Predicting Primary Categories of Business Listings for Local Search Ranking

Changsung Khan[a], Jeehaeng Lee[a], Roi Blanco[b], Yi Chang[a]

[a] *Yahoo Labs*
Sunnyvale, US
[b] *Yahoo Labs*
Barcelona, Spain

Abstract

We consider the problem of identifying primary categories of a business listing among the categories provided by the owner of the business, in order to enhance local search and browsing. The category information submitted by business owners cannot be trusted with absolute certainty since they may purposefully add some secondary or irrelevant categories to increase recall in local search results, which makes category search very challenging for local search engines. Thus, identifying primary categories of a business is a crucial problem in local search. This problem can be cast as a multi-label classification problem with a large number of categories. However, the large scale of the problem makes it infeasible to use conventional supervised-learning-based text categorization approaches.

We propose a large-scale classification framework that leverages multiple types of classification labels to produce a highly accurate classifier with fast training time. We effectively combine the complementary label sources to refine prediction. The experimental results indicate that our framework achieves very high precision and recall and outperforms a competitive baseline using a centroid-based method.

We also propose a new ranking feature based on the mapping of queries and documents to category space and show that the new feature leads to ranking relevance improvements for local search.

Email addresses: `ckang@yahoo-inc.com` (Changsung Khan), `jeehaeng@yahoo-inc.com` (Jeehaeng Lee), `roi@yahoo-inc.com` (Roi Blanco), `yichang@yahoo-inc.com` (Yi Chang)

Keywords: Vertical search, Text categorization, Primary category, Ranking Relevance

1. Introduction

Consumers accessing online information through their handset devices is one of the main reasons why local search is outgrowing Desktop search. Recent studies show that at least 20% of Web queries have some sort of local intent [31]. Local search queries usually respond to an instant informational need, and they are gaining traction among marketeers because they are issued by users deeper in the purchase funnel, who are more likely to make quick decisions. Local search intent spans different types, from a user wanting to find a shop nearby to learning the development of recent news events. As with generic vertical search, relevance is domain-specific and comprises many different well-defined aspects like relatedness, closeness, reputation among others. Different types of queries in local search pose different challenges for ranking, and the results quality yielded by search engines differ; for instance, category queries such as “Restaurants" are known to be harder than business name queries such as “Best Buy" [17].

In this paper we focus on the problem of business ranking. The main source of information are registered business listings which provide information of varying quality about a physical store. One of the challenges in category queries in local search is that category descriptions submitted by business owners are often incorrect. For instance, it is a common practice from owners registering their business to add some secondary or irrelevant categories in order to increase recall. For example, the owner of a Japanese restaurant may add “Korean Restaurants" to the category description of the business, in the hope that her store may appear in the search results for the query “Korean Restaurants" as well as for the query “Japanese restaurants". This motivates the problem of identifying *primary (or true) categories* of a business. This can be regarded as an multi-label classification problem [26] in which we can assign multiple primary categories to a business.

Text categorization has been an active field of research in the natural language processing and machine learning communities (See Section 2). The scenario just portrayed, however, differs from previous studies in that the large number of categories in local search (2000 categories in total) makes it impossible to use most of the conventional supervised-learning-based text categorization methods.

This work presents a solution to the large-scale primary category prediction (multi-label classification) problem by combining three complementary sources of information (business labels):

- Labels provided by human judges
- Labels provided by business owners
- Click signals provided by users in local search

The first step is to derive a set of highly predictive features from labels from business owners and click signals. Then, a classifier is trained from these features using labels by human judges as targets. These features are engineered using textual similarity features, clicks and exploiting the relationship between different categories of business listings. Experimental results demonstrate that integrating these multiple sources of labels is highly beneficial for a large-scale classification problem. In order to accommodate all the different sources of information, we develop an iterative algorithm that is able to discard incorrectly classified labels. We demonstrate that, under certain benign conditions such that the confidence of the classifier exceed a certain threshold, the algorithm improves classification accuracy. Furthermore, we make use of the category (taxonomy) structure in order to generate classifiers with a lower error rate.

A second main contribution of this paper is to develop a new rank feature that leverages our primary category classifier. The key idea of this new feature is to map queries and documents (listings) to category space and perform matching in the same category space. The outcome of an experiment using a large number of queries coming from a commercial search engine demonstrates that the derived classification features can directly improve relevance significantly.

The techniques developed here can be applied to learn highly effective ranking models for local search results, as well as serving as tools to provide browsing capabilities over faceted search results, which have been enriched with category information. This is crucial when the search is performed in an environment with limited display capabilities such a mobile device [11].

This paper is organized as follows. Section 2 summarizes previous work related to this paper. Section 3 presents our proposed approach for primary category prediction. In Section 4, we introduce a new rank feature based on the primary category classifier. Section 5 reports experimental results for primary category prediction and ranking. Finally, Section 6 concludes this paper.

2. Related Work

The main bulk of this work is closely related to text categorization. Textual categorization is the task of assigning a category label from predefined categories to a given document, comprised of textual features (as opposed to multimedia classification), and it has been a blooming field of interest in the machine learning and natural language processing/information retrieval communities for over 20 years.

The predominant approach for automatic classification is to machine learn a document representation using a hand-labelled partition of a document collection. There has been a plethora of machine learning approaches introduced for text categorization, such as k-Nearest Neighbors [23, 27], Naive Bayes [18], Decision Trees [33, 35], Neural Networks [29] Support Vector Machines (SVM) [1, 9, 15, 25] and Centroid Classifier [14, 20].

One of the key aspects of this is the document representation cost when the algorithms have to deal with large-scale training and test sets. Non-parametric data-based representations (like word-level unigrams or N-grams) require a large amount of computational resources to store the high amount low-level features. In our domain of interest (large scale multi-class classification) this overhead is apparent at training but more critically at training time. We employ a variant of the centroid classifier as our baseline [14], which can be seen as a specialization of Rocchio's method [24] (nearest centroid classifier), which is widely adopted in the IR literature. The algorithm computes one representation per class using the vector-space model, which corresponds to the centroid vector of all the positive training instances (documents) for that category. Some prior work has shown that this method provides comparable or superior performance to that of k-Nearest Neighbors and 2/3-Gram Naive Bayes classifiers [14].

Most of the above approaches rely on labels provided by human judges. In the context of web page classification and query classification, there have been efforts to leverage click-through data [7, 19, 21] to improve classification performance. Click-through data have been applied to other related problems such as Web search ranking systems [8, 16].

Cao et al. [7] introduce a conditional random field classifier that is able to classify search queries employing the information contained in user sessions (i.e., queries previously issued by the same user within a short time-span). Kim et al. [19] leverage query logs with a semi-supervised algorithm, which augments the training set by propagating labels from similar documents. Similarity is defined by the click-graph, this is, two documents are regarded similar when users click in both of them after issuing similar queries. Li et al. [21] present an overview of the KDD-cup query classification challenge, which describes systems and approaches for classifying 800K queries into 67 predefined classes. Joachims [16] presents an approach to optimize the retrieval effectiveness of search engines using click-through data, by learning a SVM. Similarly, Chapelle and Zang [8] present a method to infer editorial labels from query log and click information, using a Dynamic Bayesian Network that provides relevance estimates for clicked documents. Previous results have shown that categories are able to reduce the time spent by users in finding a particular search result [11]. Given that we also focus enhancing ranking with classification, in order to extract query-document features for ranking business listings, we need to map the user query into the category space in which the actual documents have been classified. Query classification has been a key area of development for improving the performance of search engines [22]. We highlight some works mostly related to ours. Broder et al. [5] show how matches between ad creatives and queries can be improved by an aggregation of classes of URLS returned as a query result list. Relatedly, Bennet et al. [2] show that category features can be beneficial for ranking Web search results, and experiment on a system that enriches the Web search index with metadata related with the category information. Blanco et al [3] propose a combination of a term- and a concept-based retrieval model that closes the semantic gap between queries and documents expanding both of them with category information. We depart from those works in several ways. The main focus of our work is to perform large-scale classification as well as improving retrieval performance, as the former is also important for providing browsing and filtering capabilities in the local search scenario. This context is challenging, given the large number of classes the algorithm has to deal with and also the domain is narrower than general Web search, implying that there is a high content overlap between classes. Note that this setting departs from other large-scale hierarchical text classification initiatives,[1] which target a larger number of classes (up to 300K) but

[1]http://lshtc.iit.demokritos.gr

the amount of data is smaller (2.5M documents maximum), and the main challenge is to overcome data-sparsity. Cissé et al. [10] assume that classes can be arranged within a pre-established hierarchy and propose a method that learns compact class codes using autoencoders which leverage a binary code defined on similarity assumed to be available over the different classes. Weston et al.Ê[34] present a general approach for converting an algorithm which has linear time in the size of the set of labels to a sublinear one via label partitioning, using a previously trained label scorer.

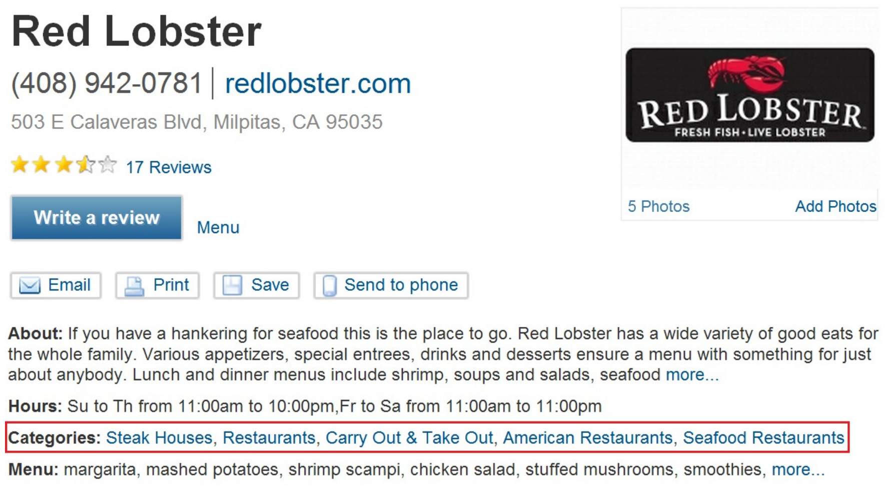
Our method also leverages different sources of labels, and learns to discard unreliable labels in an iterative fashion. Furthermore, we exploit category structure in order to enhance classification performance. In our work, we do not rely on a single label source such as editorial labels or click-through data. Instead, we aim at effectively combining multiple label sources to improve the performance of text categorization in the context of local search. All this highly predictive classification features are useful for improving the effectiveness of a state of the art learning to rank function.

3. Predicting Primary Categories

In this section, we propose a machine learning approach to identify primary categories of a business listing page.

3.1. Problem Formulation

Let $D = \{d_1, d_2, \ldots\}$ be the set of all business listing pages stored in a local search index and $C = \{c_1, c_2, \ldots\}$ be the set of all categories for local businesses, as defined by human editors. We follow a vector-space representation of business pages d, in which vector components are drawn from the set $T = \{t_1, t_2, \ldots\}$, comprising all terms appearing in D. The set of categories assigned to a business listing d is denoted by C_d. Categories are organized in a tree hierarchical manner, in which category nodes present in higher levels of the tree denote semantically broader concepts than nodes in lower levels. Formally, $c_i \succ c_j$ if there is a path in the tree encoding C from c_i to c_j. An example would be *food services* $\succ$ *restaurant* $\succ$ *steak house.* We denote by $D_c = \{d \mid c \in C_d, d \in D\}$ the set of all listings that have c assigned as one of its categories. A category $c \in C$ is a *primary category* of a business listing $d \in D$ if c represents a relevant label for the listing, denoting precisely the type of business d relates to, and there are not narrower relevant categories in C for d, this is, $\nexists\, c_i \in C$ s.t. $c \succ c_i$.


Red Lobster

(408) 942-0781 | redlobster.com

503 E Calaveras Blvd, Milpitas, CA 95035

17 Reviews

About: If you have a hankering for seafood this is the place to go. Red Lobster has a wide variety of good eats for the whole family. Various appetizers, special entrees, drinks and desserts ensure a menu with something for just about anybody. Lunch and dinner menus include shrimp, soups and salads, seafood more...

Hours: Su to Th from 11:00am to 10:00pm,Fr to Sa from 11:00am to 11:00pm

Categories: Steak Houses, Restaurants, Carry Out & Take Out, American Restaurants, Seafood Restaurants

Menu: margarita, mashed potatoes, shrimp scampi, chicken salad, stuffed mushrooms, smoothies, more...

Figure 1: An example of a business listing page. Categories are highlighted in a red box.

The primary category classification problem is posed as follows: given a business listing d and a category $c \in C_d$, determine whether c is a primary category of d.

Figure 1 shows an example of a business listing page. Based on the above definitions, C_d of this listing is the set {"Steak Houses", "Restaurants", "Carry Out & Take Out", "American Restaurants", "Seafood Restaurants"}. These categories are either provided by the owner of the business or a third-party information provider. In this example, only "Seafood Restaurants" is a primary category of the business and the other categories are deemed as secondary or irrelevant.

Note that this problem can be regarded as a multi-label text categorization problem since it is possible for a business to have multiple primary categories. Furthermore, we assume that there is at least one primary category for each business listing. This assumption is useful for performing feature normalization, which is discussed in Section 3.3.3.

3.2. Proposed Solution

The main challenge for our problem is the large number of categories (2000 categories in total), which makes it very difficult to apply conventional supervised-learning-based text categorization approaches. Firstly, most standard multi-class approaches (e.g. multi-class SVMs) decompose the problem

Table 1: Comparing different types of label sources.

Label sources	Accuracy	Coverage (no. of covered listings)	Usage
Cats. assigned by owners (C_d)	medium	high	feat. in Section 3.3.1
User clicks	high	medium	features in Section 3.3.2
Editorial labels	high	low	labels y in training & testing

into a number of binary independent classification tasks, and therefore have a $O(n^2)$ complexity in the number of classes n. Most importantly, obtaining enough labels to train a classifier for each category is indeed very expensive and not feasible in domains like ours. In this work we turn into leveraging some other types of pseudo labels to train the classifiers. Indeed, there are two types of such pseudo labels available for our problem. One is the category description C_d assigned to listings. Despite of the set C_d containing some categories that have been incorrectly assigned, a collection of listings within the same category convey valuable information about the category. The second source is comprised of the user clicks gathered from local search click logs. Category names (e.g., “chinese restaurants") are common queries in local search. User clicks on business listings in the search results provide important signals about the relationship between categories and business listings.

Our proposed solution is as follows. We first derive a set of features $\mathbf{x}$ to be employed by a classifier from the above two pseudo-label sources (discussed in detail in Section 3.3). Note that a feature vector $\mathbf{x}$ is defined for a business listing-category pair (d,c). For example, the click-through rate of d for c (from the search results when c is used as a query) is such a feature. Lastly, we train a classifier $f(d, c)$ using training data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots\}$ where y_i is a label provided by human judges.

Table 1 compares the three different types of category label sources leveraged in our solution. We refer to *accuracy* as the ratio between the number of correct categories assigned to listings over the number of assigned categories. Conversely, *coverage* refers to the average amount of categories assigned by business listing. It is apparent that the three sources complement one another in terms of accuracy and coverage. Since categories by owners and user clicks have large coverage, they are appropriate for being used as targets to generate features (discussed in Section 3.3 in detail). On the other hand, the editorial labels by human judges are very accurate although they are not enough to train a classifier for each class. Thus, we use it as the final learning targets to combine the features (in Section 3.4).

After we obtain a classifier $f(d,c)$, the categories in C_d can be adjusted by the classifier in the following manner. If a category $c \in C_d$ is classified as not being a primary one, it is eliminated from C_d. Since the actual categories belonging to C_d are used to generate features for the classifier, the updated C_d generates different values of the features. Then, we can train a new classifier based on the new feature values. Thus, it is possible to repeat this process to further improve prediction. This iterative method is described in Section 3.5.

3.3. Features

In this section, we discuss how features are derived. Note that each feature is defined for a business listing-category pair (d,c) to be used as a signal for our classifier $f(d,c)$.

3.3.1. Centroid-based Similarity Features

For each category $c \in C$, we define

$$Centroid_c = \sum_{d \in D_c} d \tag{1}$$

where d is a business listing represented as a tf-idf weight vector. In other words, $Centroid_c$ is the cumulated weight vector for all the listings that share the category c in their assigned categories. Then, we can compute the cosine similarity measure between a business listing d and the centroid vector $Centroid_c$ for a category c:

$$cosine\_sim(d,c) = \frac{d \cdot Centroid_c}{||d||\ ||Centroid_c||} \tag{2}$$

We use $cosine\_sim(d,c)$ as a feature for our classifier. The motivation for this features is as follows. Each centroid vector $Centroid_c$ is a mixture of the true distribution of terms for the category c and the noise due to errors in C_d. However, the true distribution dominates the centroid since the error rate of categories assigned by owners is low (around 10%). Also, it should be noted that even when the owner of a business may assign secondary or irrelevant categories to the category description part of the content (C_d) to increase recall for local search, it is less likely for the owner to corrupt the overall content of the business data (d). For example, “Korean Restaurants" can be easily added to the category description of a Japanese restaurant by the owner hoping that the business may appear in the search results for the query “Korean Restaurants" as well as for the query “Japanese restaurants". However, the owner would not add many Korean menu items to the content. Hence, we expect that $cosine\_sim(d, c)$ will be high when c is a primary category of d and low otherwise. Indeed, this hypothesis is verified in the experimental results which show that the centroid-based features are very effective for our classification problem.

To reduce the dimension of the document vectors and minimize the noise in the centroids, we propose another similarity feature based on new centroids generated by the χ^2 method [36]. The χ^2 statistic score $\chi^2(t, c)$ for a term t and a category c is

$$\frac{N(N_{r+}N_{n-} - N_{r-}N_{n+})^2}{(N_{r+} + N_{r-})(N_{n+} + N_{n-})(N_{r+} + N_{n+})(N_{r-} + N_{n-})}$$

where N_{r+} is the number of times t and c co-occur, N_{n+} is the number of times t occurs without c, N_{r-} is the number of times c occurs without t, N_{n-} is the number of times neither c nor t occurs, and N is the total number of documents. If the χ^2 statistic score for t is high, it belongs to characteristic vocabulary of c.

Using χ^2 statistic scores, we generate a reduced set of terms $T'_c = \{t \in T \mid \chi^2(t, c) \geq \alpha\}$ for each category c. Then, we generate a new centroid filtered by the reduced terms:

$$Centroid'_c = (M^c)(Centroid_c)$$

where M^c is a diagonal matrix with $M^c_{ii} = 1$ if $t_i \in T'_c$ and 0 otherwise. We obtain a new similarity feature:

$$cosine\_sim_{filtered\_tfidf}(d, c) = \frac{d \cdot Centroid'_c}{||d||\ ||Centroid'_c||} \tag{3}$$

Alternatively, we can directly use χ^2 statistic scores as weights for a filtered centroid. Let $Centroid''_c$ be a vector where the i-th element is $\chi^2(t_i, c)$ if $\chi^2(t_i, c) \geq \alpha$ and 0 otherwise. Then, we obtain another feature:

$$cosine\_sim_{filtered\_\chi^2}(d, c) = \frac{d \cdot Centroid''_c}{||d||\ ||Centroid''_c||}. \tag{4}$$

The complexity for computing this feature is $O(\sum_{d \in D} |C_d||T_d|)$ where $|T_d|$ is the number of terms in d. We note that this computation can be effortless implemented within a map-reduce distributed framework such Hadoop.

3.3.2. Click-based Features

In local search category queries such as “Restaurants" are very common. User clicks on business listings in the search results page provide crucial information about the relationship between the query and the clicked listings: The more clicks on a listing, the more likely the listing is to be about the query. The key observation is: When a query q matches a category name c, clicks on a listing d in the search results page for q can be translated into a positive relationship between d and c. In this section, the features for a category c are obtained from click statistics for c as a query in click logs.

The simplest form of a click-based feature is the click-through rate

$$CTR(d,c) = \frac{clicks}{views} \tag{5}$$

where clicks refers to the number of times the url of the listing d was clicked, and views the number of times d appeared in a result page when a user issued c as a query. It is well known that CTR suffers from the position bias, this is, the results at higher positions get more clicks regardless of their relevance. To address the position bias problem, we use the following two click measures in addition to CTR.

$$COEC(d,c) = \frac{\sum_{i=1}^{N} clicks_i}{\sum_{i=1}^{N} aCTR_{p_i}} \tag{6}$$

where $clicks_i \in \{0,1\}$ denotes if d was clicked in the i-th session out of N sessions in which d appeared for c, p_i is the position of d in the i-th session and $aCTR_p$ is the aggregated CTR (over all queries and sessions) for position p, this is $\sum_{i=1}^{S} clicks_i/views_i$, where $clicks_i$ and $views_i$ are the clicks and views on the position i on the search results pages, summed over all queries S.

$$SKIP\_CTR(d,c) = \frac{clicks}{clicks + skips} \tag{7}$$

where *skips* is the number of sessions in which d was not clicked but some other results below d were clicked. Note that SKIP_CTR is a good approximation of so-called *attractiveness*, defined to be the probability of a click on a document given that the document is examined by the user.

3.3.3. Adding Normalized Features

The hypothesis that there is at least one primary category for each d suggests that we need to consider the relationships among the categories in C_d. To this end, we propose to add a normalized feature $normalized(feature(d, c))$ for each feature $feature(d, c)$:

$$normalized(feature(d, c)) = \frac{feature(d, c)}{\max_{c' \in C_d} feature(d, c')}$$

For example, let $C_d = \{c_1, c_2\}$ be the category set for d and $CTR(d, c_1) = 0.4$, $CTR(d, c_2) = 0.2$. Then, $normalized$ $(CTR(d, c_1)) = 1$ and $normalized(CTR(d, c_2))$ $= 0.5$. The intuition behind this normalization is that the category $c_{max} = \text{argmax}_{c' \in C_d} feature(d, c')$ is likely to be a primary category regardless of its feature value according to the above hypothesis. The relative information provided by the normalized features combined with the original features increases the predictive power of our classifier because the deviation between among different examples (listings) might be very large. By adding the ratio of the feature over the maximum value in the example the classifier can learn to compare ratios across examples and not among distributions over numbers that have a large variance.

3.3.4. Exploiting Category Relationships

Many multi-label classification methods ignore the relationships among different categories. Those methods predict each label separately based on the probability decomposition:

$$p(y_i, \ldots, y_{|C|}|\mathbf{x}) = \prod_i p(y_i|\mathbf{x}).$$

This decomposition is possible due to the assumption that all labels are independent given a document.

However there are rich relationships among categories that can be leveraged to improve classification. In our problem we focus on how the top category (the most representative category among all primary categories for a listing) affects the probabilities of other categories. For example, consider the following categories “electronics retailers" $\succ$ { “computer software" , “music stores" }. We can assume that given knowledge of whether “electronics retailers" occurs, knowledge of whether “music stores" occurs provides no information on the likelihood of “electronics retailers" occurring, and vice-versa.

Therefore, we propose to relax the above dependence assumption and make y_i and y_j independent given the top category y^*. This leads to the probability decomposition:

$$\begin{aligned}&p(y_i, \ldots, y_{|C|} | top\_category = y^*, \mathbf{x}) \\ &= \prod_i p(y_i | top\_category = y^*, \mathbf{x}).\end{aligned}$$

As a result of the above decomposition, we can predict each label separately by modeling the probability as a function of the conditional probability of the label given the top category:

$$\begin{aligned}&p(y_i | top\_category = y^*, \mathbf{x}) \\ &= \hat{f}(\mathbf{x}, p(y_i | top\_category = y^*))\end{aligned}$$

where $\mathbf{x}$ is a vector of features that are introduced in this section and $\hat{f}$ is a classifier. The probability $p(y_i | top\_category = y^*)$ can be easily estimated if y^* is known by counting the number of listings that contain y^* also contain y_i. However, the problem is that we have to also predict y^*, even before we have $\hat{f}$. It turns out that the selection of the top category is not difficult and any reasonably good classifier (or ranker) can predict the top category accurately. Thus, we use one of the features as a function to select y^*. For example, $y^* = \arg\max_{c \in C_d} cosine\_sim(d, c)$.

3.4. Classifier Training

Given training data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots\}$, we use the gradient boosting method (GBDT) [13] to train a classifier $f(d, c)$. Each feature vector $\mathbf{x}$ consists of the features defined in the previous section: $\mathbf{x} = \{cosine\_sim(d, c)$, $normalized\ (cosine\_sim(d, c))$, $CTR(d, c)$, $normalized(CTR(d, c))$, $\ldots\}$.

The major difference between our framework and typical text categorization frameworks is that there are much fewer features in our framework but each feature is much stronger. Most classifiers for text categorization use a set of terms as features. On the other hand, we use a small number of very strong features for the classifier. Each feature used in our classifier can be even considered as a stand-alone *model*. In Section 5, we show that each feature performs reasonably well as a classifier. In this sense, the training step in our framework can be viewed as combining multiple *models* to improve prediction.

Algorithm 1 Iterative method for primary category prediction

```
Input: Listings $D$, Categories $C_d$ for each $d \in D$, Training data: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots\}$
Output: Filtered categories $C_d^n$ for each $d \in D$
 1: $C_d^0 = C_d$ for each $d \in D$
 2: $R_0 = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots\}$
 3: for $i = 0, \ldots, n-1$ do
 4:     Train a classifier $f_i(\mathbf{x})$ using $R_i$.
 5:     for each $d \in D$ do
 6:         $C_d^{i+1} = \{c | c \in C_d^i, f_i(\mathbf{x}) > \gamma\}$
 7:     end for
 8:     Obtain new training data $R_{i+1}$ using $\{C_d^{i+1} | d \in D\}$.
 9: end for
10: return $\{C_d^n | d \in D\}$
```

3.5. Iterative Method

The classifier that has been learned can be applied to not only new listings but also to the existing listings D to filter out irrelevant categories. After we obtain a classifier $f(d, c)$, the categories in C_d can be adjusted by the classifier: If a category $c \in C_d$ is classified as an irrelevant one, it is eliminated from C_d. Since the categories in C_d are used to generate features for the classifier, the updated C_d generates different values of the features. Then, we can train a new classifier based on the new feature values. Furthermore, it is possible to repeat this process to further improve prediction. We may have a situation in which all categories given by the owner are filtered out. In this case, we first augment the category set for the document by adding some related categories (based on the category-category relationships in the corpus) and then, we apply the classifier to the augmented category set.

Algorithm 1 formally describes the iterative method, which is similar in spirit to other algorithms that build centroid-based vector representations of learning examples [32, 28]. It repeats *Classifier training - Update centroids - Update features* cycles. The cycle may be repeated a fixed n times or the algorithm may stop when a certain condition is satisfied (e.g., if $C_d^{i+1} = C_d^i$ for all $d \in D$). The algorithm always stops, since the $d \in D$ have a finite number of categories and at every iteration the algorithm only removes categories, this is, $C_d^{i+1} \subseteq C_d^i$.

Under soft-conditions, the algorithm also converges and improves an up-per bound on the deviations of errors made by the classifier. Following [30], we study the properties of the maximal deviation of error frequencies, when some of the class labels employed by the learner are wrong. We limit this analysis to binary classifiers that minimize the empirical risk.

Let δ be the indicator function used for classification:

$$\delta(f, \mathbf{x}) = \begin{cases} 1 & \text{if } f(\mathbf{x} > \gamma) \\ 0 & \text{otherwise} \end{cases}$$

Let us assume a data set

$$Z^{2l} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_l, y_l), (\mathbf{x}_{l+1}, y_{l+1}) \dots (\mathbf{x}_{2l}, y_{2l})\} \tag{8}$$

of $2l$ labels, split into train and test halves. Let us denote by $\nu_1(Z^{2l})$ the frequency of errors in the first half of the data and $\nu_2(Z^{2l})$ the frequency of errors in the second half of the data:

$$\nu_1(f, Z^{2l}) = \frac{1}{l} \sum_{i=1}^{l} |y_i - \delta(f, \mathbf{x}_i)| \tag{9}$$

and

$$\nu_2(f, Z^{2l}) = \frac{1}{l} \sum_{i=l+1}^{2l} |y_i - \delta(f, \mathbf{x}_i)| \tag{10}$$

The maximal error deviation of a classifier learned from a class of functions $\mathcal{F}$ is [30]:

$$\sup_{f \in \mathcal{F}} (\nu_1(f, Z^{2l}) - \nu_2(f, Z^{2l})) \tag{11}$$

Now, let us consider a *corrupted* dataset in which half of the labels have been flipped $\overline{y}$ (in the case of binary classification $\overline{y}_i = 1 - y_i$), this is $y_i = \overline{y_i} \forall\ 1 < i \leq l$. The classifier will try to minimize the loss

$$L(f) = \frac{1}{l} \left(\sum_{i=1}^{l} (\overline{y_i} - \delta(f, \mathbf{x}_i))^2 + \sum_{i=l+1}^{2l} (y_i - \delta(f, \mathbf{x}_i))^2 \right) \tag{12}$$

Since we assume that labels and classification decisions are binary

$$(\overline{y_i} - \delta(f, \mathbf{x}_i))^2 = 1 - (y_i - \delta(f, \mathbf{x}_i))^2 \tag{13}$$

Then,

$$L(f) = 1 - \frac{1}{l}\sum_{i=1}^{l}(y_i - \delta(f, \mathbf{x}_i))^2 + \frac{1}{l}\sum_{i=l+1}^{2l}(y_i - \delta(f, \mathbf{x}_i))^2 \tag{14}$$

Minimizing $L(f)$ is equivalent to maximizing:

$$\frac{1}{l}\sum_{i=1}^{l}(y_i - \delta(f, \mathbf{x}_i))^2 - \frac{1}{l}\sum_{i=l+1}^{2l}(y_i - \delta(f, \mathbf{x}_i))^2 \tag{15}$$

which supremum is equivalent to that of

$$\nu_1(f, Z^{2l}) - \nu_2(f, Z^{2l}) \tag{16}$$

The maximum deviation of errors of the classifier is obtained when the set that contains half of the labels flipped is used for learning. The objective function in Eq. 15 can be computed for sets of varying number of *flipped* labels, being monotonically increasing with the number of $\overline{y}$ wrong labels until they amount up to l. Provided that the classifier decides to discarding wrong labels (i.e., flips $\overline{y}$'s) with some confidence ϵ,

$$p\left(f(\mathbf{x}) < \gamma, y = 0\right) > \epsilon \tag{17}$$

then the error deviation is decreased. If the initial probability of success ϵ is high enough, in the limit, the classifier learnt by the iterative algorithm will make more accurate decisions and $\epsilon_{i+1} \geq \epsilon_i$, providing a lower number of errors.

4. Improving Ranking Relevance

In this section, we show how to improve search ranking relevance using the primary category classifier described in Section 3. We first describe a *learning to rank* framework for search. Then, we propose a new rank feature based on the primary categories represented as weights.

4.1. Training a Ranking Function

Given a query q, let $\mathcal{D}_q = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ be the training data of n documents where $\mathbf{x}_i \in \mathbb{R}^m$ is the feature vector and y_i is the relevance label of the i-th document. In a ranking problem, $\mathcal{D}_q$ is given as input and a permutation τ of $\{1, \ldots, n\}$ is returned as output. $\mathbf{x}_i$ is ranked higher than $\mathbf{x}_j$ if $\tau(\mathbf{x}_i) < \tau(\mathbf{x}_j)$ and this means $\mathbf{x}_i$ is more relevant to q than $\mathbf{x}_j$. Typically, a *ranking function* $f : \mathbb{R}^m \to \mathbb{R}$ is trained and applied to $\mathcal{D}_q$. A permutation or ranking τ is generated by ordering the $f(\mathbf{x}_i)$ in the descending order.

Many *learning to rank* methods have been proposed to improve ranking relevance based on machine learning [4, 6, 12, 16, 37]. Since our focus is to develop a new feature that can be used in any learning to rank methods, we use a simple GBDT method [13].

We remark that the learning procedure for our ranking function is equivalent to that for the classifier in Section 3.4 once the training data is provided. The difference is that the feature vector for the classifier is defined for a document-category (or category-document) pair while the feature vector for the ranking function is defined for a query-document pair. Queries include category names, but not vice versa. For example, category names such as “japanese restaurants" are commonly used as queries, but queries such as “best pizza" do not belong to category taxonomy. Another difference is that there are much more features (e.g., text match features, click-based features, etc) for the ranking function than the classifier. Hence, the training data for the ranking function is much larger than that for the classifier.

4.2. Matching in Category Space

In this section, we develop a method to utilize the primary category classifier to improve ranking quality. A straightforward approach may be to create a new text field for each document that contains the categories selected by the classifier and generate a set of new *ranking features* for the new text field (such as text match features), which can be used as part of training data for a ranking function. This simple approach relies on exact term match between a query and a document. Thus, when a query does not belong to category taxonomy (e.g., “best pizza"), the improved category information by our classifier does not provide much help. Also, in a ranking problem setting, it is desirable to exploit subtle differences among the predicted categories than simply treating each category equally even if those categories are predicted to be primary categories by our classifier.

To address the above problems, we propose to map queries and documents (listings) to category space and perform matching in the same category space. The mapping is done as follow.

1. Map a document (listing) d to category space:

$LCV_d = (w_1, \ldots, w_{|C|})$ where $w_i = f(d, c_i)$ if $c_i \in C_d$ and $w_i = 0$ otherwise.

2. Map a query q to category space:
   $QCV_q = \sum_{d \in S_q} r(q,d) LCV_d$ where S_q is the set of all documents that appear in the search results for q in the search logs and $r(q,d)$ is a relevance measure (e.g., CTR of d given q) for (q,d).
3. Perform matching between q and d in category space:
   $CATSIM(q,d) = \frac{QCV_q \cdot LCV_d}{||QCV_q|| \; ||LCV_d||}$

A listing-category vector (LCV) for d contains the output values of the classifier $f(d,c)$ for each $c \in C$. A query-category vector (QCV) for q is the weighted sum of LCVs of all the documents that appear in the search logs for q. The weight $r(q,d)$ is multiplied to each LCV to exclude irrelevant documents. $r(q,d)$ can be any relevance measure between q and d. In this work, we use $COEC$ as the weight. Finally, we obtain a scalar value $CATSIM(q,d)$ for q and d, which we use as a new feature to train a ranking function.

As an example, consider a query “best pizza". In the search results, some irrelevant results such as “best buy" may appear since the document contains the term “pizza" (as part of “pizza oven"). On the other hand, most pizza restaurants do not have the term “best" in the business title (it may appear in user reviews). Thus, given the query “best pizza", there is not much difference between “best buy" and “pizza hut" in terms of syntactic match. So, if we do not know if the query is related to the category “pizza" or “electronics retailers", the ranking quality for this query may be poor. How does our method solve this problem? The “best buy" stores are not likely to be clicked on by users for the query, so the weight $r(q,\text{“best buy"})$ will be close to 0 while $r(q,\text{“pizza hut"}) \gg 0$. So, $QCV_{\text{“best pizza"}}$ will be collinear to the LCVs of pizza restaurants. In this way, $QCV_{\text{“best pizza"}}$ correctly captures the true category intent of users. Consequently, we have $CATSIM(\text{“best pizza"},\text{“pizza hut"}) \gg CATSIM(\text{“best pizza"},\text{“best buy"})$.

5. Experiments

In this section, we present experimental results to validate our methods. We conduct two types of experiments: We evaluate the performance of our primary category classifier described in Section 3. Then, we evaluate the effect of the new feature described in Section 4 for ranking.

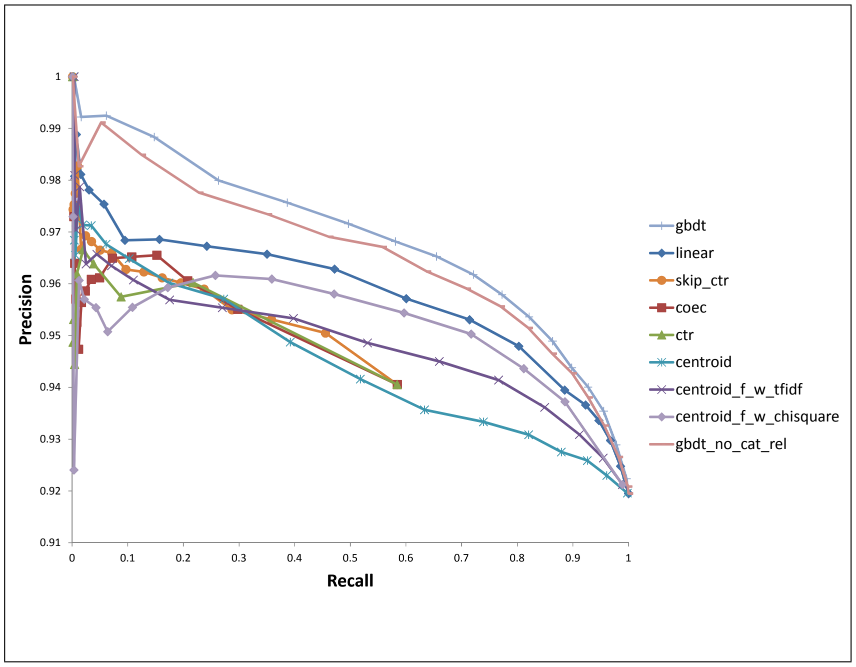


Figure 2: Precision vs. recall of different models. Each point corresponds to a threshold value for the output of a model.

5.1. Primary Category Classification

We first demonstrate that our proposed primary classifier achieves a very high precision-recall and significantly outperforms a baseline centroid-based method.

5.1.1. Data

We use data sets from a commercial local search engine. There are 20M business listings, 2K categories and 53M terms in total. That is, $||D|| =$ 20M, $||C|| =$ 2K and $||T|| =$ 53M. We obtain labels from human judges for 42K (listing, category) pairs. We generate 15 features for the 42K (listing, category) pairs in the editorial judgment data. The 15 features include 9 centroid-based similarity features and 6 click-based features. Each of 3 similarity features introduced in 3.3.1 has 3 variations: the original form, a normalized form and a hybrid form combining the two. Each of 3 click-based features in 3.3.2 has two forms: the original form and a normalized form.

Each feature has 2K dimensions (the number of categories). However, in our framework, we need to compute the feature value only for the categories given in the document. To generate click-based features in Section 3.3.2, we use 6-month click logs in the local search engine. We use 50% of the data as training data and the rest as test data. Our experiments were performed using a Hadoop-based implementation, and it took around 8 hours to compute the final models using approximately 5K nodes to process over 150GB of data.

5.1.2. Results

We evaluate the classifier trained as described in Section 3.4 using precision-recall as the evaluation metric. To see the effectiveness of our proposed method, we compare it with each of the features as a baseline:

- **gbdt**: our proposed method described in Section 3.4
- **centroid**: centroid-based similarity defined in Eq. (2)
- **centroid_f_w_tfidf**: centroid-based similarity with χ^2 filtering defined in Eq. (3)
- **centroid_f_w_chisquare**: centroid-based similarity with χ^2 filtering defined in Eq. (4)
- **ctr**: CTR defined in Eq. (5)
- **coec**: COEC defined in Eq. (6)
- **skip_ctr**: SKIP_CTR defined in Eq. (7)
- **linear**: linear regression trained on the same features used by **gbdt**
- **gbdt_no_cat_rel**: **gbdt** model trained without the category relationship feature described in Section 3.3.4

Figure 2 shows the comparison of different models based on precision-recall. Our proposed classifier **gbdt** significantly outperforms all baselines. The results show some interesting characteristics of each feature. In general, click-based features (**ctr**, **coec** and **skip_ctr**) show very high precision. However, click-based features suffer from a sudden drop in precision as recall decreases. This happens since a very high CTR, for example, is likely to

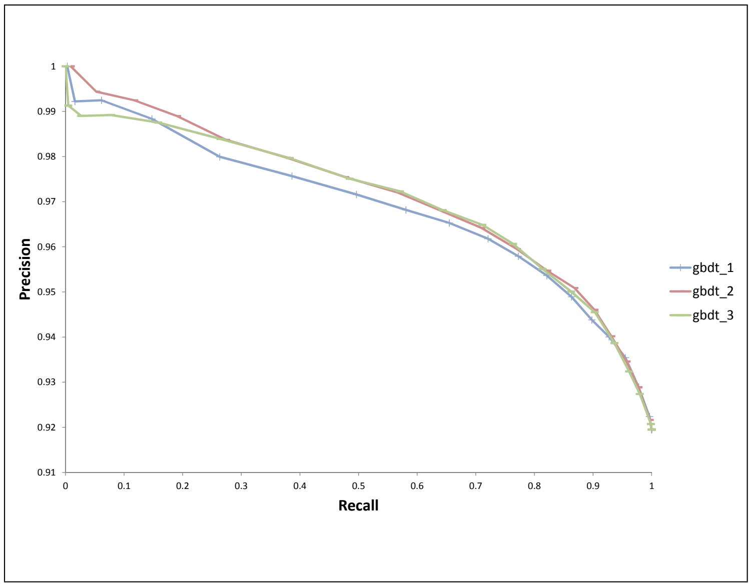


Figure 3: Effect of updating centroids with the iterative method. **gbdt_i** is the model after the i-th iteration in Algorithm 1.

be due to a very small number of views (with a similar number of clicks). Also, we can see that click-based features have limited recall compared to centroid-based features: They can never achieve recall higher than 60%. On the other hand, centroid-based features are able to achieve a significantly higher recall (over 60%).

We also observe that the χ^2-based term filtering (both **centroid_f_w_tfidf** and **centroid_f_w_chisquare**) improves prediction. Comparing **centroid_f_w_tfidf** and **centroid_f _w_chisquare**, we find that **centroid_f_w_chisquare** outperforms **centroid_f_w_tfidf** in the high recall region but suffers from a sudden drop in precision as recall decreases. This shows a problem of using χ^2 statistics scores alone as weights. The cosine similarity for **centroid_f_w_chisquare** can be increased by having only a few terms with high χ^2 statistics scores. The results show that it is important for a document to have the term distribution similar to the term distribution in a centroid.

Although each model (or feature) has the pros and cons as mentioned above, our final classifier can combine the models effectively to improve classification. The experimental results clearly show that our classifier combines the benefits of different models.

Effect of the category relationship feature: Figure 2 also shows that the category relationship feature described in Section 3.3.4 improves classification. **gbdt_no_cat_rel** does not use the category relationship feature and performs clearly worse than **gbdt** using all features.

Iterative method: Figure 3 shows the result of the iterative method described in Section 3.5. As expected, the performance improves after the first iteration. However, after the second iteration, the results do not improve. It is because the model after the first iteration $f_0(\mathbf{x})$ removes most of the irrelevant categories from C_d^0. Thus, after the second iteration, $f_1(\mathbf{x})$ affects only a small number of documents. So, $C_d^2 \approx C_d^1$.

Table 2 shows the ordered categories by **gbdt** for "Red Lobster" in Figure 1. The primary category of the business "Seafood Restaurants" is on top while a irrelevant category "Carry Out & Take Out" is on bottom.

5.2. Ranking and Filtering

In this section, we evaluate the effectiveness of the method described in Section 4 for search ranking. We are also interested in how our method affects another task, filtering for local search results included in a Web search results page (SERP). In this task, the goal is to remove irrelevant results from the section in a Web SERP showing local search results. We show that the new feature $CATSIM$ improves both ranking and filtering.

5.2.1. Data and Evaluation Metrics

We use data sets from a commercial local search engine. In the training data, there are 64K feature vectors (query-document pairs). The test data has 16K feature vectors. In each feature vector, there are 651 features including text match features, click-related features, etc. For each query-document pair, we collect editorial judgments using 3 grades (Good, Fair and Bad). The evaluation is based on three metrics DCG_1, DCG_3 and DCG_5. DCG_k is defined to be

$$\text{DCG}_k = \sum_{i=1}^{k} \frac{G_i}{\log_2(i+1)}$$

where G_i denotes the numeric value of the label of the document at position i.

Category
Seafood Restaurants
Restaurants
Steak Houses
American Restaurants
Carry Out & Take Out

Table 2: Categories sorted by the **gbdt** outputs for “Red Lobster" in Figure 1.

DCG_1 Gain	DCG_3 Gain	DCG_5 Gain
1.02%	0.99%	0.66%

Table 3: Ranking relevance gains by using the new feature $CATSIM$. Gains are statistically significant (t-test, $p\text{-value} < 0.05$).

5.2.2. Results

We compare the following two ranking functions.

- **catsim**: the ranking function trained on the existing features and the new feature $CATSIM$ described in Section 4.2
- **no_catsim**: the ranking function trained without $CAT\ SIM$

Ranking: Table 3 shows the DCG gains of **catsim** over **no_catsim**. Considering the size of the existing training data and the fact that we are adding a single feature, the DCG gains around 1% are not small improvements.

Filtering: In order to remove bad results from the local search results section in a Web SERP, we typically set a threshold for the score of the local search ranking function: If the score of the ranking function for a document falls below the threshold, the document is filtered out. To measure the performance of filtering, we treat “Good" as a positive label and “Fair" and “Bad" as a negative label. Thus, the filtering problem can be considered as a classification problem and we use precision-recall as the evaluation metric. Figure 4 shows the precision-recall comparison of the two ranking functions in which **catsim** clearly outperforms **no_catsim**.

6. Conclusions

In this paper we presented a solution to a large-scale multi-label classification problem in the context of finding primary categories of local businesses. We showed that we can combine multiple label sources effectively to train a highly accurate classifier and demonstrated that our classifier outperforms a Centroid-based method. We also proposed a new ranking feature based

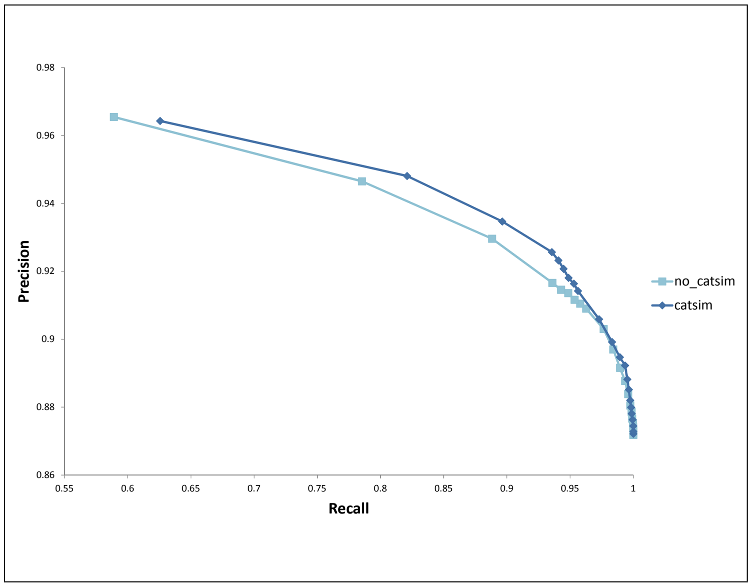


Figure 4: Effect of the new feature *CATSIM* for filtering.

on the mapping of query and document to category space and showed that the proposed rank feature clearly improves ranking and filtering for local search results. Future work will further explore the use of our taxonomy derived features in other classification and search tasks, to determine whether this document representation technique generalizes well enough in scenarios different than the one we approached in this study.

References

[1] Acir, N. 2006. A support vector machine classifier algorithm based on a perturbation method and its application to ecg beat recognition systems. *Expert Systems with Applications 31,* 1, 150 – 158.

[2] Bennett, P. N., Svore, K. M., and Dumais, S. T. 2010. Classification-enhanced ranking. In *World Wide Web Conference Series.* 111–120.

[3] Blanco, R., Matthews, M., and Mika, P. 2015. Ranking of daily deals with concept expansion. *Information Processing and Management 51,* 4, 359 – 372.

[4] Boyan, J., Freitag, D., and Joachims, T. 1996. A machine learning architecture for optimizing web search engines. In *Proceedings of the AAAI Workshop on Internet-Based Information Systems*.

[5] Broder, A., Fontoura, M., Josifovski, V., and Riedel, L. 2007. A semantic approach to contextual advertising. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '07. ACM, New York, NY, USA, 559–566.

[6] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*. 89–96.

[7] Cao, H., Hu, D. H., Shen, D., Jiang, D., Sun, J.-T., Chen, E., and Yang, Q. 2009. Context-aware query classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '09. ACM, New York, NY, USA, 3–10.

[8] Chapelle, O. and Zhang, Y. 2009. A dynamic bayesian network click model for web search ranking. In *WWW '09: Proceedings of the 18th international conference on World wide web*. ACM, New York, NY, USA, 1–10.

[9] Chen, R.-C. and Hsieh, C.-H. 2006. Web page classification based on a support vector machine using a weighted vote schema. *Expert Systems with Applications 31,* 2, 427 – 435.

[10] CissÃľ, M., ArtiÃĺres, T., and Gallinari, P. 2012. Learning compact class codes for fast inference in large multi class classification. In *ECML/PKDD (1)*, P. A. Flach, T. D. Bie, and N. Cristianini, Eds. Lecture Notes in Computer Science Series, vol. 7523. Springer, 506–520.

[11] Dumais, S., Cutrell, E., and Chen, H. 2001. Optimizing search by showing results in context. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '01. ACM, New York, NY, USA, 277–284.

[12] Freund, Y., Iyer, R., Schapire, R., and Singer, Y. 1998. An efficient boosting algorithm for combining preferences. In *Proceedings of the Fifteenth International Conference on Machine Learning*.

[13] Friedman, J. 2001. Greedy function approximation: a gradient boosting machine. *Ann. Statist. 29*, 1189–1232.

[14] Han, E.-H. and Karypis, G. 2000. Centroid-based document classification: Analysis and experimental results. In *Principles of Data Mining and Knowledge Discovery*, D. Zighed, J. Komorowski, and J. Zytkow, Eds. Lecture Notes in Computer Science Series, vol. 1910. Springer Berlin / Heidelberg, 116–123.

[15] Joachims, T. 2001. A statistical learning learning model of text classification for support vector machines. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '01. ACM, New York, NY, USA, 128–136.

[16] Joachims, T. 2002. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD*. ACM Press, New York, NY, USA, 133–142.

[17] Kang, C., Wang, X., Chang, Y., and Tseng, B. 2012. Learning to rank with multi-aspect relevance for vertical search. In *Proceedings of the fifth ACM international conference on Web search and data mining*. WSDM '12. ACM, New York, NY, USA, 453–462.

[18] Kim, S.-B., Rim, H.-C., Yook, D., and Lim, H. 2002. Effective methods for improving naive bayes text classifiers. In *Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence*. PRICAI '02. Springer-Verlag, London, UK, UK, 414–423.

[19] Kim, S.-M., Pantel, P., Duan, L., and Gaffney, S. 2009. Improving web page classification by label-propagation over click graphs. In *Proceedings of the 18th ACM conference on Information and knowledge management*. CIKM '09. ACM, New York, NY, USA, 1077–1086.

[20] Lertnattee, V. and Theeramunkong, T. 2004. Effect of term distributions on centroid-based text categorization. *Information Sciences 158,* 0, 89 – 115.

[21] Li, X., Wang, Y.-Y., and Acero, A. 2008. Learning query intent from regularized click graphs. In *Proceedings of the 31st annual interna-*

tional ACM SIGIR conference on Research and development in information retrieval. SIGIR '08. ACM, New York, NY, USA, 339–346.

[22] Li, Y., Zheng, Z., and Dai, H. K. 2005. Kdd cup-2005 report: facing a great challenge. *SIGKDD Explor. Newsl. 7,* 2, 91–99.

[23] Liao, Y. and Vemuri, V. 2002. Use of k-nearest neighbor classifier for intrusion detection. *Computers & Security 21,* 5, 439–448.

[24] Manning, C. D., Raghavan, P., and Schutze, H. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

[25] Min, S.-H., Lee, J., and Han, I. 2006. Hybrid genetic algorithms and support vector machines for bankruptcy prediction. *Expert Systems with Applications 31,* 3, 652 – 660.

[26] Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Comput. Surv. 34,* 1, 1–47.

[27] Tan, S. 2006. An effective refinement strategy for knn text classifier. *Expert Syst. Appl. 30,* 2, 290–298.

[28] Tan, S. 2008. An improved centroid classifier for text categorization. *Expert Syst. Appl. 35,* 1-2, 279–285.

[29] Trappey, A. J., Hsu, F.-C., Trappey, C. V., and Lin, C.-I. 2006. Development of a patent document classification and search platform using a back-propagation network. *Expert Systems with Applications 31,* 4, 755 – 765. Computer Supported Cooperative Work in Design and Manufacturing.

[30] Vapnik, V., Levin, E., and Lecun, Y. 1994. Measuring the VC-Dimension of a Learning Machine. *Neural Computation 6*, 851–876.

[31] Venetis, P., Gonzalez, H., Jensen, C. S., and Halevy, A. Y. 2011. Hyper-local, directions-based ranking of places. *PVLDB 4,* 5, 290–301.

[32] Wang, D., Wu, J., Zhang, H., Xu, K., and Lin, M. 2013. Towards enhancing centroid classifier for text classification - a border-instance approach. *Neurocomputing 101*, 299–308.

[33] Wang, L.-M., Li, X.-L., Cao, C.-H., and Yuan, S.-M. 2006. Combining decision tree and naive bayes for classification. *Know.-Based Syst. 19,* 7, 511–515.

[34] Weston, J., Makadia, A., and Yee, H. 2013. Label partitioning for sublinear ranking. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, S. Dasgupta and D. Mcallester, Eds. Vol. 28. JMLR Workshop and Conference Proceedings, 181–189.

[35] Wu, M.-C., Lin, S.-Y., and Lin, C.-H. 2006. An effective application of decision tree to stock trading. *Expert Syst. Appl.*, 270–274.

[36] Yang, Y. and Pedersen, J. O. 1997. A comparative study on feature selection in text categorization. Morgan Kaufmann Publishers, 412–420.

[37] Zheng, Z., Chen, K., Sun, G., and Zha, H. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th ACM SIGIR conference.*