Using Web Searches on Important Words to Create Background Sets for LSI Classification

Sarah Zelikovitz and **Marina Kogan**[*]
College of Staten Island of CUNY
2800 Victory Blvd
Staten Island, NY 11314

Abstract

The world wide web has a wealth of information that is related to almost any text classification task. This paper presents a method for mining the web to improve text classification, by creating a background text set. Our algorithm uses the information gain criterion to create lists of important words for each class of a text categorization problem. It then searches the web on various combinations of these words to produce a set of related data. We use this set of background text with Latent Semantic Indexing classification to create an expanded term by document matrix on which singular value decomposition is done. We provide empirical results that this approach improves accuracy on unseen test examples in different domains.

Introduction

Text Classification and Unsupervised Learning

Categorizing textual data has many practical applications, including email routing, news filtering, topic spotting, and classification of technical documents. Traditional machine learning programs use a training corpus of hand-labeled training data to classify new unlabeled test examples. Often the training sets are extremely small, due to limited availability of data or to the difficult and tedious nature of labeling, and classification decisions can therefore be difficult to make with high confidence.

Recently, there have been many researchers that have looked at combining supervised text learning algorithms with unsupervised learning (Nigam *et al.* 2000; Joachims 2003; Belkin & Niyogi 2004). By augmenting the training set with additional knowledge, it has been shown that accuracy on test sets can be improved using a variety of learning algorithms including Naive Bayes (Nigam *et al.* 2000), support vector machines (Joachims 1999; 2003), and nearest-neighbor algorithms (Zelikovitz & Hirsh 2002). This additional knowledge is generally in the form of unlabeled examples, test corpora that are available, or related background knowledge that is culled from other sources.

When unlabeled examples, or test examples, are incorporated into the text classification process one is assured that these added examples are relevant to the task domain. These examples can either be classified and then used for further help in creation of a model of the domain (Nigam *et al.* 2000), or placed in the space of documents and evaluated according to some criteria of the learner that is used (Joachims 1999; Zelikovitz & Marquez 2005). Given the huge proliferation of articles, Web sites, and other source of information that often exist, it is important for text classifiers to take advantage of these additional resources in much the same way as unlabeled examples are used. This information can be looked at as background knowledge that can aid in the classification task. Zelikovitz and Hirsh consider a broad range of background text for use in classification, where the background text is hopefully relevant to the text classification domain, but doesn't necessarily take the same general form of the training data. For example, a classification task given labeled Web-page titles might have access to large amounts of Web-page contents. Rather than viewing these as items to be classified or otherwise manipulated as if they were unlabeled examples, the pieces of background knowledge are used to provide information about words in the task domain, including frequencies and co-occurrences of words.

Using Background Knowledge

One method of incorporating background knowledge in a nearest neighbor paradigm, uses a latent semantic indexing (Deerwester *et al.* 1990) (LSI) approach (Zelikovitz & Hirsh 2001). LSI creates a matrix of documents, and uses singular value decomposition to reduce this space to one that hopefully reflects the relationships between words in the textual domain. The addition of background knowledge into this matrix allows for the decomposition to reflect relationships of words in the background knowledge as well. However, in order for the additional knowledge to be useful for classification it must be related to the text categorization task and to the training data.

Outline

In the next section we describe LSI and illustrate how we use it for nearest neighbor text classification in conjunction with background knowledge. We then describe our method for obtaining background knowledge from the world-wide

[*] Supported by the PSC-CUNY Research Award Program

web. Finally we present results on a few data sets to show that this method can enhance classification.

Latent Semantic Indexing

Latent Semantic Indexing (Deerwester *et al.* 1990) is based upon the assumption that there is an underlying semantic structure in textual data, and that the relationship between terms and documents can be redescribed in this semantic structure form. Textual documents are represented as vectors in a vector space. Each position in a vector represents a term (typically a word), with the value of a position *i* equal to 0 if the term does not appear in the document, and having a positive value otherwise. Based upon previous research (Dumais 1993) we represent the positive values as a local weight of the term in this document multiplied by a global weight of the term in the entire corpus. The local weight of a term *t* in a document *d* is based upon the log of the total frequency of *t* in *d*. The global weight of a term is the entropy of that term in the corpus, and is therefore based upon the number of occurrences of this term in each document. The entropy equals $1 - \sum_d \frac{p_{td} log(p_{td})}{log(n)}$ where n is the number of documents and p_{td} equals the number of times that t occurs in d divided by the number of total number of times that t occurs. This formula gives higher weights to distinctive terms. Once the weight of each term in every document is computed we can look at the corpus as a large term-by-document $(t \times d)$ matrix X, with each position x_{ij} corresponding to the absence or weighted presence of a term (a row i) in a document (a column j). This matrix is typically very sparse, as most documents contain only a small percentage of the total number of terms seen in the full collection of documents.

Unfortunately, in this very large space, many documents that are related to each other semantically might not share any words and thus appear very distant, and occasionally documents that are not related to each other might share common words and thus appear to be closer than they actually are. This is due to the nature of text, where the same concept can be represented by many different words, and words can have ambiguous meanings. LSI reduces this large space to one that hopefully captures the true relationships between documents. To do this, LSI uses the singular value decomposition of the term by document $(t \times d)$ matrix.

The singular value decomposition (SVD) of the $t \times d$ matrix, X, is the product of three matrices: TSD^T, where T and D are the matrices of the left and right singular vectors and S is the diagonal matrix of singular values. The diagonal elements of S are ordered by magnitude, and therefore these matrices can be simplified by setting the smallest k values in S to zero.[1] The columns of T and D that correspond to the values of S that were set to zero are deleted. The new product of these simplified three matrices is a matrix $\hat{X}$ that is an approximation of the term-by-document matrix. This new matrix represents the original relationships as a set of orthogonal factors. We can think of these factors as combining meanings of different terms and documents; documents are then re-expressed using these factors.

Expanding the LSI Matrix

Intuitively, the more training examples available, the better the SVD will be at representing the domain. What is most interesting to us about the singular value decomposition transformation is that it does not deal with the classes of the training examples at all. This gives us an extremely flexible learner, for which the addition of other available data is quite easy. Since LSI is an unsupervised learner and it simply creates a model of the domain based upon the data that it is given, there are a number of alternative methods that we could use to enhance its power. Instead of simply creating the term-by-document matrix from the training examples alone, we can combine the training examples with other sources of knowledge to create a much larger term-by-"document" matrix, X_n. If the text categorization problem consists of classifying short text strings, this additional data would be especially useful. The additional data can contain words that are very domain related but do not appear in the training set at all. These words might be necessary for the categorization of new test examples.

Classification of Test Examples

A new test example that is to be classified can be reexpressed in the same smaller space that the training examples (or training examples and background knowledge) has been expressed. This is done by multiplying the transpose of the term vector of the test example with matrices T and S^{-1}. Using the cosine similarity measure, LSI returns the nearest training neighbors of a test example in the new space, even if the test example does not share any of the raw terms with those nearest neighbors. We can look at the result of the LSI comparison as a table containing the tuples

$$\langle \textit{train-example}, \textit{train-class}, \textit{cosine-distance} \rangle$$

with one line in the table per document in the training collection. There are many lines in the table with the same *train-class* value that must be combined to arrive at one score for each class. We use the noisy-or operation to combine the similarity values that are returned by LSI to arrive at one single value per class. If the cosine values for documents of a given class are $\{s_1, \ldots, s_n\}$, the final score for that class is $1 - \prod_{i=1}^{n}(1 - s_i)$. Whichever class has the highest score is returned as the answer to the classification task. Based upon (Yang & Chute 1994; Cohen & Hirsh 1998) only the thirty closest neighbors are kept and combined.

It has been shown in prior work (Zelikovitz & Hirsh 2001) that the incorporation of background knowledge into the LSI process can aid classification. When training examples are short, or there are few training examples, the reduced space created by the SVD process more accurately models the domain, and hence a greater percentage of test examples can be classified correctly. The challenge is in obtaining a background set that is related to the classification task in an inexpensive manner.

[1] The choice of the parameter k can be very important. Previous work has shown that a small number of factors (100-300) often achieves effective results.

Automatic Creation of Background Sets

Suppose we wish to classify the *titles* of web pages for a veterinary site as belonging to specific animals (cat, dog, horse, etc) as in www.netvet.wustl.edu, to facilitate the organization of a web site that will be helpful to people interested in these individual topics. The text categorization task can be defined by looking at the titles as the individual examples, and the classes as the list of animals that could be assigned to any specific title. Some training/test examples can be seen in Table 1. If many web-page titles have already been classified manually, machine learning algorithms can be used to classify new titles. If only a small number of web-page titles are known to fit into specific categories, these algorithms may not be useful enough. However, it is clear that the WWW contains many pages that discuss each of these animals. These pages can be downloaded and organized into a corpus of background text that can be used in the text classification task.

Searching Procedures

For each of our categorization tasks we automatically created a background corpus consisting of related documents from the World Wide Web. Our application to do this, written in Java, created queries, as described in the next few sections, that were input to the Google search engine. Google provides an API, which can be used to query its database from Java, thus eliminating the need to parse sophisticated set of parameters to be passed to it through the API. We restricted our queries to only retrieve documents written in the English language and we restricted the document type to be of *html* or *htm*. This avoided the return of .pdf files, .jpeg files, as well as other non-text files that are on the WWW. Once the Google API returned the results of the search, our application then started a thread to download the each individual page from the URLs. The thread was given a maximum of 5 seconds to download and retrieve the document before timing out. We removed all pages whose domain names matched the source of the data set, and we saved the textual sections of the remaining documents that were downloaded, and each one became a background piece of text.

Creating Queries

In order to form queries that can be input to Google, we found the *important words* in the corpus, and used those words as the basic search terms that build the queries. The pages matched by Google are hopefully related to the text classification task, and are therefore used as the background set. To find the *important words* in the corpus, we began by using the information gain criterion to rank all words in the training set; no stemming was used to facilitate query creation later. For a supervised text classification task, each word that is present in the training corpus can be seen as a feature that can be used for classification. Given the training set of classified examples, T, we partition it by the presence or absence of each word. Each word gives a partition of the training set, $P = \{T_0, T_1\}$ of T. The information gain for this word is defined as $entropy(T) - (entropy(T_0) \times \frac{|T_0|}{|T|} + entropy(T_1) \times \frac{|T_1|}{|T|})$. Words with high information gain create partitions of the original training set that overall have a lower entropy, and hence are reflective of the particular classification scheme. A list of the words with the highest information gain for the veterinary problem can be seen in 2. Once we obtained the information gain value for all words, we sorted all words in the corpus in descending order based upon this value. To create queries from this list of ranked words, we wished to combine those words that best reflected each individual class. To do this, we created a list of n words [2] that best reflected each class. We labeled each of the words with the class whose training examples most reflected this word, i.e. whose training examples contained that actual word. We then chose the top words for each of the classes. An example of the ten top words for some of the classes in the veterinary set can be seen in Table 3. Many of the training examples did not contain the important words that were associated with their class. For example, in the veterinary domain, only 10% of the training examples from class *cat* contained the important words that best reflected that class. For the veterinary task, this number ranged from 8% to 34%. Although this number looks low, these words are used for querying the world wide web, and if the words properly reflect the class, these queries can return pages that contain words that are in the many other training examples as well.

Table 1: Veterinary Set

Training/Test Example	Class
Montana Natural Heritage Program	Wildlife
Visual Rhodesian Ridgeback	Dog
Dog Lovers Pedigree Service	Dog
Finch and Canary World Magazine	Bird

Table 2: List of Words with Information Gain

Word	Information Gain
Horse	0.0774
Wildlife	0.0750
Cat	0.0624
Dog	0.0613
Bird	0.0558
Fishing	0.0259
Cattle	0.0249
Aquarium	0.0247
Laborator	0.0246

At this point we had a list of ten words per class that was both high information gain, and reflected a particular class from which to create our queries. This is important, because we wanted the pages that were returned to be domain related, but to fit into one (or possibly a few) classes, so that co-occurrences can be used by the SVD process to create features that help for the specific classification of the problem.

[2] This is a user defined parameter, and we actually ranged from 3 to 10 in our studies.

Table 3: Important Words Per Class

10 words	Class
Bird Birds Birding Society Poultry Audubon Raptor Wild Aviary Parrot	Bird
Dog Pet Dogs Club Rescue Kennels Wolf Canine Retriever German	Dog
Wildlife Museum Natural Conservation Nature History Species Zoological Exotic National	Wildlife
Cat Cattery Cats Feline Maine Pets Coon Care Kitty Litter	Cat

Practically speaking, Google's API has a ten page limit on returns for any individual query. We needed many more than ten pages in order to augment the term by document matrix for LSI classification, since background text that is domain related is important for the SVD process to reflect relationships between words. Also, as with all search engines, giving queries that consisted of ten words to Google was too restrictive. Since there are very few pages (perhaps none!) that contained all ten of the top words for many classes, not many pages were returned. We therefore used the ten words to create numerous different queries, each of which was input to the Google API. We chose all 3 word combinations from the 10 word sequences, and used each of these as an individual query for which we received ten pages through the Google API to be added to the background text. Some examples of these queries can be seen in Table 4. Due to variation in the search (not all pages returned were text and not all queries returned the full ten pages), and removal of related pages, for our runs, the number of pages used to create the background text for the data sets that are described below ranged from 16,000 to 24,000.

Table 4: Sample Queries

3 Word Query	Class
Bird Society Poultry	Birds
Bird Society Birding	Birds
Bird Society Audubon	Birds
Bird Society Ostrich	Birds
Bird Society Raptor	Birds
Pet Dogs Kennels	Dogs
Pet Dogs Canine	Dogs
Fish Aquarium Fisheries	Fish
Fish Aquarium Shark	Fish
Fish Aquarium Reef	Fish
Fish Aquarium Flyfishing	Fish

An example of a top page that is returned as a result of one of the queries for the class *bird* in the veterinary data set is from http://www.ummz.lsa.umich.edu/birds/wos.html. The first part of the text of the page follows:

```
The Wilson Society, founded in 1888, is a
 world-wide organization of nearly 2500
people who share a curiosity about birds.
Named in honor of Alexander Wilson, the
Father of American Ornithology, the
Society publishes a quarterly journal of
ornithology, The Wilson Bulletin, and
holds annual meetings.

Perhaps more than any other biological
science, ornithology has been advanced by
the contributions of persons in other
chosen professions. The Wilson Society
recognizes the unique role of the serious
amateur in ornithology. Fundamental to
its mission, the Society has
```

The word *ornithology* does not appear in list of ten most important words for the *bird* class, and therefore is not one of the search terms for this run. However, the home page for the "Wilson Ornithological Society" is returned as a top match for some of search terms, and allows LSI to find relationships about many other important words in the domain.

It still might be the case that many of these pages do not clearly fit into any one of the specific categories. For example, pages might discuss pets, which would be relevant to both cats and dogs, but not perhaps to primates and frogs. Still, these pages are in the proper domain and have relevance to the topic and can help learners in terms of word frequencies and co-occurrences. An example of the text of one of the pages returned from a query that was created for the class *wildlife* follows:

```
Exotics and Wildlife

American Zoo & Aquarium Association
Representing virtually every
professionally operated zoological park,
aquarium, oceanarium, and wildlife park
in North America, the AZA web page
 provides a multitude of resources.

California Department of Fish and Game
Along with loads of information, you can
also view current job opportunities
(including seasonal employment)
```

This page includes the word *fish*, as well as the word *aquarium*, which were actually words that were used for querying for the class *fish*. This page, therefore, does not actually fit clearly into any one of the specific classes, although it is clearly not about farm animals or household pets, and so hence does reflect some properties of the classification. Therefore, it could still be useful in the SVD process.

Data Sets for Evaluation

As test beds for exploring the evaluation of background knowledge, we used two training/test sets, variation of which have been used in previous machine learning work

Table 5: Accuracy Rates

Data Set	Without Background	With Background
Physics Paper Titles/ Full Training Set	89.2%	94.0%
Physics Paper Titles/ Small Training Set	79.7%	93.9%
Veterinary/ Full Training Set	45.4%	69.4%
Veterinary/ Small Training Set	40.5%	68.0%

(Cohen & Hirsh 1998; Zelikovitz & Marquez 2005). These data sets are described below. Results that are reported are five-fold cross validated, and for each of the cross validated runs, a training test was formed, queries were created and sent to Google, and a background set was created.

Technical papers One common text categorization task is assigning discipline or sub-discipline names to technical papers. We created a data-set from the physics papers archive (http://xxx.lanl.gov), (Zelikovitz & Hirsh 2001; Zelikovitz & Marquez 2005) where we downloaded the titles for all technical papers in the first two areas in physics for one month (March 1999). In total there were 1066 examples in the training-test set combined.

Veterinary Titles As described in the paragraphs above, the titles of web pages from the site http://www.netvet.wustle.edu were used to create a 14 class problem. Each title is classified by the type of animal that it is about. There were a total of 4214 examples in the training and test set combined.

Results are shown in Table 5. Each line of the table shows the accuracy rate on unseen test examples both with and without the automatically selected background knowledge. The line for the full data set is the average across the five-fold cross validated trials. We then took each cross-validated trial and trained on only one fifth of the training set, keeping the test set steady. Results are in the line labeled as a small set. For each cross-validated trial query creation was done on the training set, as was singular value decomposition. As can be seen from the table, the background sets were extremely useful in aiding the classification task. This is partially the case in these two data sets, because many of the training and test examples contain one or two words that are very indicative of the class that they are in.

It is interesting to see that the inclusion of background knowledge compensates for the lack of training examples. To show this, we graphed the accuracy rates with and without background text for the physics paper problem. The x axis represents the percentage of the training set that was used for background text creation and for classification. The y axis represents accuracy. For the sets without background knowledge, smaller sets had lower accuracy, as expected. For the sets with background knowledge, the line is basically flat. Accuracy did not degrade much with fewer training examples, as the background set was able to enrich the vocabulary for LSI.

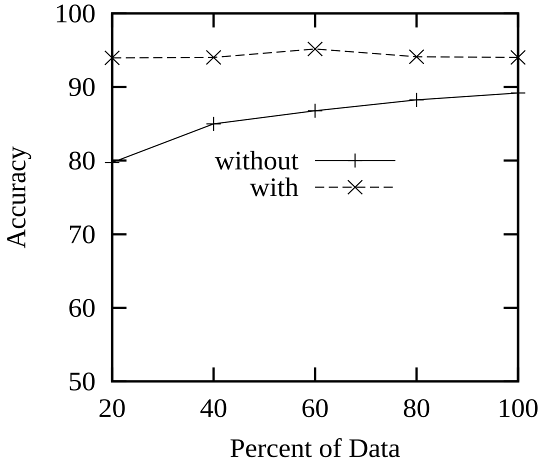

Figure 1: LSI with and without the background set for the physics problem

Conclusion and Research Questions

In summary, we have presented a method for automatically querying the web to create a set of background text. Our method uses information gain to rank words and combine them into queries to be submitted to Google. The returned pages are then added to the Latent Semantic Indexing process to improve classification.

However, there are a number of issues that we wish to explore further in the creation of background text. We wish to study which pages were actually helpful in terms of classification. To do this, we must classify using only portions of the background text, and compare resulting accuracies. We are currently studying how much background text is actually needed, and on which types of data sets this approach works best.

References

Belkin, M., and Niyogi, P. 2004. Semi-supervised learning on manifolds. *Machine Learning Journal: Special Issue on Clustering* 209–239.

Cohen, W., and Hirsh, H. 1998. Joins that generalize: Text categorization using WHIRL. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 169–173.

Deerwester, S.; Dumais, S.; Furnas, G.; and Landauer, T.

1990. Indexing by latent semantic analysis. *Journal for the American Society for Information Science* 41(6):391–407.

Dumais, S. 1993. LSI meets TREC: A status report. In Hartman, D., ed., *The first Text REtrieval Conference: NIST special publication 500-215*, 105–116.

Joachims, T. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 200–209.

Joachims, T. 2003. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 290–297.

Nigam, K.; Mccallum, A. K.; Thrun, S.; and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning* 39(2/3):103–134.

Yang, Y., and Chute, C. 1994. An example-based mapping method for text classification and retrieval. *ACM Transactions on Information Systems* 12(3):252–295.

Zelikovitz, S., and Hirsh, H. 2001. Using LSI for text classification in the presence of background text. In *Proceedings of the Tenth Conference for Information and Knowledge Management*, 113–118.

Zelikovitz, S., and Hirsh, H. 2002. Integrating background knowledge into nearest-Neighbor text classification. In *Advances in Case-Based Reasoning, ECCBR Proceedings*, 1–5.

Zelikovitz, S., and Marquez, F. 2005. Transductive learning for short-text classification problems using latent semantic indexing. *International Journal of Pattern Recognition and Artificial Intelligence* 19(2):143–163.