level of subjectivity. Automatic localization is a form of personalization, and so we pose the question: *do users generally agree on which queries should be localized*? To address this issue, we administered a second user survey. As our goal is to now see whether users agree on which queries are localizable, and approximately 15% of queries in the log are localizable, we felt a random sample from the query log would not provide sufficient opportunity for users to disagree. We constructed a list of 102 queries, approximately one half of which we believed to be localizable. This list was presented to 8 participants who were asked to make a binary judgment for each query about whether it would benefit from localization or not.

The results were tabulated to determine whether users agree on which queries would benefit from localization. Figure 1 shows a plot of user agreement, with the number of users who *disagreed* with the majority along the X-axis, and the number of queries on the Y-axis. We found that users agreed that queries for goods and services, such as "food supplies" and "home health care providers" were localizable, while more general queries for information, such as "calories coffee" and "eye chart" are not. The intent of other queries, such as "medical license" and "marathon" are more vague, and our survey participants were evenly divided.

We note that subjects were asked to make their best interpretation of the query intent given only the query text, and so some level of discrepancy is expected. The overall results are encouraging, as we see that users are evenly divided on only 8 of the queries, while at most one person disagreed with the majority for about 50% of the queries.

### 3.3 Preliminary Results

Without a complete implementation of a "localizing" search system to perform experiments with, we must find other ways to estimate the overall user satisfaction with query localization. User studies by Joachims et. al. [9] suggest that clickthroughs are a reasonable approximation of relevance feedback. During query feature collection, we measured the clickthrough rates for both the localized and non-localized

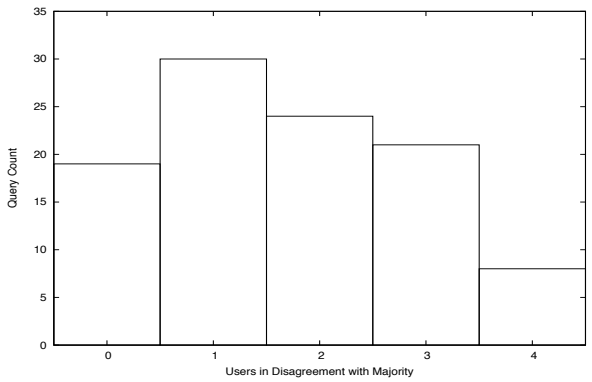


**Figure 1: User Agreement Survey.**

form of the same "base" query, and found that the average clickthrough rate for the localized instances of a localizable query is approximately 17% higher than for the non-localized instances. While not a perfect interpretation of user preference, it is an auspicious precursory indicator.

# 4. OVERVIEW

For any query $q$, we wish to efficiently determine whether $q$ is localizable or not. Our basic approach is to build a query classifier using features collected from a web search engine query log. This classifier can then be used by the search engine to make realtime decisions about localization on a per-query basis.

We begin identifying localizable queries by finding previously issued queries which contain an explicit localization modifier, with the assumption that the "base" of these queries may be generally localizable. We identify all entries in the query log which contain "locations" and extract the "base" of these queries. Once we have the set of all base queries, we select a sample to use for classifier training. For this subset of queries we compute relevant distinguishing features and evaluate multiple well-known supervised classifiers to determine which are best suited for our task. Each of these steps are discussed in the following sections.

# 5. QUERY LOG ANALYSIS

In our analysis we use a search query log from America Online [20], which contains queries from 657,426 distinct users over a three month period from March 1 to May 31, 2006. The log contains approximately 36 million rows of data, covering 10 million textually unique queries from 21 million search "instances".

We start construction of our classifier by finding queries in the log which contain location modifiers. The AOL query log contains queries in the English language, and so we have focused our location identification on states, counties, and cities in the United States using a list available from the U.S. Census Bureau[5]. For queries which contain one or more of these locations, we consider the location as a contextual modifier added by the user, and remove it to find the "base" query. For example, the base of the query "san francisco public parks" is "public parks". These base queries are the ones which we would like to automatically localize.

[5]http://www.census.gov/

| Query ID | Base Query | Location Tag |
|---|---|---|
| 10005397 | county florida animal shelter | city:lee |
| 10005397 | \|—— county animal shelter | city:florida |
| 10005397 | \|—— county animal shelter | state:florida |
| 10005397 | florida animal shelter | county:lee county |
| 10005397 | \|—— animal shelter | city:florida |
| 10005397 | \|—— animal shelter | state:florida |
| 10005397 | lee county animal shelter | city:florida |
| 10005397 | \|—— county animal shelter | city:lee |
| 10005397 | \|—— animal shelter | county:lee county |
| 10005397 | lee county animal shelter | state:florida |

**Table 1: Base Query Generation**

In the remainder of this section, we will discuss how base queries are identified from the query log, how queries sharing a "similar" base are grouped together, and some considerations for matching user queries to entries in the log.

## 5.1 Identifying Base Queries

Queries are typically very short, consisting of only 2-3 terms [6] rather than complete, grammatically correct sentences. Additionally, all queries in the log have been normalized to lower case. As a result it is difficult to employ general linguistic approaches, such as parts-of-speech tagging, to aid location tagging. Likewise, techniques based on other indicators, such as capitalization or punctuation, may not reliably label locations within queries.

Instead of using cumbersome grammatical tools to identify locations, we use a simple string matching process, and ensure accuracy using a set of features carefully selected to eliminate false positives. While relatively straightforward, this technique proves quite effective, and the simplicity supports scalability as well as language independence.

To identify localized queries, we inspect the text of each query and compare it to the Census Bureau list of locations. Every match generates a new base query, where the matched portion of text is tagged with the detected location type (state, county, or city). Queries may contain multiple localizations, such as a city and state name. Rather than complicate our tagger, we choose to simply remove the tagged tokens and enqueue the remainder of the query for further processing. As a result, a single entry in the query log may produce multiple base queries.

We favored this technique over removing all "locations" and generating a single base query from each entry because, in general, we cannot be certain when query terms are specifying a location. Several words in the English language are also used as city names, such as Parks, Arizona. If we choose to remove all terms matching a location in a single step, we would not be able to identify the correct base "public parks" in the example "san francisco public parks" discussed above.

Table 1 shows all of the base queries generated from the source query "lee county florida animal shelter". Indentation is used to illustrate how the original query is processed to ultimately result in each of the possible base queries shown. For example, the first row is obtained by removing "lee" from the original query. The second and third rows are generated by further tagging the resulting base query "county florida animal shelter". For the approximately 10 million distinct entries in the query log, we identify 4.9 million unique base queries.

## 5.2 Base Query Grouping

As Table 1 shows, tagging the query "lee county florida animal shelter" generates 10 entries comprising 5 textually distinct base queries. After processing the entire query log, we group together queries which share a "similar" base query $q_b$, and define $L(q_b)$ as the set of location tags which occur with $q_b$. We explored several alternatives for this similarity mapping, ranging from an exact string match to a bag-of-words model with stopwords eliminated and terms stemmed using Porter's suffix stemming algorithm [21].

The choice of mapping function has implications on the accuracy and coverage of our classifier, as well as how we determine which base query a potentially localizable user query issued to a search engine corresponds to. We will now briefly discuss some of the options considered.

### 5.2.1 Exact Match

An exact match model produces the largest set of distinct base queries, as we only group together the entries that are textually equivalent. Using the entire text of a query allows us to distinguish between semantically different queries whose text may, from an algorithmic point of view, only differ in seemingly insignificant ways.

Exact matching, however, also potentially introduces many unintelligible base queries. With location terms removed from a query, the remaining text may never actually be issued to a search engine as a query by itself. Extracting the base of the query "parks in [city]" would produce "parks in", which is unlikely to appear as a user query, and in fact does not appear in the query log. The query "parks", however, occurs 53 times.

### 5.2.2 Stopword Elimination

Many modern information retrieval systems ignore common words, such as conjunctions and prepositions, frequently referred to as stopwords. By eliminating stopwords from queries, we more easily group together logically equivalent queries, such as "parks in [city]" and "parks near [city]" into a single base "parks".

### 5.2.3 Bag of Words

A bag of words model ignores the ordering of terms in a query. Combined with stopword elimination, this may help consolidate semantically equivalent queries such as "airport shuttle" and "shuttle to airport" into a single common base. In some cases, however, ignoring the word ordering may actually change the meaning of the query.

### 5.2.4 Term Stemming

Term stemming algorithms, such as the suffix stemming algorithm described by Porter [21], can help normalize term tense and plurality. While this may improve precision for some semantically equivalent queries such as "restaurant" and "restaurants", it may also occasionally result in collisions between distinct terms which share a common stem, reducing precision. For example, "universe" and "university" share the common stem "univers".

Other forms of "stemming", such as morphological analysis and lemmatization, may produce more accurate results for related term grouping than algorithmic affix or suffix stemming. Lemmatization is frequently discussed in the field of statistical machine translation [16]. Such techniques are significantly more complex, however, typically requiring additional data sources such as a lexicon and parts-of-speech tagger.

| Stopwords Eliminated | Bag-of-words | Stemmed | Queries |
|---|---|---|---|
| No | No | No | 4,898,589 |
| Yes | No | No | 3,940,233 |
| Yes | Yes | No | 3,808,215 |
| Yes | No | Yes | 3,790,692 |
| Yes | Yes | Yes | 3,640,513 |

**Table 2: Base Query Grouping**

## 5.3 Evaluation

In our classifier evaluation, we found that stopword elimination is the only preprocessing step which has significant impact on the final classification results. Fundamentally, we feel it provides the best combination of normalizing logically equivalent queries with minimal semantic loss. As Table 2 shows, additional processing does not noticeably reduce the size of the base query set, and as a result, calculated feature scores will not change significantly. In the remainder of this paper, when we refer to the base of a localized query, we are referring to the stopword eliminated version.

# 6. FEATURE SELECTION

Our tagging process generates a base query any time it finds text which matches a location. Several city and state names have homonyms, and thus text matching is not sufficient. For example, "kansas" may refer to the state, one of several cities, the rock band, or even a particular movie with that title. When we find a query containing "kansas", how do we know whether the user was referring to a location or one of the other senses of the word?

In this section we discuss a set of features measurable from a query log which a supervised classifier may use to make that determination, and discriminate localizable queries from the false positives. We investigated several features, both about the individual queries as well as aggregate measures of the grouped queries. Some query features, such as frequency counts, have relatively straightforward interpretations. Others are more subtle and require additional discussion.

Although our analysis is performed over a window of user queries contained within a log, real-world query logs collected by search engines are constantly expanding. This necessitates the ability to adapt our classifications as new examples are collected, and we felt it was important to consider this when selecting features. We therefore focused our feature selection on those which are easily calculated incrementally as the data expands.

## 6.1 Localization Ratio

Online information sources such as Wikipedia[6] rely on the collective expertise of their users to ensure the knowledge base is accurate. We adapt a similar model for web search queries, where every query instance can be treated as a "vote". In our case, users vote for the localization of query $q$ by submitting it to the search engine with a location specified.

For every textually distinct query $q_i$ we define $b_i \in [0, 1]$ as the fraction of users who would benefit from the localization of $q_i$. Every query $q_i$ has an associated value $r_i \in [0, 1]$,

[6]http://www.wikipedia.org

| Location Tag | Count |
|---|---|
| city:independence | 156 |
| city:homestead | 5 |
| state:texas | 4 |
| city:lincoln | 2 |

**Table 3: Locations Occurring with "declaration"**

defined as

$$r_i = \frac{Q_i(L)}{Q_i + Q_i(L)} \quad (1)$$

$Q_i$ is the count of all instances of $q_i$ in the query log, and $Q_i(L)$ is the count of all query instances tagged with some location $\ell \in L$, for which $q_i$ is the base. This $r_i$ value represents the "localization ratio" for $q_i$. Assuming some fraction of the users who issued $q_i$ to the search engine implicitly wanted localized results, $r_i$ defines an estimated lower bound on $b_i$.

Localization ratio provides some insight into what fraction of users believe that a particular query would benefit from localization. It is, however, susceptible to small sample sizes, as a query issued by a single user may have an $r_i$ value of 1. Localization ratio is also unable to identify false positives resulting from incorrectly tagged locations. For example, the base query "barnes" has an $r$ value of over 0.75, yet a vast majority of its occurrences come from incorrectly tagging "noble" as a location in the query "barnes and noble". For these reasons, localization ratio is insufficient as the sole feature for classification.

## 6.2 Location Distribution

Our query tagging process is based on string comparison, creating a match for any text which is listed as a US state, county, or city. Some of these locations are homonyms in the English language, introducing false positives to the candidate list which must be filtered out. As an example, "Independence" is a city in Missouri, and is tagged as such in the query "declaration of independence". The base query "declaration" occurs a total of 176 times with some location, 113 alone of which are due to the query "declaration of independence". Table 3 shows the top 4 most frequently occurring locations for the base query "declaration".

To aid in identifying these entries, we start with a basic assumption about any localized query $q_l$:

$$\forall \ell \in L(q_b) \Pr[\ell \in q_l | q_b = base(q_l)] \approx \frac{1}{|L(q_b)|} \quad (2)$$

That is, given an instance of any localized query $q_l$ with base $q_b$, the probability of $q_l$ containing location $\ell$ is approximately equal across all possible locations of $q_b$. Base queries which have a highly skewed distribution of location occurrence counts suggest that either the query is only relevant to those locations, or the tagged "location" is actually part of the query, rather than a localization modifier.

To estimate the distribution, we calculate several measures for the set of locations $\ell \in L(q_b)$, including minimum, maximum, mean, median, and standard deviation of their occurrence counts.

## 6.3 Clickthrough Rates

Search result clickthrough rates have been used in studies evaluating and improving the effectiveness of web search engines [8]. User studies by Joachims et. al. [9] suggest that clickthroughs are a reasonable approximation of relevance feedback.

These studies focus on improving the document ordering by treating user clickthroughs as relative relevance judgments and adjusting for any bias the presented ordering introduces. We are primarily concerned, however, with the relevance of the overall result set, as opposed to the relative ranking of the returned documents. In our experiments, we use clickthrough events recorded in the query log as a comparator of user satisfaction with the localized and non-localized versions of their search results. We define a binary user satisfaction function with the results of query instance $q_i$ as:

$$S(q_i) = \begin{cases} 1 & \text{if at least one result was clicked on,} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

We compute the total clickthrough count for each base query $q_b$ as the sum of $S$ over all instances of $q_b$ in the query log. This sum is then divided by the number of instances of $q_b$ to calculate the clickthrough rate. Likewise, we calculate the clickthrough count and rate for the set of localized queries with base $q_b$. We may then compare the clickthrough rates of both the localized and base forms of a query, where a positive difference is considered as an indicator of increased user satisfaction with the results.

## 6.4 Frequency Counts

At first glance, frequency count measures may seem like potential red herrings, given that there is no practical bound on their values. For example, the query "barnes and noble" is tagged as "barnes and [city:noble]" and "[city:barnes] and noble", due to the cities of Noble, Oklahoma and Barnes, Kansas. The query "barnes and nobel" occurs in 2679 unique search sessions, significantly higher than the average of just over 2 occurrences. If we judged localizabilty based on a threshold of popularity, both of the base queries "barnes" and "noble" would likely be (incorrectly) identified as localizable.

Despite this, frequency counts are still useful measures. In particular, frequency count serves as a normalizing or significance factor for other features, such as $r$, by taking into account a query's popularity.

### 6.4.1 User Distribution

In addition to query occurrence counts, we also consider the sources for those occurrences. For every query $q_i$, we calculate the number of distinct users who have issued the query, in both its localized and non-localized forms. These two measures provide a different form of normalization for the occurrence counts $q$ and $q_L$ by adjusting for bias introduced from a single user issuing the same query multiple times.

# 7. EXPERIMENTAL RESULTS

Using the features discussed in the previous section, we now turn our attention to evaluating classification algorithms for learning localizable queries. We manually tagged a training data set and evaluated the effectiveness of several supervised learning algorithms, including naive Bayesian, decision trees, support vector machines, and neural networks. In addition to these individual classifiers, we evaluated techniques for improving accuracy by combining multiple classifiers, in-

| Symbol | Definition |
|---|---|
| $q$ | Occurrence count of a query |
| $q_L$ | Localization count of a query |
| $r$ | Localization ratio |
| $n_L$ | Number of distinct locations occurring with $q$ |
| $\bar{n_L}$ | Average occurrence count for the $n_L$ locations of $q$ |
| $\tilde{n_L}$ | Median occurrence count for the $n_L$ locations of $q$ |
| $\sigma_{n_L}$ | Standard deviation of occurrence count for the $n_L$ locations of $q$ |
| $n_{L_{min}}$ | Minimum occurrence count for the $n_L$ locations of $q$ |
| $n_{L_{max}}$ | Maximum occurrence count for the $n_L$ locations of $q$ |
| $u_q$ | Number of users who issued the query $q$ |
| $u_{q_L}$ | Number of users who issued the localized query $q_L$ |
| $c_q$ | Number of clickthroughs for query $q$ |
| $c_{q_L}$ | Number of clickthroughs for localized query $q_L$ |
| $\bar{c_q}$ | Normalized clickthrough rate for query $q$ |
| $\bar{c_{q_L}}$ | Normalized clickthrough rate for localized query $q_L$ |

**Table 4: Summary of Features**

cluding AdaBoost [4], Bayesian boosting [11], and independent majority voting.

Our experiments were conducted using classifiers and boosting techniques implemented as part of the RapidMiner [18] machine learning framework.

## 7.1 Training Data

In order to eliminate the most impertinent data, we performed two filtering steps when selecting our training data set. Prior to selection, we removed candidate queries with localization count $q_L <= 1$, which reduced the size of the stopword-eliminated candidate localizable query set to approximately 1.7 million. We selected a random sample of 200 entries from this set and, prior to tagging it, performed an additional filtering step as follows: we removed queries which occurred with only one distinct location modifier ($n_L <= 1$), were only issued by a single user ($u_q = 1$), or whose base form was never issued to the search engine ($q = 0$).

After this filtering, we manually tagged the 102 remaining entries from our random sample of candidate localizable queries, 48 of which the author deemed to be localizable. This training set consisting of 48 positive (localizable) and 54 negative (non-localizable) examples was used in a series of classification experiments discussed below. This set comprises the same queries presented to users in our survey discussed in section 3.2, where the author's classification agreed with the majority for 91 of the 102 entries.

## 7.2 Classifier Evaluation

We compare the effectiveness of several well-known supervised classifiers using standard precision and recall measures. As our overall goal is to identify localizable queries and ultimately use that knowledge to alter search query results, we feel it is important to emphasize precision over recall, and in particular, the accuracy of positive (localizable) classifications. In terms of user satisfaction, we believe correctly localizing a smaller subset of all localizable queries is preferable to localizing a larger subset at the expense of increasing the number of incorrectly localized queries.

The precision and recall measures discussed below are for positive example identification based on 10-fold cross-validation experiments. To compensate for our filtering step on the training data, we consider the queries removed to be classified as non-localizable. While filtering does not affect the computed precision, based on our survey in section 3.1 we approximate 15 of these 98 queries are localizable, and adjust the recall score accordingly.

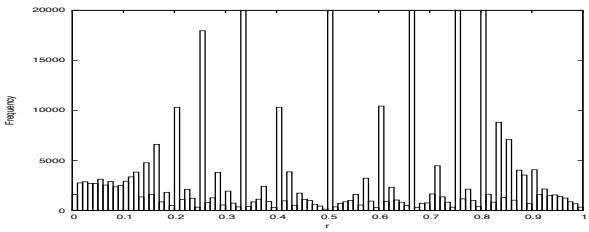


**Figure 2: Localization ratio ($r$) distribution.**

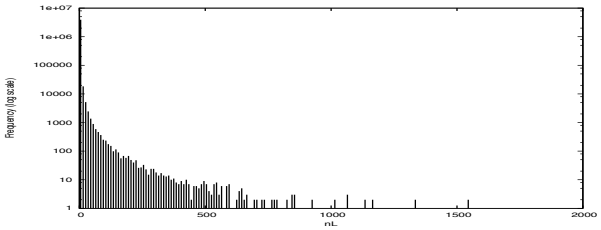


**Figure 3: $n_L$ distribution.**

### 7.2.1 Naive Bayes

Using a set of (assumed independent) feature scores, a naive Bayesian classifier estimates the probability a given instance belongs to each of the possible discrete output classes. In our case, each instance is a user query, and the output is a boolean variable specifying whether the query is localizable or not. Despite simplistic independence assumptions, naive Bayes classifiers typically perform comparably to more complex classifiers [23]. For our data set, the naive Bayes classifier achieves 55% precision at 59% recall.

In addition to feature independence, naive Bayes classifiers assume continuous variables follow a Gaussian probability distribution. The distribution for some features, such as localization ratio ($r$), follows such a distribution. Other features, such as location frequency count ($n_L$) do not follow a Gaussian, as seen in Figures 2 and 3.

As the Gaussian assumption does not hold for all features, we investigate an alternative. Flexible Naive Bayes classifiers use a kernel-based density estimation function for continuous variables, and have been shown to greatly reduce the error rate of naive Bayes classifiers [10]. A kernel-based naive Bayes classifier improves the classification accuracy to 64% precision, albeit at a reduction in recall to 43%.

### 7.2.2 Decision Trees

Decision trees are widely used in data mining and machine learning applications. When constructing a decision tree, the training example set is recursively divided into subgroups based on a particular feature. In our experiments, we construct decision trees with three distinct split criteria: information gain, the Gini coefficient, and the normalized information gain ratio. Table 5 shows the precision and recall measurements for each of these criteria.

| Criteria | Precision | Recall |
|---|---|---|
| Information Gain | 67% | 57% |
| Information Gain Ratio | 64% | 56% |
| Gini Coefficient | 68% | 51% |

**Table 5: Decision Tree Performance**

| Hidden Layers | Precision | Recall |
|---|---|---|
| 1 | 79% | 54% |
| 2 | 85% | 52% |
| 3 | 76% | 49% |
| $> 4$ | n/a | 0% |

**Table 6: Neural Network Performance**

| Base Classifier | Precision | Recall |
|---|---|---|
| Naive Bayes | 63% | 54% |
| Kernel Naive Bayes | 68% | 44% |
| Information Gain | 72% | 57% |
| Information Gain Ratio | 64% | 43% |
| Gini Coefficient | 67% | 56% |

**Table 7: Classifiers with Boosting**

A significant advantage of decision trees is the transparency of the final classifier. We inspected each of the three separate decision trees generated to study which features were the most distinguishing. The localization ratio $r$ was used in all three trees, as were some combination of location distribution measures ($n_L$, $n\bar{i}_L$, and $n\tilde{i}_L$). Click-through rates ($\bar{c_q}$ and $c^-_{q_L}$) were factors in two of the three trees.

#### 7.2.3 SVM

Support Vector Machines (SVMs) [3] are a popular form of supervised learner. SVM is well suited to binary classification problems, where each instance can be represented by a set of $n$ distinct numeric values.

Like many vector-based techniques, SVM classifiers are relatively opaque, making it more difficult to manually inspect and determine which features contributed most significantly to the classification. Regardless, the accuracy and recall of SVM for our classification task surpasses decision trees, achieving 75% precision at an 62% recall rate.

#### 7.2.4 Neural Network

Neural Networks are relatively complex systems capable of, among other tasks, supervised learning for classification [1]. The nodes in a neural network can be separated into input and output layers, and some number of internal "hidden" layers. We evaluated feedforward neural networks comprising of one to three hidden layers, beyond which recall for positive training examples dropped to zero. Table 6 show the results, which indicate that neural networks are the most accurate of the individual classifiers evaluated.

#### 7.2.5 Boosting

Boosting algorithms, such as AdaBoost [4], have been shown to improve the accuracy of "weak" learning classifiers. The final "strong" classifier produced by the boosting algorithm generally consists of a weighted combination of multiple weak classifiers, iteratively trained with a weighted set of examples based on previous classification errors. We evaluated the effectiveness of BayesBoost [11] with naive Bayesian classifiers and AdaBoost with decision trees. The results were mixed, as shown in Table 7. In some cases, precision and recall actually decreased.

#### 7.2.6 Ensemble Classifiers

While the supervised classifiers discussed above produce relatively high precision results, we noticed that the set of false positives (queries incorrectly classified as localizable) produced by the individual classifiers did not fully overlap. We experimented with another style of aggregate learner, where the final classification is determined by the majority vote from a set of discrete classifiers. Unlike boosting, which builds a final classifier from multiple instances of the same learning algorithm trained on varying example sets, this "ensemble" style classifier comprises distinct learners trained on the same example set.

| Decision Tree Criteria | Precision | Recall |
|---|---|---|
| Information Gain | 94% | 46% |
| Information Gain Ratio | 90% | 44% |
| Gini Coefficient | 93% | 41% |

**Table 8: Ensemble Classifier Results**

We choose to combine the best individual performing classifiers using a simple majority vote scheme, where each component classifier is given equal weight, we achieved significantly higher precision than any individual classifier: up to 94% precision at 46% recall. Table 8 shows the results for three such voting classifiers, each consisting of a neural network with two hidden layers, an SVM classifier, and a decision tree with the specified split criteria.

### 7.3 Discussion

Our evaluations demonstrate that conventional supervised learning algorithms are capable of distinguishing localizable queries with relatively high levels of precision. Neural networks successfully identify over one half of localizable queries with 85% accuracy. SVMs identify a larger subset of localizable queries than neural networks, while precision decreases to 75%. Taking the majority vote of these two independent classifiers along with a decision tree, we are able to achieve over 90% classification accuracy.

## 8. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a scalable technique for determining which query strings submitted to a web search engine would benefit from automatic localization. Using data from a query log, we have shown that straightforward query tagging combined with an appropriate set of features and a standard supervised classifier can achieve up to 85% precision. A meta-classifier comprised of three conventional classifiers performs even better, achieving 94% precision in cross-validation experiments.

Our focus has been on identifying explicit locations within the United States, such as city or state names. This could be expanded, however, to include other locale data, such as specific sites or landmarks (e.g. "... near the Eiffel Tower") or relative locations (e.g. "... by the public library").

Our work has focused on determining whether a particular query is localizable or not. Once these queries are identified, a next logical step is to evaluate techniques for integrating the classifier into an information retrieval system. With proper indexing, it should be possible to compute the feature scores and classify a user query in realtime. Once a decision to localize has been made, the system must determine the proper degree of localization. For example, should the query be localized to the state or city level? Our tagging process maintains information about the specific locations which occur with each query, making this data readily available.

The features discussed in section 6 were selected based on the static nature of the available query log data. With a "live" system, we may design relevance experiments to collect more dynamic features for use in classification. Incorporating a mix of localized and non-localized results for a user query and measuring user activity (e.g. clickthoughs) may be used to evaluate user preferences, and act as a feedback loop to future iterations of the classification algorithm to further improve precision.

## 9. ACKNOWLEDGEMENTS

This work is partially supported by NSF grants, IIS-0534784 and IIS-0347993. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding institutions.

## 10. REFERENCES

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, November 1995.

[2] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART: TREC 3. In *Text REtrieval Conference*, pages 0–, 1994.

[3] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[4] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[5] C.-K. Huang, L.-F. Chien, and Y.-J. Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. *J. Am. Soc. Inf. Sci. Technol.*, 54(7):638–649, 2003.

[6] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36(2):207–227, 2000.

[7] G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 271–279, New York, NY, USA, 2003. ACM Press.

[8] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[9] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, New York, NY, USA, 2005. ACM Press.

[10] G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI '95)*, pages 338–345, 1995.

[11] Y.-H. Kim, S.-Y. Hahn, and B.-T. Zhang. Text filtering by boosting naive bayes classifiers. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 168–175, New York, NY, USA, 2000. ACM.

[12] R. Kraft, C. C. Chang, F. Maghoul, and R. Kumar. Searching with context. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 477–486, New York, NY, USA, 2006. ACM Press.

[13] R. Kraft and J. Zien. Mining anchor text for query refinement, 2004.

[14] T. Lau and E. Horvitz. Patterns of search: Analyzing and modeling web query refinement. In A. Press, editor, *Proceedings of the Seventh International Conference on User Modeling*, 1999.

[15] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *WWW2005*, 2005.

[16] Y.-S. Lee. Morphological analysis for statistical machine translation. In D. M. Susan Dumais and S. Roukos, editors, *HLT-NAACL 2004: Short Papers*, pages 57–60, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[17] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 558–565, New York, NY, USA, 2002. ACM Press.

[18] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: rapid prototyping for complex data mining tasks. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, 2006. ACM.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[20] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. *The First International Conference on Scalable Information Systems*, June 2006.

[21] M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997.

[22] A. Pretschner and S. Gauch. Ontology based personalized search. In *ICTAI '99: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, page 391, Washington, DC, USA, 1999. IEEE Computer Society.

[23] I. Rish. An empirical study of the naive bayes classifier. In *IJCAI-01 workshop on "Empirical Methods in AI"*, 2001.

[24] M. Speretta and S. Gauch. Personalized search based on user search histories. In *WI '05: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 622–628, Washington, DC, USA, 2005. IEEE Computer Society.

[25] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11, New York, NY, USA, 1996. ACM Press.