Coupling Feature Selection and Machine Learning Methods for Navigational Query Identification

Yumao Lu Fuchun Peng* Xin Li Nawaaz Ahmed
Yahoo! Inc.
701 First Avenue
Sunnyvale, California 94089
{yumaol, fuchun, xinli, nawaaz}@yahoo-inc.com

ABSTRACT

It is important yet hard to identify navigational queries in Web search due to a lack of sufficient information in Web queries, which are typically very short. In this paper we study several machine learning methods, including naive Bayes model, maximum entropy model, support vector machine (SVM), and stochastic gradient boosting tree (SGBT), for navigational query identification in Web search. To boost the performance of these machine techniques, we exploit several feature selection methods and propose coupling feature selection with classification approaches to achieve the best performance. Different from most prior work that uses a small number of features, in this paper, we study the problem of identifying navigational queries with thousands of available features, extracted from major commercial search engine results, Web search user click data, query log, and the whole Web's relational content. A multi-level feature extraction system is constructed.

Our results on real search data show that 1) Among all the features we tested, user click distribution features are the most important set of features for identifying navigational queries. 2) In order to achieve good performance, machine learning approaches have to be coupled with good feature selection methods. We find that gradient boosting tree, coupled with linear SVM feature selection is most effective. 3) With carefully coupled feature selection and classification approaches, navigational queries can be accurately identified with **88.1%** F1 score, which is **33%** error rate reduction compared to the best uncoupled system, and **40%** error rate reduction compared to a well tuned system without feature selection.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Experimentation

Keywords

Navigational Query Classification, Machine Learning

1. INTRODUCTION

Nowadays, Web search has become the main method for information seeking. Users may have a variety of intents while performing a search. For example, some users may already have in mind the site they want to visit when they type a query; they may not know the URL of the site or may not want to type in the full URL and may rely on the search engine to bring up the right site. Yet others may have no idea of what sites to visit before seeing the results. The information they are seeking normally exists on more than one page.

Knowing the different intents associated with a query may dramatically improve search quality. For example, if a query is known to be navigational, we can improve search results by developing a special ranking function for navigational queries. The presentation of the search results or the user-perceived relevance can also be improved by only showing the top results and reserving the rest of space for other purposes since users only care about the top result of a navigational query. According to our statistics, about 18% of queries in Web search are navigational (see Section 6). Thus, correctly identifying navigational queries has a great potential to improve search performance.

Navigational query identification is not trivial due to a lack of sufficient information in Web queries, which are normally short. Recently, navigational query identification, or more broadly query classification, is drawing significant attention. Many machine learning approaches that have been used in general classification framework, including naive Bayes classifier, maximum entropy models, support vector machines, and gradient boosting tree, can be directly applied here. However, each of these approaches has its own advantages that suit certain problems. Due to the characteristics of navigational query identification (more to be addressed in Section 2), it is not clear which one is the best for the task of navigational query identification. Our first contribution in this paper is to evaluate the effectiveness of these machine learning approaches in the context of navigational query identification. To our knowledge, this paper is the very first attempt in this regard.

*Dr. Peng contributes to this paper equally as Dr. Lu.

CIKM'06, November 5–11, 2006, Arlington, Virginia, USA.

Machine learning models often suffer from the curse of feature dimensionality. Feature selection plays a key role in many tasks, such as text categorization [18]. In this paper, our second contribution is to evaluate several feature selection methods and propose coupling feature selection with classification approaches to achieve the best performance: ranking features by using one algorithm before another method is used to train the classifier. This approach is especially useful when redundant low quality heterogeneous features are encountered.

Most previous studies in query identification are based on a small number of features that are obtained from limited resources [12]. In this paper, our third contribution is to explore thousands of available features, extracted from major commercial search engine results, user Web search click data, query log, and the whole Web's relational content. To obtain most useful features, we present a three level system that integrates feature generation, feature integration, and feature selection in a pipe line.

The system, after coupling features selected by SVM with a linear kernel and stochastic gradient boosting tree as classification training method, is able to achieve an average performance of **88.1%** F1 score in a five fold cross-validation.

The rest of this paper is organized as follows. In the next section, we will define the problem in more detail and describe the architecture of our system. We then present a multi-level feature extraction system in Section 3. We describe four classification approaches in Section 4 and three feature selection methods in Section 5. We then conduct extensive experiments on real search data in Section 6. We present detailed discussions in Section 7. We discuss some related work in Section 8. Finally, we conclude the paper in Section 9.

2. PROBLEM DEFINITION

We divide queries into two categories: navigational and informational. According to the canonical definition [3, 14], a query is navigational if a user already has a Web-site in mind and the goal is simply to reach that particular site. For example, if a user issues query "amazon", he/she mainly wants to visit "amazon.com". This definition, however, is rather subjective and not easy to formalize. In this paper, we extend the definition of navigational query to a more general case: a query is navigational if it has one and only one *perfect* site in the result set corresponding to this query. A site is considered as *perfect* if it contains complete information about the query and lacks nothing essential.

In our definition, navigational query must have a corresponding result page that conveys perfectness, uniqueness, and authority. Unlike Broder's definition, our definition does not require the user to have a site in mind. This makes data labeling more objective and practical. For example, when a user issues a query "Fulton, NY", it is not clear if the user knows the Web-site "www.fultoncountyny.org". However, this Web-site has an unique authority and perfect content for this query and therefore the query "Fulton, NY" is labeled as a navigational query. All non-navigational queries are considered informational. For an informational query, typically there exist multiple excellent Web-sites corresponding to the query that users are willing to explore.

To give another example, in our dataset, query "national earth science teachers association" has only one perfect corresponding URL "http://www.nestanet.org/" and therefore is labeled as navigational query. Query "Canadian gold maple leaf" has several excellent corresponding URL's, including "http://www. goldfingercoin.com/ catalog_gold/ canadian_maple_leaf.htm", "http://coins.about.com/ library/weekly/ aa091802a.htm" and "http://www.onlygold.com/Coins/ CanadianMapleLeafsFullScreen.asp". Therefore, query "Canadian gold maple leaf" is labeled as non-navigational query.

Figure 1 illustrates the architecture of our navigational query identification system. A search engine takes in a query and returns a set of URLs. The query and returned URLs are sent into a multi-level feature extraction system that generates and selects useful features; details are presented in the next section. Selected features are then input into a machine learning tool to learn a classification model.

3. MULTI-LEVEL FEATURE EXTRACTION

The multiple level feature system is one of the unique features of our system. Unlike prior work with a limited number of features or in a simulated environment [11, 12], our work is based on real search data, a major search engine's user click information and a query log. In order to handle large amount of heteorgeneous features in an efficient way, we propose a multi-level feature system. The first level is the feature generation level that calculates statistics and induces features from three resources: a click engine, a Web-map and a query log. The second level is responsible for integrating query-URL pair-wise features into query features by applying various functions. The third level is a feature selection module, which ranks features by using different methods. Below we present the details of the first two levels. The third level will be presented separately in Section 5 since those feature selection methods are standard.

3.1 Feature Generation

Queries are usually too short and lack sufficient context to be classified. Therefore, we have to generate more features from other resources. We use three resources to generate features: a click engine, a Web-map, and query logs. The click engine is a device to record and analyze user click behavior. It is able to generate hundreds of features automatically based on user click through distributions [16]. A Web-map can be considered as a relational database that stores hundreds of induced features on page content, anchor text, hyperlink structure of webpages, including the inbound, outbound URLs, and etc. Query logs are able to provide bag-of-words features and various language model based features based on all the queries issued by users over a period of time.

Input to feature generation module is a query-URL pair. For each query, the top 100 ULRs are recorded and 100 query-URLs are generated. Thus for each query-URL pair, we record a total of 197 features generated from the following four categories:

- Click features: Click features record the click information about a URL. We generate a total number of 29 click features for each query-URL pair. An example of a click feature is the click ratio (CR). Let n_k^i denote the number of clicks on URL k for query i and total number of clicks

$$n^i = \sum_k n_k^i.$$

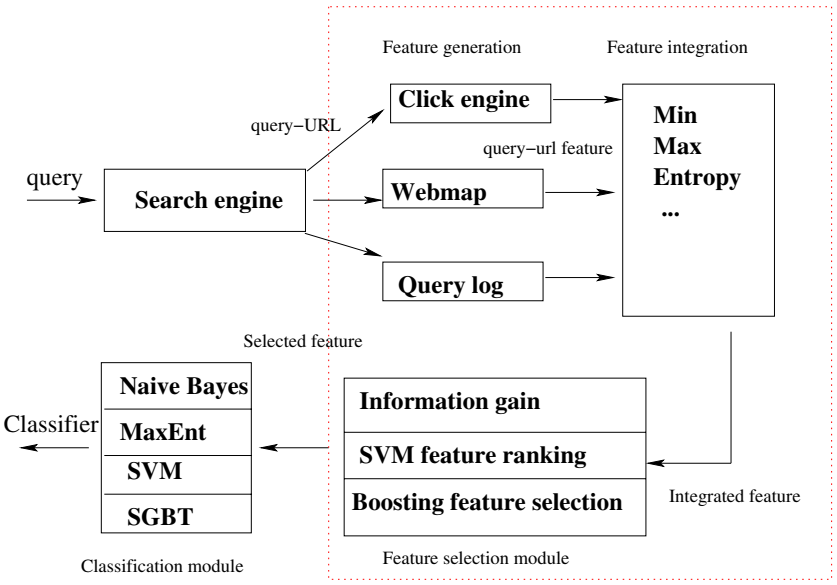

Figure 1: Diagram of Result Set Based Navigational Query Identification System

The click ratio is the ratio of number of clicks on a particular URL K for query i to the total number of clicks for this query, which has the form

$$CR(i, K) = \frac{n_K^i}{n^i}.$$

- URL features: URL features measure the characteristics of the URL itself. There are 24 URL based features in total. One such feature is a URL match feature, named *urlmr*, which is defined as

$$\text{urlmr} = \frac{l(p)}{l(u)}$$

where $l(p)$ is the length of the longest substring p of the query that presents in the URL and $l(u)$ is the length of the URL u. This feature is based on the observation that Web-sites tend to use their names in the URL's. The distributions confers uniqueness and authority.

- Anchor text features: Anchor text is the visible text in a hyperlink, which also provides useful information for navigational query identification. For example, one anchor text feature is the entropy of anchor link distribution [12]. This distribution is basically the histogram of inbound anchor text of the destination URL. If an URL is pointed to by the same anchor texts, the URL is likely to contain perfect content. There are many other anchor text features that are calculated by considering many factors, such as edit distance between query and anchor texts, diversity of the hosts, etc. In total, there are 63 features derived from anchor text.

Since we record the top 100 results for each query and each query URL pair has 197 features, in total there are 19,700 features available for each query. Feature reduction becomes necessary due to curse of dimensionality [5]. Before applying feature selection, we conduct a feature integration procedure that merges redundant features.

3.2 Feature Integration

We design a feature integration operator, named normalized ratio r_k of rank k, as follows:

$$r_k(f_j) = \frac{\max(f_j) - f_{jk}}{\max(f_j) - \min(f_j)}, k = 2, 5, 10, 20. \quad (1)$$

The design of this operator is motivated by the observation that the values of query-URL features for navigational query and informational query decrease at different rates. Taking the *urlmr* feature for example and considering a navigational query "Walmart" and an informational query "Canadian gold maple leaf", we plot the feature values of top 100 URLs for both queries, as shown in Figure 2. As we can see, the feature value for the navigational query drops quickly to a stable point, while an information query is not stable. As we will see in the experiment section, this operator is most effective in feature reduction.

Besides this operator, we use other statistics for feature integration, including mean, median, maximum, minimum, entropy, standard deviation and value in top five positions of the result set query-URL pair features. In total, we now have 15 measurements instead of 100 for the top 100 URLs for each query. Therefore, for each query, the dimension of a feature vector is $m = 15 \times 197 = 2955$, which is much smaller than $197,000$.

4. CLASSIFICATION METHODS

We apply the most popular generative (such as naive Bayes method), descriptive (such as Maximum Entropy method), and discriminative (such as support vector machine and stochastic gradient boosting tree) learning methods [19] to attack the problem.

4.1 Naive Bayes Classifier

A simple yet effective learning algorithm for classification

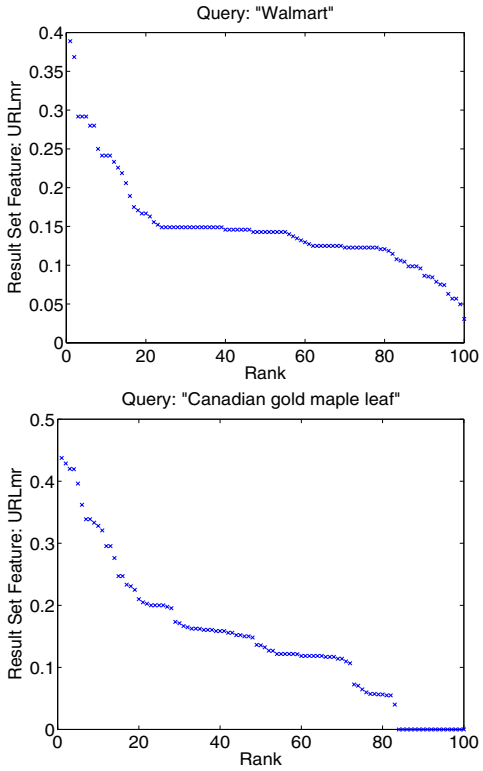

Figure 2: *urlmr* **query-URL feature for navigational query (upper) and a informational query (lower)**

is based on a simple application of *Bayes' rule*

$$P(y|q) = \frac{P(y) \times P(q|y)}{P(q)} \quad (2)$$

In query classification, a query q is represented by a vector of K attributes $q = (v_1, v_2, ...v_K)$. Computing $p(q|y)$ in this case is not trivial, since the space of possible documents $q = (v_1, v_2, ...v_K)$ is vast. To simplify this computation, the naive Bayes model introduces an additional assumption that all of the attribute values, v_j, are independent given the category label, c. That is, for $i \neq j$, v_i and v_j are conditionally independent given q. This assumption greatly simplifies the computation by reducing Eq. (2) to

$$P(y|q) = P(y) \times \frac{\prod_{j=1}^{K} P(v_j|y)}{P(q)} \quad (3)$$

Based on Eq. (3), a maximum a posteriori (MAP) classifier can be constructed by seeking the optimal category which maximizes the posterior $P(c|d)$:

$$y^* = \arg\max_{y \in Y} \left\{ P(y) \times \prod_{j=1}^{K} P(v_j|y) \right\} \quad (4)$$

$$= \arg\max_{y \in Y} \left\{ \prod_{j=1}^{K} P(v_j|y) \right\} \quad (5)$$

Eq. (5) is called the *maximum likelihood* naive Bayes classifier, obtained by assuming a uniform prior over categories.

To cope with features that remain unobserved during training, the estimate of $P(v_j|y)$ is usually adjusted by Laplace smoothing

$$P(v_j|y) = \frac{N_j^y + a_j}{N^y + a} \quad (6)$$

where N_j^y is the frequency of attribute j in D^y, $N^y = \sum_j N_j^y$, and $a = \sum_j a_j$. A special case of Laplace smoothing is *add one* smoothing, obtained by setting $a_j = 1$. We use add one smoothing in our experiments below.

4.2 Maximum Entropy Classifier

Maximum entropy is a general technique for estimating probability distributions from data and has been successfully applied in many natural language processing tasks. The over-riding principle in maximum entropy is that when nothing is known, the distribution should be as uniform as possible, that is, have maximal entropy [9]. Labeled training data are used to derive a set of constraints for the model that characterize the class-specific expectations for the distribution. Constraints are represented as expected values of features. The improved iterative scaling algorithm finds the maximum entropy distribution that is consistent with the given constraints. In query classification scenario, maximum entropy estimates the conditional distribution of the class label given a query. A query is represented by a set of features. The labeled training data are used to estimate the expected value of these features on a class-by-class basis. Improved iterative scaling finds a classifier of an exponential form that is consistent with the constraints from the labeled data.

It can be shown that the maximum entropy distribution is always of the exponential form [4]:

$$P(y|q) = \frac{1}{Z(q)} \exp(\sum_i \lambda_i f_i(q; y))$$

where each $f_i(q; y)$ is a feature, λ_i is a parameter to be estimated and $Z(q)$ is simply the normalizing factor to ensure a proper probability: $Z(q) = \sum_y exp(\sum_i \lambda_i fi(q; y))$. Learning of the parameters can be done using generalized iterative scaling (GIS), improved iterative scaling (IIS), or quasi-Newton gradient-climber [13].

4.3 Support Vector Machine

Support Vector Machine (SVM) is one of the most successful discriminative learning methods. It seeks a hyperplane to separate a set of positively and negatively labeled training data. The hyperplane is defined by $w^T x + b = 0$, where the parameter $w \in \mathbf{R}^m$ is a vector orthogonal to the hyperplane and $b \in \mathbf{R}$ is the bias. The decision function is the hyperplane classifier

$$H(x) = \text{sign}(w^T x + b).$$

The hyperplane is designed such that $y_i(w^T x_i + b) \geq 1 - \xi_i, \forall i = 1, ..., N$, where $x_i \in \mathbf{R}^m$ is a training data point and $y_i \in \{+1, -1\}$ denotes the class of the vector x_i. The margin is defined by the distance between the two parallel hyperplanes $w^T x + b = 1$ and $w^T x + b = -1$, i.e. $2/||w||_2$. The margin is related to the generalization of the classifier [17]. The SVM training problem is defined as follows:

$$\begin{array}{ll} \text{minimize} & (1/2)w^T w + \gamma \mathbf{1}^T \xi \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 - \xi_i,\ i = 1, ..., N \\ & \xi \geq 0 \end{array} \quad (7)$$

where the scalar γ is called the regularization parameter, and is usually empirically selected to reduce the testing error rate.

The basic SVM formulation can be extended to the nonlinear case by using nonlinear kernels. Interestingly, the complexity of an SVM classifier representation does not depend on the number of features, but rather on the number of support vectors (the training examples closest to the hyperplane). This property makes SVMs suitable for high dimensional classification problems [10]. In our experimentation, we use a linear SVM and a SVM with radial basis kernel.

4.4 Gradient Boosting Tree

Like SVM, gradient boosting tree model also seeks a parameterized classifier. It iteratively fits an additive model [8]

$$f_t(x) = T_t(x;\Theta_0) + \lambda \sum_{t=1}^{T} \beta_t T_t(x;\Theta_t),$$

such that certain loss function $L(y_i, f_T(x+i)$ is minimized, where $T_t(x;\Theta_t)$ is a tree at iteration t, weighted by parameter β_t, with a finite number of parameters, Θ_t and λ is the learning rate. At iteration t, tree $T_t(x;\beta)$ is induced to fit the negative gradient by least squares. That is

$$\hat{\Theta} := \arg\min_{\beta} \sum_{i}^{N} (-G_{it} - \beta_t T_t(x_i);\Theta)^2,$$

where G_{it} is the gradient over current prediction function

$$G_{it} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{t-1}}.$$

The optimal weights of trees β_t are determined

$$\beta_t = \arg\min_{\beta} \sum_{i}^{N} L(y_i, f_{t-1}(x_i) + \beta T(x_i,\Theta)).$$

If the L-2 loss function $[y_i - f(x_i)]^2/2$ is used, we have the gradient $G(x_i) = -y_i + f(x_i)$. In this paper, the Bernoulli loss function

$$-2\sum_{i}(y_i f(x_i) - log(1+\exp(f(x_i))))$$

is used and the gradient has the form

$$G(x_i) = y_i - \frac{1}{1+\exp(-f(x_i))}.$$

During each iteration of gradient boosting, the feature space is further partitioned. This kind of rectangular partition does not require any data preprocessing and the resulting classifier can be very robust. However, it may suffer from the dead zoom phenomenon, where prediction is not able to change with features, due to its discrete feature space partition. Friedman (2002) found that it helps performance by sampling uniformly without replacement from the dataset before estimating the next gradient step [6]. This method was called stochastic gradient boosting.

5. FEATURE SELECTION

Many methods have been used in feature selection for text classification, including information gain, mutual information, document frequency thresholding, and Chi-square statistics. Yang and Pedersen [18] gives a good comparison of these methods. Information gain is one of the most effective methods in the context of text categorization. In addition to information gain, we also use feature selection methods based on SVM's feature coefficients and stochastic gradient boosting tree's variable importance.

5.1 Information Gain

Information gain is frequently used as a measure of feature goodness in text classification [18]. It measures the number of bits of information obtained for category prediction by knowing the presence or absence of a feature. Let $y_i : i = 1..m$ be the set of categories, information gain of a feature f is defined as

$$\begin{aligned} IG(f) = &-\sum_{i=1}^{m} P(y_i) log P(y_i) \\ &+ P(f)\sum_{i=1}^{m} P(y_i|f) log P(y_i|f) \\ &+ P(\overline{f})\sum_{i=1}^{m} P(y_i|\overline{f}) log P(y_i|\overline{f}) \end{aligned}$$

where $\overline{f}$ indicates f is not present. We compute the information gain for each unique feature and select top ranked features.

5.2 Linear SVM Feature Ranking

Linear SVM (7) produces a hyperplane as well as a normal vector w. The normal vector w serves as the slope of the hyperplane classifier and measures the relative importance that each feature contribute to the classifier. An extreme case is that when there is only one feature correlated to sample labels, the optimal classifier hyperplane must be perpendicular to this feature axle.

The L-2 norm of w, in the objective, denotes the inverse margin. Also, it can be viewed as a Gaussian prior of random variable w. Sparse results may be achieved by assuming a laplace prior and using the L-1 norm [2].

Unlike the previous information gain method, the linear SVM normal vector w is not determined by the whole body of training samples. Instead, it is determined by an optimally determined subset, support vectors, that are *critical* to be classified. Another difference is obvious: normal vector w is solved jointly by all features instead of one by one independently.

Our results show that linear SVM is able to provide reasonably good results in feature ranking for our navigational query identification problem even when the corresponding classifier is weak.

5.3 Stochastic Gradient Boosting Tree

Boosting methods construct weak classifiers using subsets of features and combines them by considering their predication errors. It is a natural feature ranking procedure: each feature is ranked by its related classification errors.

Tree based boosting methods approximate relative influence of a feature x^j as

$$J_j^2 = \sum_{\text{splits on } x^j} I_k^2$$

where I_k^2 is the empirical improvement by k-th splitting on x^j at that point.

Unlike the information gain model that considers one feature at a time or the SVM method that considers all the feature at one time, the boosting tree model considers a set of features at a time and combines them according to their empirical errors.

Let $R(\mathcal{X})$ be a feature ranking function based on data set $\mathcal{X}$. Information gain feature ranking depends on the whole training set $R_{\text{Info}}(\mathcal{X}) = R_{\text{Info}}(\mathcal{X}_{\text{tr}})$. Linear SVM ranks features is based on a set of optimally determined dataset. That is, $R_{\text{SVM}}(\mathcal{X}) = R_{\text{SVM}}(\mathcal{X}_{\text{SV}})$, where $\mathcal{X}_{\text{SV}}$ is the set of support vectors. The stochastic gradient boosting tree (GSBT) uses multiple randomly sampled data to induce trees and ranks feature by their linear combination. Its ranking function can be written as $R_{\text{SGBT}}(\mathcal{X}) = \sum_{t=1}^{T} \beta_t R_{\text{SGBT}}^t(\mathcal{X}_t)$, where $\mathcal{X}_t$ is the training set randomly sampled at iteration t.

6. EXPERIMENTS

6.1 Data Set

A total number of 2102 queries were uniformly sampled from a query log over a four month period. The queries were sent to four major search engines, including Yahoo, Google, MSN, and Ask. The top 5 URL's returned by each search engine were recorded and sent to trained editors for labeling (the number 5 is just an arbitrary number we found good enough to measure the quality of retrieval). If there exists one and only one perfect URL among all returned URLs for a query, this query is labeled as navigational query. Otherwise, it is labeled as non-navigational query.

Out of 2102 queries, 384 queries are labeled as navigational. Since they are uniformly sampled from a query log, we estimate there are about 18% queries are navigational.

The data set were divided into five folders for the purpose of cross-validation. All results presented in this section are average testing results in five fold cross validations.

6.2 Evaluation

Classification performance is evaluated using three metrics: precision, recall and F1 score. In each test, Let n_{++} denotes the number of positive samples that correctly classified (true positive); n_{-+} denotes the number of negative samples that are classified as positive (false positive); n_{+-} denotes the number of false positive samples that are classified as negative (false negative); and n_{--} denotes the number of negative samples that are correctly classified (true negative). Recall is the ratio of the number of true positives to the total number of positives samples in the testing set, namely

$$\text{recall} = \frac{n_{++}}{n_{++} + n_{+-}}.$$

Precision is the ratio of the number of true positive samples to the number samples that are classified as positive, namely

$$\text{precision} = \frac{n_{++}}{n_{++} + n_{-+}}.$$

F1 is a single score that combines precision and recall, defined as follows:

$$F1 = \frac{2 \times \text{precsion} \times \text{recall}}{\text{precsion} + \text{recall}}.$$

6.3 Results

6.3.1 Feature Selection Results

Table 1 shows the distributions of the top 50 features selected by different methods. All methods agree that click features are the most important. In particular, linear SVM and boosting tree select more click features than information gain. On the other hand, information gain select many features from anchor text and other metrics such as spam scores.

Table 1: Distributions of the Selected Top 50 Features According to Feature Categories

Feature Set	Info. Gain	Linear SVM	Boosting
Click	52%	84%	74%
URL	4%	2%	6%
Anchor Text	18%	2%	12%
Other metrics	26%	12%	8%

Table 2 shows the distribution of the selected features according to feature integration operators. It shows which operators applied to result set query-URL pair wise features are most useful. We group the 15 operators into 5 types: vector, normalized ratios ($r_k, k = 2, 5, 10, 20$), min/max, entropy/stand deviation, and median/mean. Vector group includes all query-URL pair features in top 5 positions; normalized ratios are defined in (1). As we can see from the table, all feature integration operators are useful.

Table 2: Distributions of the Selected Top 50 Features According to Integration Operators

Operators	Info. Gain	Linear SVM	Boosting
vector	40%	22%	28%
normalized ratios	8%	38%	22%
min/max	6%	20%	16%
entropy/std	20%	16%	18%
mean/median	26%	4%	16%

The number of selected features directly influence the classification performance. Figure 3 shows relationship between the boosting tree classification performance and the number of selected features. As we can see, performance increases with cleaner selected features. However, if the number of selected feature is too small, performance will decrease. A number of 50 works the best in our work.

6.3.2 Classification Results

We first apply four different classification methods: naive Bayes, maximum entropy methods, support vector machine and stochastic gradient boosting tree model over all available features. The results are reported in Table 3. As we can see, stochastic gradient boosting tree has the best performance with an F1 score of **0.78**.

We then apply those methods to machine selected features. We test 4 different feature sets with 50 number of features, selected by information gain, linear SVM and boosting tree. The combined set consists of 30 top features selected by linear SVM and 29 top features selected by boosting tree. Please note that the total number of features are still 50 since linear SVM and boosting tree selected 9 same features in their top 30 feature set.

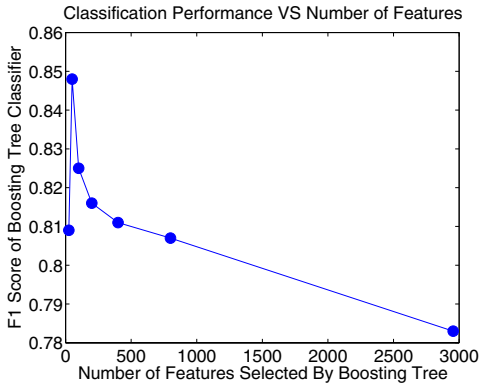

Figure 3: Classification performance F1 against number of features: 25, 50, 100, 200, 400, 800, and 2955 (all features)

Table 3: Results of Various Classification Methods over All Features

	Recall	Precision	F1
Naive Bayes	0.242	0.706	0.360
SVM (Linear Kernel)	0.189	1.000	0.318
Maximum Entropy	0.743	0.682	0.711
SVM (RBF Kernel)	0.589	0.485	0.528
Boosting Trees	0.724	0.845	0.780

Table 4 presents the results of the coupled feature selection and classification methods. It is obvious that the performance of each method is improved by applying them to machine selected *clean* features, except naive Bayes classifier. Surprisingly, the features selected by linear SVM are the best set of features. The results show that even if the underlying problem is not linear separable, the linear coefficients of the large margin linear classifier still convey important feature information. When the stochastic gradient boosting tree is applied over this set of features, we get the best performance with **0.881** F1 score among all cross-methods evaluations. Without feature ablation, SGBT is only able to achieve 0.738 F1 score. That is, feature selection has an effect of error reduction rate **40%**. Without introducing linear SVM in feature ablation, if SGBT works on the feature set selected by its own variable importance ranking, it achieves **0.848** F1 score. That is to say, a cross methods coupling of feature selection and classification causes a **33%** error reduction.

7. DISCUSSION

An interesting result from Table 1 is the features selected for navigational query identification. Those features are mostly induced from user click information. This is intuitively understandable because if a query is navigational, the navigational URL is the most clicked one. On the other hand, it might be risky to completely rely on click information. The reasons might be 1) user click features may be easier to be spammed, and 2) clicks are often biased by various presentation situation such as quality of auto abstraction, etc.

From Table 4, we observe that linear SVM and boosting tree have better feature selection power than information gain. The reason that information gain performs inferior to linear SVM and boosting tree is probably due to the fact that information gain considers each feature independently while linear SVM considers all features jointly and boosting tree composites feature rank by sum over all used features. The results show that URL, anchor text and other metrics are helpful only when they are considered jointly with click features.

The most important result is that the stochastic gradient boosting tree coupled with linear SVM feature selection method achieves much better results than any other combination. In this application, the data has very high dimension considering the small sample size. The boosting tree method needs to partition an ultra-high dimensional feature space for feature selection. However, the stochastic step does not have enough data to sample from [6]. Therefore, the boosted result might be biased by earlier sampling and trapped in a local optimum. Support vector machine, however, is able to find an optimally determined subset of training samples, namely support vectors, and ranks features based on those vectors. Therefore, the SVM feature selection step makes up the disadvantage of the stochastic boosting tree in its initial sampling and learning stages that may lead to a local optimum.

As expected, naive Bayes classifier hardly works for the navigational query identification problem. It is also the only classifier that performs worse with feature selection. Naive Bayes classifiers work well when the selected features are mostly orthogonal. However, in this problem, all features are highly correlated. On the other hand, classification methods such as boosting tree, maximum entropy model and SVM do not require orthogonal features.

8. RELATED WORK

Our work is closely related to query classification, a task of assigning a query to one or more categories. However, general query classification and navigational query identification are different in the problems themselves. Query classification focuses on content classification, thus the classes are mainly topic based, such as shopping and products. While in navigational query identification, the two classes are intent based.

In the classification approaches regard, our work is related to Gravano, et al. [7] where authors applied various classification methods, including linear and nonlinear SVM, decision tree and log-linear regression to classify query locality based on result set features in 2003. Their work, however, lacked carefully designed feature engineering and therefore only achieved a F1 score of 0.52 with a linear SVM. Beitzel, et al.[1] realized the limitation of a single classification method in their query classification problem and proposed a semi-supervised learning method. Their idea is to compose the final classifier by combining classification results of multiple classification methods. Shen, et al. [15] also trained a linear combination of two classifiers. Differently, instead of combining two classifiers for prediction, we couple feature selection and classification.

In the feature extraction aspect, our work is related to Kang and Kim 2003 [11] where authors extracted heterogenous features to classify user queries into three categories: topic relevance task, the homepage finding task and service finding task. They combined those features, for example URL feature and content feature, by several linear empiri-

Table 4: F1 Scores of Systems with Coupled Feature Selection and Classification Methods

Methods	Info. Gain	Linear SVM	Boosting	Combined Set
SVM (Linear Kernel)	0.124	0.733	0.712	0.738
Naive Bayes	0.226	0.182	0.088	0.154
Maximum Entropy	0.427	0.777	0.828	0.784
SVM (RBF Kernel)	0.467	0.753	0.728	0.736
Boosting Tree	0.627	**0.881**	0.848	0.834

cal linear functions. Each function was applied to a different binary classification problem. Their idea was to emphasize features for different classification purposes. However, the important features were not selected automatically and therefore their work is not applicable in applications with thousands of features.

9. CONCLUSION

We have made three contributions in the paper. First, we evaluate the effectiveness of four machine learning approaches in the context of navigational query identification. We find that boosting trees are the most effective one. Second, we evaluate three feature selection methods and propose coupling feature selection with classification approaches. Third, we propose a multi-level feature extraction system to exploit more information for navigational query identification.

The underlying classification problem has been satisfactorily solved with 88.1% F1 score. In addition to the successful classification, we successfully identified key features for recognizing navigational queries: the user click features. Other features, such as URL, anchor text, etc. are also important if coupled with user click features.

In future research, it is of interest to conduct cross methods co-training for the query classification problem to utilize unlabeled data, as there is enough evidence that different training methods may benefit each other.

10. REFERENCES

[1] S. Beitzel, E. Jensen, D. Lewis, A. Chowdhury, A. Kolcz, and O. Frieder. Improving Automatic Query Classification via Semi-supervised Learning. In *The Fifth IEEE International Conference on Data Mining*, pages 27–30, New Orleans, Louisiana, November 2005.

[2] C. Bhattacharyya, L. R. Grate, M. I. Jordan, L. El Ghaoui, and I. S. Mian. Robust Sparse Hyperplane Classifiers: Application to Uncertain Molecular Profiling Data. *Journal of Computational Biology*, 11(6):1073–1089, 2004.

[3] A. Broder. A Taxonomy of Web Search. In *ACM SIGIR Forum*, pages 3–10, 2002.

[4] S. della Pietra, V. della Pietra, and J. Lafferty. Inducing Features of Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4), 1995.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley, New York, NY, 2nd edition, 2000.

[6] J. H. Friedman. Stochastic Gradient Boosting. *Computational Statistics and Data Analysis*, 38(4):367–378, 2002.

[7] L. Gravano, V. Hatzivassiloglou, and R. Lichtenstein. Categorizing Web Queries According to Geographical Locality. In *ACM 12th Conference on Information and Knowledge Management (CIKM)*, pages 27–30, New Orleans, Louisiana, November 2003.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Predication*. Springer Verlag, New York, 2001.

[9] E. T. Jaynes. *Papers on Probability, Statistics, and Statistical Physics*. D. Reidel, Dordrecht, Holland and Boston and Hingham, MA, 1983.

[10] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, pages 137–142, Chemnitz, Germany, 1998.

[11] I.-H. Kang and G. Kim. Query Type Classification for Web Document Retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 64 – 71, Toronto Canada, July 2003.

[12] U. Lee, Z. Liu, and J. Cho. Automatic Identification of User Goals in Web Search. In *Proceedings of the 14th International World Wide Web Conference (WWW)*, Chiba, Japan, 2005.

[13] R. Malouf. A Comparison of Algorithms for Maximum Entropy Parameter Estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL)*, Taipei, China, 2002.

[14] D. E. Rose and D. Levinson. Understanding User Goals in Web Search. In *Proceedings of The 13th International World Wide Web Conference (WWW)*, 2004.

[15] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Q2C at UST: Our Winning Solution to Query Classification in KDDCUP 2005. *SIGKDD Explorations*, 7(2):100–110, 2005.

[16] L. Sherman and J. Deighton. Banner advertising: Measuring effectiveness and optimizing placement. *Journal of Interactive Marketing*, 15(2):60–64, 2001.

[17] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[18] Y. Yang and J. Pedersen. An Comparison Study on Feature Selection in Text Categorization. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in informaion retrieval*, Philadelphia, PA, USA, 1997.

[19] S.C. Zhu. Statistical modeling and conceptualization of visual patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):619–712, 2003.