Transductive LSI for Short Text Classification Problems

Sarah Zelikovitz
The College of Staten Island of CUNY
2800 Victory Blvd
Staten Island, NY 10314
zelikovitz@mail.csi.cuny.edu

Abstract

This paper presents work that uses Transductive Latent Semantic Indexing (LSI) for text classification. In addition to relying on labeled training data, we improve classification accuracy by incorporating the set of test examples in the classification process. Rather than performing LSI's singular value decomposition (SVD) process solely on the training data, we instead use an expanded term-by-document matrix that includes both the labeled data as well as any available test examples. We report the performance of LSI on data sets both with and without the inclusion of the test examples, and we show that tailoring the SVD process to the test examples can be even more useful than adding additional training data. The test set can be a useful tool to combat the possible inclusion of unrelated data in the original corpus.

Introduction

Text Classification and Unsupervised Learning

The task of classifying textual data is both difficult and intensively studied (Joachims 2002; Sebastiani 1999; Nigam *et al.* 2000). Traditional machine learning programs use a training corpus of hand-labeled data to classify new unlabeled test examples. Often the training sets are extremely small, due 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 1999; Bennet & Demiriz 1998; Joachims 2003; Belkin & Niyogi). 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; Bennet & Demiriz 1998), 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.

A number of researchers have explored the use of large corpora of unlabeled data to augment smaller amounts of labeled data for classification (such as augmenting a collection of labeled Web-page titles with large amounts of unlabeled Web-page titles obtained directly from the Web). Nigam et al. (Nigam *et al.* 2000) use such background examples by first labeling them using a classifier formed on the original labeled data, adding them to the training data to learn a new classifier on the resulting expanded data, and then repeating anew the labeling of the originally unlabeled data. This approach yielded classification results that exceed those obtained without the extra unlabeled data.

Zelikovitz and Hirsh (Zelikovitz & Hirsh 2000) consider a broader 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 in nearest neighbor algorithms to find training examples that are similar to a test example that could not be found by a simple comparison between the test example and training set. Zelikovitz and Hirsh (Zelikovitz & Hirsh 2000; 2001) show that their approach is especially useful in cases with small amounts of training data and when each item in the data has few words.

A third example of background knowledge concerns cases where the data to which the learned classifier will be applied is available at the start of learning. For such learning problems, called *transductive learning* (Joachims 1999; 2003), these unlabeled examples may also prove helpful in improving the results of learning. For example, in transductive support vector machines (Joachims 1999; Bennet & Demiriz 1998) the hyperplane that is chosen by the learner is based on both the labeled training data and the unlabeled test data. Joachims shows that this is a method for incorporating priors in the text classification process and performs well on such tasks.

Transductive Learning

A transductive learner, as defined by Vapnik (Vapnik 1998) as opposed to an inductive one, makes use of the test examples in the learning process. The learner tailors itself to the actual set of test examples that it will need to classify. In this paper, we present a transductive Latent Semantic Indexing

algorithm. LSI uses singular value decomposition to reexpress examples in a smaller, simpler space that hopefully reflects some semantic meanings of the space. As opposed to using the training examples for the construction of the new space, and 'folding' (Deerwester *et al.* 1990; Dumais 1995; Berry, Dumais, & O'Brien 1995) the test examples into the new space, or updating the singular value decomposition, (Berry, Dumais, & Letsche 1995) we actually *use* the test examples during the singular value decomposition. In this way the reexpression of the space of training and test examples more closely reflects the set to be tested.

In the next section we describe LSI, and show how we use it for nearest neighbor text classification. We then present our method for incorporating test examples with a set of results on varied tasks as well as a discussion of why this method is particularly useful for this type of learner and these types of tasks.

Latent Semantic Indexing

Latent Semantic Indexing (Dumais 1995; 1993; 1996; 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.

LSI for Retrieval

When LSI is used for retrieval, a query is represented in the same new small space in which the document collection is represented in. This is done by multiplying the transpose of the term vector of the query with matrices T and S^{-1}. Once the query is represented this way, the distance between the query and documents can be computed using the cosine metric, which represents a numerical similarity measurement between documents. LSI returns the distance between the query and all documents in the collection. Those documents that have higher cosine distance value than some cutoff point can be returned as relevant to the query.

LSI for Classification

We are using LSI for text *classification*, so we can henceforth refer to the document collection as the training examples and the query as a test example. Using the cosine similarity measure, LSI returns the nearest 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 query 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. This method of nearest neighbor in conjunction with LSI we term LSI-nn.

Incorporating the Test Examples

What is most interesting to us about the singular value decomposition transformation is that it does not deal with the

[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.

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. One such method would be to include the test examples in the creation of the reduced space. If the original set of training documents is expanded to include both the training set and the test set, and SVD is run on this expanded matrix, hopefully semantic associations that are relevant to classifying the test instances would be found that could not be found if the SVD was run on the training set alone.

This type of learning is related to what is termed by Vapnik (Vapnik 1998) as "transductive learning." Instead of simply using the training data, transduction makes use of the test examples in choosing the hypothesis of the learner. In the case of the nearest neighbor algorithm that we presented above, we do not really find a *hypothesis* for the learner. However, the recreation of the space with the incorporation of the test examples does choose a *representation* based upon the test examples. The reduction of dimensionality of the training/test set combined allows the smaller space to more accurately reflect the test set to which it will be applied for classification. The inclusion of the test examples into the original matrix allows LSI to calculate entropy weights of words with the vocabulary and examples and co-occurrences of words in the test set.

In particular, when using small samples of data, the inclusion of the test set will outperform classification done by 'folding in' the test set. In general, this is because LSI, as an unsupervised learning algorithm will be helped by the inclusion of more data. For small samples of data, we can look at the test set as simply a way of obtaining pieces of unlabeled data that are guaranteed to be relevant to the task. With short text classification tasks often there are words in a test set that have not occurred in the training set at all.

To illustrate this, let us look at a sample piece of data from the Electronic Zoo (www.netvet.wustl.edu), where the text classification task is to place a new Web page title under the correct animal category. The test example:

Equine Protozoal Myeloencephalitis (Clara K. Fenger, DVM, PhD, DACVIM)

which is the title of a Web page discussing diseases of horses, is misclassified when the test set is not included in the original singular value decomposition. To see why, two of the nearest neighbors from the training set are:

Bayer Equine WellCare Program, horse
Bayer Advantage Flea Adulticide Page, cat

The word *Bayer* which occurs with the word *Equine* in the first neighbor causes the second neighbor to be returned as well. Obviously, these two training examples are close to each other in the reduced space. When all thirty nearest neighbors are combined, the final result that is returned is *cat*, which is the incorrect class. When the test examples are included, however, this second training example is not returned. Instead all the nearest neighbors belong to the correct category.

Empirical Results

Data Sets

We ran this program on five different data sets that we (and others) have used for text classification. A table showing the name of the data set, url, number of examples, total number of classes, and average length of examples can be seen in Table 1.

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, where we downloaded the titles for all technical papers in the first three areas in physics (astrophysics, condensed matter, and general relativity and quantum cosmology) for the month of March 1999 (Zelikovitz & Hirsh 2000).

Web page titles We have taken two data sets from previous work on text classification (Cohen & Hirsh 1998; Zelikovitz & Hirsh 2000). The first, NetVet, included the Web page headings for its pages concerning cows, horses, cats, dogs, rodents, birds and primates. For example, a training example in the class birds might have been: "Wild Bird Center of Walnut Creek".

Companies The second of these data sets consisted of a training set of company names, 2472 in all, taken from the Hoover Web site (http://www.hoovers.com) labeled with one of 124 industry names.

News Another data set that we created was obtained from Clarinet news. We downloaded all articles under the sports and banking categories for one month, using the most recent for training and test sets. The training/test set consisted of the first 9 words of each article.

Thesaurus Roget's thesaurus places all words in the English language into one of six major categories: space, matter, abstract relations, intellect, volition, and affection. From http://www.thesaurus.com, we created a labeled training/test set of 1000 words, sometimes including a description of the word if the web site contained a description for disambiguation, with each word associated with one category.

Results

We ran LSI without the test examples, which we term LSI-nn, and LSI incorporating the test examples, which we term LSI-test on all the data sets, using the full number of training examples as well as subsets of the training examples. Each result reported for the Physics titles, NetVet, Business, and Thesaurus data sets represents an average of five cross-validated runs. For each cross-validated run, four-fifths of the data was used as the training set and one-fifth was used as the test set. Holding each test set steady, the number of examples in the training sets were varied. Each of the five data sets was tested with LSI-nn and LSI-test using 20, 40, 60, 80, and 100 percent of the training data.

Table 1: Details of the Data Sets

Data Set	URL	Size	Number of Classes	Avg Length in Words
Physics	xxx.lanl.gov	1066	2	12
NetVet	www.netvet.wustle.edu	1789	7	5
Business	www.hoovers.com	2472	124	4
News	www.clarinet.com	1033	2	9
Thesaurus	wwww.thesaurus.com	1000	6	1

In Figures 1– 5 we present the accuracy rates for LSI-nn and LSI-test on five data sets. As we can see from the numbers presented, LSI-test always outperforms LSI-nn, although often by a small amount.

An interesting observation that we can make from these graphs is that supplementing the test examples is more useful than simply adding additional labeled examples. In the physics data set and the business data set, for example, incorporating the test set with 40% of the data performed better than 100% of the data without the test set. Since all these results were 5-fold cross validated, the number of examples in the test set is equal to 20% of the full set of data, yet the accuracy of LSI-test at a given point is higher than LSI-nn, at points to the right of it in the graph.

A major drawback of this method is that it assumes that the entire corpus of test examples is available when SVD is done, which is not always the case. In the examples that we present, for instance, the test sets are quite small compared to a possible set of unlabeled examples, which may be much larger. The gains that could be made in accuracy by adding the test set into the singular value decomposition process is therefore limited, because the number of test examples that are available is limited. However, it is still the case that we can look at the test set as an external corpus of background knowledge and use it alone, or in conjunction with other sources of knowledge to create the new semantic space. The test examples can be particularly useful in the LSI approach, because since the test examples are most closely related to the training set, we do not have to worry about modeling an unrelated concept that appears in the background knowledge, but that does not appear in the training and test set. The incorporation of the test set also allows the SVD process to place emphasis on terms that will be useful in the classification process, since they appear in the test set.

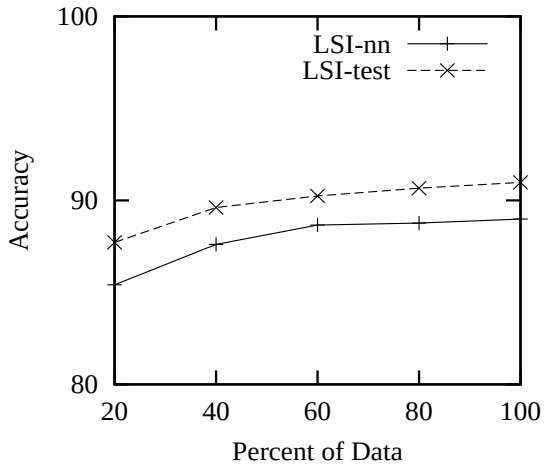

Figure 1: LSI with and without test examples for the two class paper title problem

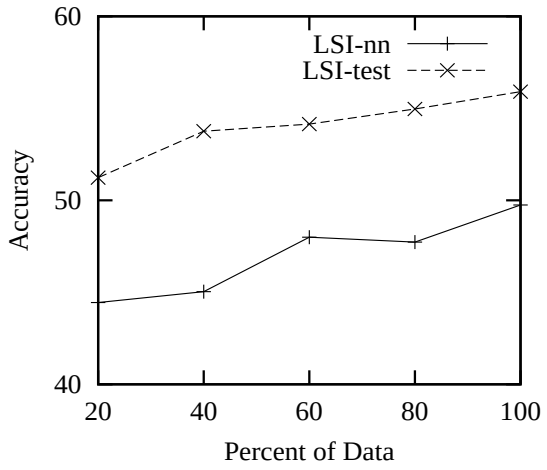

Figure 2: LSI with and without test examples for the netvet problem

Summary

We have presented a method for incorporating test examples into text classification using LSI. The singular value decomposition is performed on a term-by-document matrix that includes both the training examples and test examples. This allows test examples to be compared to the training examples in a new space that reflects patterns in the text in the domain that may not be found when confronted solely with training data. We have shown empirically that this increases the accuracy rates for classification problems.

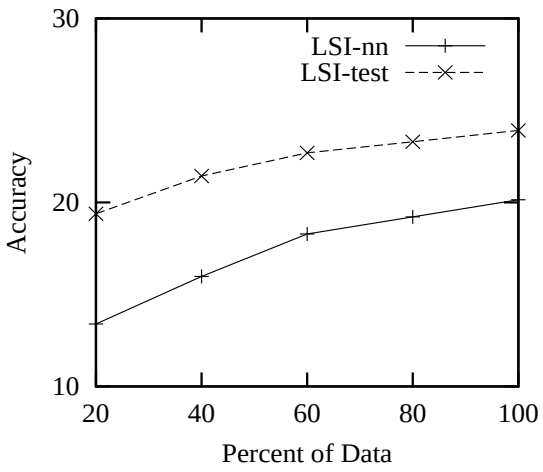

Figure 3: LSI with and without test examples for the business problem

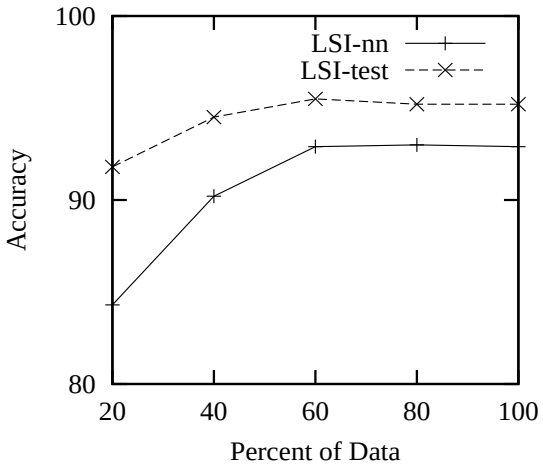

Figure 4: LSI with and without test examples for the news problem

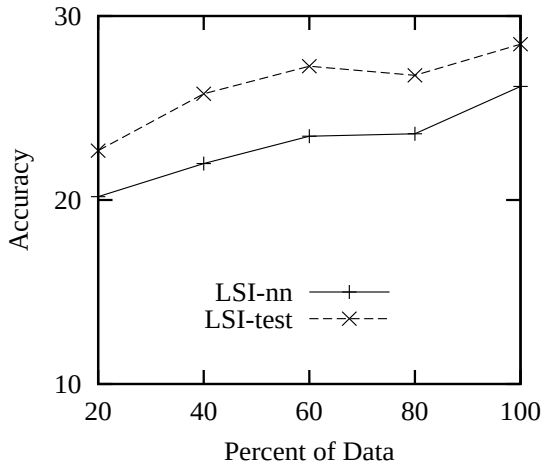

Figure 5: LSI with and without test examples for the thesaurus problem

References

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

Bennet, K., and Demiriz, A. 1998. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems* 12:368–374.

Berry, M. W.; Dumais, S. T.; and Letsche, T. A. 1995. Computational methods for intelligent information access. In *Proceedings of Supercomputing 95*.

Berry, M.; Dumais, S.; and O'Brien, G. W. 1995. Using linear algebra for intelligent information retrieval. *SIAM Review* 37(4):573–595.

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.

Dumais, S. 1995. Latent semantic indexing (LSI): TREC-3 report. In Hartman, D., ed., *The Third Text REtrieval Conference, NIST special publication 500-225*, 219–230.

Dumais, S. 1996. Combining evidence for effective information filtering. In *AAAI Spring Symposium on Machine Learning and Information Retrieval, Tech Report SS-96-07*.

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. 2002. *Learning to Classify Text using Support Vector machines*. Ph.D. Dissertation, University of Dortmund.

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.

Sebastiani, F. 1999. Machine learning in automated text categorization. *Technical Report IEI-B4-31-1999*.

Vapnik, V. N. 1998. *Statistical Learning Theory*. Wiley.

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. 2000. Improving short text classification using unlabeled background knowledge to assess document similarity. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 1183–1190.

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.