Conversational Topic Classification using SVMs and Features Induced by Clustering

Kevin Duh
Department of Electrical Engineering
University of Washington, Seattle, USA
duh@ee.washington.edu

Abstract

This work explores the use of Support Vector Machines (SVM) for topic classification of conversations. An All-vs-One SVM system is used as the baseline. Several methods in feature weight scaling and feature selection are compared. Results suggest that the conversation domain requires a different set of methods from the written text domain. Finally, a feature selection method based on hierarchical clustering is presented with promising results.

1 Introduction

Automatic classification of documents by topics is an ever more important task as the volume of information available at our fingertips explodes. The goal is to assign any given text to a fixed number of predetermined topics. There is a wide variety of applications, ranging from information retrieval to text filtering.

To date, most research in this area has concentrated on classification of *written text* corpora such as news stories (*e.g.*Reuters), Web documents, scientific journal abstracts (*e.g.*OHSUMED), and newsgroups (*e.g.*20 Newsgroups). This work deviates from the trend and explores topic classification of *speech transcripts*. This is particularly relevant in spoken language applications, such as automatic classification of meeting transcripts and real-time language model adaption in speech recognition systems.

The corpus used is part of Switchboard, a collection of five minute telephone conversations between strangers. These volunteers chat about one of 67 topics ranging from "politics" to "hobbies." Like all speech corpora, Switchboard is full of disfluencies (*e.g.*filled pauses like "um" and backchannels like "yeah, uh-huh.") Further, conversations often get off-topic. This noisiness presents a challenge for classification. Also, the distribution of topics is skewed in Switchboard. Together these issues make the problem of conversation classification interesting in its own right.

This paper explores the use of Support Vector Machines (SVM) in this task. There are several motivations for using SVM's:

- **High dimensional feature space** The number of features used to represent text is often proportional to the vocabulary size. The size is often too large, creating overfitting problems. SVM's ability to handle large feature space makes it a natural candidate for this problem.
- **Discriminative vs. Descriptive Classification** A discriminative learner like SVM should be better suited for noisy data like Switchboard. Descriptive learners run the risk of tuning their parameters to noise. SVM's simply minimize empirical risk given the available data.
- **Good performance in written text** SVM's ranks among the top statistical learning algorithms for classification of Reuters and OHSUMED corpora (Joa98b; SJDM98). It is interesting to see if SVM's strengths carry across different corpora.

The goal of this project is to design a SVM classifier system optimized for conversation transcript data. Several issues will be examined in detail:

- How to combine make multi-class decisions given that SVM's are inherently binary classifiers.
- How to scale feature weights to capture the characteristics of spoken data
- How to do feature selection to optimize performance

The paper goes as follows: First, we will review prior work in text classification and

SVM's. Then, the baseline system with a specific multi-class decision scheme is described. Sections 5 and 6 discuss experiments on feature weight scaling and feature selection. In section 6.3, feature selection by hierarchical clustering is discussed in detail. Finally, section 7 presents final evaluation scores and conclusions.

2 Text Classification

Text classification is a rich field with contributions from both information retrieval and machine learning communities. For a good overview, see (Seb02).

Traditionally, Naïve Bayes classifiers show good results despite their simplifying assumptions. Recently, many other methods have been tested. (YL99) presents a methodical comparison of Neural Networks, SVM's, Naïve Bayes, k-Nearest Neighbors (kNN), and Linear Least Squares Fit (LLSF). (SJDM98) compares Rocchio's method, Decision Trees, Naïve Bayes, Bayesian Networks, and SVM's. Both works report that SVM's perform well, though (YL99) notes that all methods perform comparably when there is abundant data. The fact that there is no single winner means that the entire classifier system design, from feature selection and text representation to classifier implementation and tuning, is very important. Thus, in this work, we focus on the designing classifier subparts and examine its effects on overall performance.

One subsystem of significant interest is text representation. Text representation is intriguing because current state-of-the-art systems still employ a bag-of-words approach. In this approach, a text document is represented as a vector where each position corresponds to a word in the vocabulary. Intuitively, we would expect a more "meaningful" representation using syntactic and semantic information to work better. There have been efforts using Natural Language Processing (NLP) techniques. (*e.g.*(CS96)). However, significant improvement has yet to be seen. (Lew92) explains that perhaps the semantically meaningful features derived by humans might not be statistically meaningful for machine learning algorithms. For this reason, this work will use the conventional bag-of-words vector representation.

Once the bag-of-words representation is chosen, the specific feature weight scaling method must be decided. Usually, the bag-of-words vector representation means that the relative count of each word is used to represent the document vector. A popular counting method is *tf*idf* (topic freq. times inverse document freq.), in which a word is counted less if it is common. Another method is binary counting, which represents only the presence or absence of words. Section 5 explores the best feature weight scaling for conversational speech transcripts.

The second subsystem of considerable importance is feature selection. The enormous vocabulary size implies that good feature selection can substantially reduce the risk to overfitting. A vocabulary size on the order of 10 thousand words is not uncommon; in fact, the Switchboard data contains 17,062 unique words after stemming and stopword removal.

Two main approaches exist for feature selection:

- *Term selection* Given an original vocabulary of size r, choose $r^{'} < r$ terms (words) that are "most important." Methods differ by how they select the "most important" words. Section 6 will compare mutual information, document frequency thresholding, and hierarchical clustering as selection criteria.

- *Term extraction* For $r^{'} < r$, choose $r^{'}$ terms that are transformations or combinations of the original terms. If the original terms are words, the extracted terms may not correspond to words at all. In Latent Semantic Analysis (LSA), singular value decomposition on document vectors brings out the most important "directions" (WPW95). In term clustering, a clustering algorithm is applied to form word classes subject to the constraint of minimizing mutual information loss (DMK02). Term extraction is not investigated due to time constraints.

3 Support Vector Machines

SVM's are based on the principle of *structural risk minimization*, which says that the decision surface should minimize test error on unseen samples. In the case of linear SVM's, this surface is a hyperplane that separates the positive and negative training samples by the widest margin. Intuitively, SVM's find the plane that best separates the training data.

Define C as the margin we want to maximize. Let $\mathbf{x_i}$ be one of N training sample vectors and $y_i \in (-1, 1)$ be its corresponding label. The optimal separating hyperplane defined by the

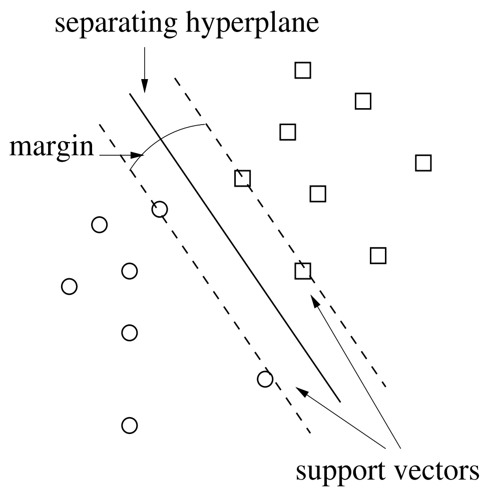


Figure 1: A Linear SVM.

normal vector $\mathbf{w}$ and intercept b is found by the following equation:

$$\max_{\mathbf{w},b,\|\mathbf{w}\|=1} C$$
$$\text{subject to } y_i(\mathbf{w}\cdot\mathbf{x_i}+b)\geq C,\ i=1,\ldots,N. \tag{1}$$

This is equivalent to minimizing the normal vector:

$$\min_{\mathbf{w},b} \|\mathbf{w}\|$$
$$\text{subject to } y_i(\mathbf{w}\cdot\mathbf{x_i}+b)\geq 1,\ i=1,\ldots,N. \tag{2}$$

A quadratic programming algorithm is usually used to solve this contrained optimization problem. A pictorial example of a SVM hyperplane is shown in figure 1.

To allow for data that is not linearly separable, one can introduce a soft margin that allows samples to be on the wrong side of the hyperplane. One can also transform the feature space into a higher dimensional space. The separating hyperplane in that space will map back down into a nonlinear boundary. SVM's employ several "kernel tricks" (*e.g.*radial basis functions, polynomial kernals) to calculate the inner products in these higher spaces effeciently.

There is good theoretical result that bounds the VC dimension of a linear SVM. For the hyperplane described by equations 1 or 2, Vapnik (Vap92) showed that VC dimension d is upper-bounded by:

$$d \leq \min\{(R^2\|\mathbf{w}\|^2), n\}+1 \tag{3}$$

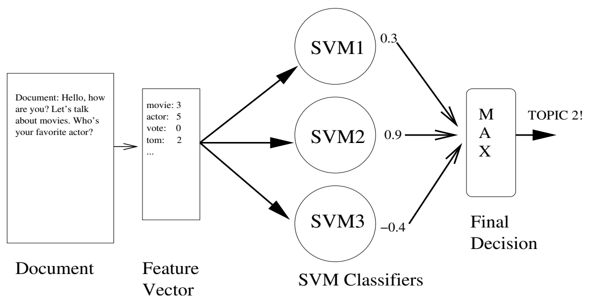


Figure 2: All-vs-One SVM Classifier System

where n is the number and R is the "spread" of training samples, respectively. The important thing to note is that the VC dimension does not depend on the number of features, which means that SVM's can generalize well in high dimensional feature spaces like text (Joa98b).

4 Baseline System

The baseline system is a All-vs-One SVM system using *tf*idf* feature weight scaling and minimal feature selection. All-vs-One is a scheme that combines multiple binary classifiers to make multi-class decisions. If there are K topics, then K SVM's are trained. Each SVM i ($i = 1...K$) is in charge of deciding whether a given document is in its own topic i or not. It is trained by using documents in topic i as positive samples and *all* documents not in topic i as negative samples.

For each test document, the output of each SVM is the positive or negative distance to the optimal separating hyperplane. A large positive value means that the classifier is confident about the document belonging to its class. Thus, the final decision is based on the classifer with the maximum positive confidence (see figure 2).

There exist other schemes for combining binary classifiers. (HL01) compares the All-vs-One against using $K*(K-1)/2$ pairwise classifiers in a voting system and Directed Acyclic Graph (DAG) system, and concludes that accuracy rates are not statistically different. In (Ber99), multiclass-decision is viewed as transmitting information across a noisy channel, and multiple SVM's are used to "decode" the error correcting codes. The latter method would be interesting to try in the context of conversation classification (future research). Nevertheless, the All-vs-One scheme is chosen as the baseline because of its good performance, fast runtime, and quick implementation. These advantages outweight the potential problem of skewed classes.

The specific implementation is as follows: Sixty-seven SVM classifiers are trained using the SVM^{light} program (Joa98a) for the 67 Switchboard topics. Linear SVM's are used due to their their good performance and fast computation. (Joa98b) claims that text is linearly separable, and experiments with radial basis functions and polynomial kernels confirm that. The higher order kernels not only took substantially more computation, but also suffered a decrease in accuracy. This result is similar to (SJDM98), who used Linear SVM's as well.

The method for feature weight scaling is *tf*idf*. For words that are common across topics, the effective feature weight is zero. Rare words that occur frequently only in a few topics are given higher weight. This corresponds to the notion that rare words are more important.

Feature selection is minimal; only stemming and stopword removal is performed. The stopword list comes from Karypis' `doc2mat` script(Kar03) and contains the set of stopword commonly removed for written text classification (*e.g.*words like "you", "if", "too"). Additional speech-only stopwords (disfluencies) are not removed. Although it may be better to derive a linguistically-motivated stopword list for speech, we rely on *tf*idf* and feature selection methods instead.

The baseline system is simple and performs considerably well without any parameter tuning. On the development set, the baseline achieved an accuracy of 76%. The majority of experiments in this work will focus on finding the best feature weight scaling and feature selection to improve upon this result.

5 Feature Weight Scaling

*tf*idf* performs well in written text classification, but its performance in conversation transcript classification has not been reported. In the following experiment, three methods feature weight scaling methods are compared:

- *tf*idf*
- Binary (0/1) representation
- M-ary representation

5.1 *tf*idf*

In *tf*idf*, each word count C is scaled according to the formula:

$$C_{tf*idf} = tf * \log \frac{D}{df} \qquad (4)$$

where tf is the term frequency (*i.e.*normalized number of counts of the word in the document), *df* is the document frequency (*i.e.*number of documents that contain the word at least once), and D is the number of documents in total. If a word is so common that it occurs in almost all documents, then the $\log \frac{D}{df}$ will be close to zero and the word loses importance. On the contrary, a rare word is multiplied in importance by the log term.

In the specific context of topic classification, the document frequency *df* is replaced by the topic frequency. The total number of documents D is replaced by the total number of topics. The reason is that weights should be scaled to discriminate topics. If document frequency were used instead, the skewed class distributions of Switchboard will place an extra bias on some words.

5.2 Binary Representation

In a binary representation, each word is counted according to its presence or absence:

$$C_{binary} = \begin{cases} 1 & \text{if word count} \geq 1 \\ 0 & \text{if word count} = 0 \end{cases}$$

At first sight, it might seem that binary representation throws away valuable information about the specific distribution of word counts. However, if we consider that much of the data is noisy, the limited resolution of binary representation can actually provide some noise-robustness. Counting absence/presence of words is inherently less variable than counting the number of words.

Binary representation is built on the notion that knowledge of what words are present and absent is sufficient for discriminating between classes. This is true for topics that are unrelated: for example, suppose a test document contains the words "voter," "elections", and "polls," and does not contain the words "bass," "tents," and "cats." Then immediately topics like `fishing`, `camping`, and `pets` can be eliminated; further, the topic `elections` easily becomes top candidate. If each topic contains a core set of distinctive words, then binary representation works.

5.3 M-ary Representation

M-ary representations can be seen as a middle-ground between hard binary representation and raw word counting (similar to *tf*idf*). Instead

Method	Accuracy (%)
*tf*idf*	76.67
binary	80.70
ternary (0/1/2)	83.16
10-ary	87.02
Raw count (∞-ary)	84.21

Table 1: Comparison of feature weight scaling methods. The scores represent the precision tested on a development set

of thresholding the counts to be no greater than 1 (as in binary representation), the counts can be from zero to M:

$$C_{M-ary} = \begin{cases} M & \text{if word count} \geq M \\ C_W & \text{if word count } C_W < M \end{cases}$$

The motivation for this approach comes from the fact that binary representation performs poorly when distinguishing between similar topics. In the above example, where a document contains the words "voter," "elections," and "polls," it is easy to rule out most topics in the Switchboard task. However, very similar topics do exist, and a more fine grain word count distribution is necessary to discriminate them. For example, to discriminate between the topics `elections` and `politics`, the distribution of word counts might be more helpful. An M-ary representation tries to achieve both the noise-robustness of hard thresholds and fine-grain discriminatory power of word counting methods.

5.4 Experiments with Feature Weight Scaling

The three feature weight scaling methods are tested on a held-out development set. In the M-ary case, three cases of M are tested (2, 10, ∞). The training set consists of 797 conversations, while the development set consists of 570 samples. All documents are preprocessed with only stemming and stopword removal to allow comparison with the baseline system (section 4).

Performance is measured by accuracy (precision), which is the number of correct classifications out of the entire development set. Table 1 summarizes the results.

Results show that the baseline *tf*idf* method performs poorly in comparison to binary and M-ary representations. This could imply that conversations are indeed substantially noisier than written text. Specifically, when a speaker gets off-topic, the rare off-topic words will be viewed as important words in the *tf*idf* representation! In this case, the noise-robustness of binary and M-ary schemes are critical.

Comparing the performance of binary versus M-ary methods is interesting. M-ary performs better on all cases of M, which implies that fine-grain word distribution is important. The performance of M-ary relative to binary can be seen as a rough guide of how similar Switchboard topics are. Further, M-ary drops in performance as the M approaches ∞, indicating that noisiness could be a problem again. Empirical tuning showed the best results with $M = 10$ on this data set, so subsequent experiments will use 10-ary representation.

6 Feature Selection

Feature selection is important for large feature space problems like text classification. In the following experiment, three feature selection methods are compared:

- Mutual Information Criteria
- Document Frequency (DF) Thresholding
- Feature Selection by Hierarchical Clustering

6.1 Mutual Information

Feature selection by mutual information works on the notion that words that have high mutual information with topics are the important words. Intuitively, high mutual information means that the joint occurence of the word and topic is relatively higher than their independent occurences. The following formula (YP97) describes the mutual information between a word w and a topic t:

$$I(w,t) = \log \frac{P(w,t)}{P(w)P(t)} \tag{7}$$

To find the mutual information of a specific word to all topics, we can either take the maximum or the average:

$$I_{\max}(w) = \max_i \{I(w,t_i)\} \tag{8}$$

$$I_{ave}(w) = \sum_i P(t_i) I(w,t_i) \tag{9}$$

The max version performed better, so it is the version used in subsequent experiments (I think the average version suffered from numerical imprecisions because some $P(t_i)$ are very small). The selected features are the words with the highest mutual information values.

6.2 DF Thresholding

DF Thresholding operates on the notion that rare words are noise, and therefore should be deleted. In this method, a word is included as a feature only if it is above a predetermined threshold (usual values are 1 to 10).

6.3 Feature Selection by Hierarchical Clustering

Clustering has been used to drive feature selection. Prior work has focused much on clustering words directly to reduce features (DMK02). The idea is similar, in spirit, to vector quantization for compression. However, the approach that will be described here stems from a different idea and clusters on different objects. Rather than clustering words, we cluster topics. Feature selection of words becomes an indirect by-product of the clustering process.

The basic motivation is the idea that all feature selection methods described so far are global optimization methods. In mutual information, the term is ranked according to its mutual information relative to *all* the topics. In DF thresholding, *all* documents, whether they are the same topic or not, are pooled together to calculate document frequency of a word.

Feature selection by hierarchical clustering starts with the question: What if feature selection is performed locally rather than globally? We know that some topics are more similar than others. Words that are discriminative for one topic pair may not be useful for another topic pair. Perhaps finding features by looking at pairs of topics one at a time may have greater impact on performance than looking for features in all topics at once.

A natural way to observe pairs of topics one at a time is to use an agglomerative clustering algorithm. In agglomerative clustering, two nodes are merged if the result induces the least loss of within-cluster similarity and between-cluster distance. The words that contribute most to the merge decision are selected as the most important local features.

The algorithm overview:

1. Begin with 67 topics, each as its own cluster
2. Apply agglomerative clustering, merging a pair of topics each time
3. Pick words that contribute most to the merging decision. These are the locally discriminative words.
4. Iterate and collect the discriminative words for each merge. Stop when only one cluster remains.
5. The union of all collected words will be the feature set

6.4 Clustering Implementation and Output

The CLUTO clustering toolkit (Kar03) is used to run agglomerative clustering. Clusters are compared by their cosine similarity measure. Let $\mathbf{x}$ and $\mathbf{y}$ be the vector representation of two documents. Then their cosine similarity is defined as:

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} \tag{10}$$

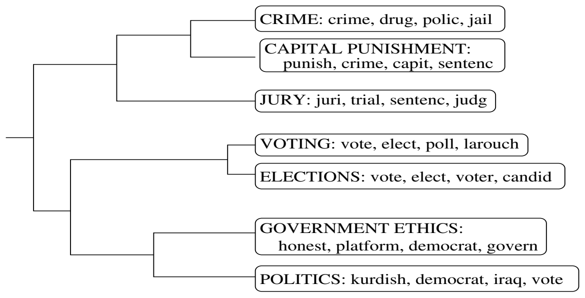
The aggregation criteria is defined as:

$$\max \sum_t \sqrt{\sum_{u,v \in S_t} sim(u, v)} \tag{11}$$

S_t is the set of current clusters. u and v are elements in one cluster, and the sum $\sum_{u,v \in S_t} sim(u, v)$ computes the within-cluster similarity on S_t. This is computed for all clusters and summed to provide the total within-cluster similarity of some choice of merging. Different merge hypotheses are evaluated based on this total within-cluster similarity for different merging scenarios. The scenario with the highest total within-cluster similar is chosen as the agglomeration.

The contribution of individual words to this agglomeration can be calculated by evaluating the percentage of the total within-cluster similarity resulting from that word. Words that ultimately contribute a lot to total within-cluster similarity can be seen as words that contribute a lot to between-cluster distance. These words are chosen as the features.

Figure 3 shows an example hierarchy tree produced by agglomerative clustering. The leaves represent the topics in Switchboard. The (stemmed) words on the right of each node corresponds to the most important words for that merge. Most of the selected features are semantically related to the topic. Further, a natural hierarchy of topic similarity emerges. The feature selection algorithm will pick the top k words from each node and add them to the feature set. In this example, $k = 4$ and the 24 unique words are selected as features.



CRIME: crime, drug, polic, jail
CAPITAL PUNISHMENT: punish, crime, capit, sentenc
JURY: juri, trial, sentenc, judg
VOTING: vote, elect, poll, larouch
ELECTIONS: vote, elect, voter, candid
GOVERNMENT ETHICS: honest, platform, democrat, govern
POLITICS: kurdish, democrat, iraq, vote

Figure 3: Example of hierarchy (dendrogram) produced by agglomerative clustering. The leaf nodes represent separate Switchboard topics. The (stemmed) words are the four words from each topic that contribute most to merging. They are the words used in the final feature set

Method	**Vocab Size (words)**	**Accuracy (%)**
No selection	17,062	87.02
Mutual info	10,000	82.28
DF Threshold (thresh=5)	3,447	88.42
Cluster-feature (size=30)	1,528	90.00

Table 2: Comparison of feature selection methods. The best accuracy individually optimized for each method is shown. All methods use 10-ary feature weight scaling

6.5 Experiments with Feature Selection

Table 2 shows the resulting vocabulary size and classification accuracy of different feature selection methods. The optimized version of each method is shown so that cross-method comparisons can be made.

Mutual Information feature selection actually performed worse than the baseline (which has no feature selection except stemming and stopword removal). An hypothesis is that mutual information is biased toward very rare words, which could be off-topic words in conversations. Equation 7 can be rewritten as:

$$I(w,t) = \log \frac{P(w|t)}{P(w)} \tag{12}$$

Thus, if two words both have the same class-conditional $P(w|t)$, but one word has a smaller marginal $P(w)$, that rarer word is biased to have larger mutual information. These high mutual information features may mistakenly contain many off-topic words.

Both DF thresholding (at threshold 5) and cluster-driven features improved the performance of the SVM. The DF Thresholding result implies that common words do matter in classification, while the cluster-feature result implies that locally chosen words is a step in the right direction.

It is interesting to see whether the combination of DF thresholding and cluster-driven features will improve results, since they are based on different concepts and may complement each other. Interestingly, experiments on both the union and intersection of their feature sets did not exhibit improvements. Instead the accuracy using the combined feature set seem to be the average of individual feature set accuracies. (For example, DF threshold at 9 has accuracy 86.8% and cluster-feature at 30 features per node has accuracy 90.0%. The intersection and union of their feature sets gave accuracies of 89.8% and 87.0%, respectively.)

7 Final Evaluation and Conclusions

Several feature weight scaling and feature selection methods were tested on a All-vs-One SVM system. Results on Switchboard indicate that the best system consists of 10-ary feature weight scaling and uses a cluster-driven feature set.

Interestingly, both *tf*idf* and Mutual Information feature selection performed poorly. These are the most popular methods used in conventional topic classification of written text. This suggests that methods that work well in one type of text cannot be easily transferred to another type of text without parameter tuning. Especially, the fact that conversations tend to wander off-topic seem to degrade methods that view rare words as the most informative words.

The final evaluation results were consistent with experiments on the development set. For the final evaluation, the development set and training set were pooled together, and the SVM system was retrained. Part of the convenience of SVM's is that no hand-tuning is necessary at this step. The new optimal separating hyperplane is simply the theoretical best for minimizing empirical risk. The accuracy of the final evaluation set is 91.85%.

There are many interesting new paths for further study. It will be interesting to examine cluster-driven feature selection in more detail. Why did it work? What does this imply about the nature of conversation topics? Furthermore, a promising direction is the incorporation of cluster-driven features in a hierarchical SVM system. Rather than an All-vs-One system for multi-class decision, binary classifiers are placed at each node on the hierarchy tree. There are results showing that hierarchical classification allows the classifier at each node to focus on the features that matter. (KS97). It will be interesting to see if cluster-driven features improve hierarchical classification.

Acknowledgement

I would like to thank to Prof. Mari Ostendorf and Prof. Maya Gupta for inspiring discussions and guidance. Also, thanks go to Jeremy Kahn, Dustin Hillard, and Bill McNeill for mutual encouragement!

References

A. Berger. Error-correcting output coding for text classification, 1999.

William Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In *SIGIR-96*, 1996.

Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. Enhanced word clustering for hierarchical text classification. In *Proceedings of the eigth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 191–200, 2002.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. Technical report, National Taiwan University, Department of Computer Science, 2001.

T. Joachims. Making large-scale support vector machine learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Machines.* MIT Press, Cambridge, MA, 1998.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of the European Conference on Machine Learning*, 1998.

George Karypis. Cluto: A clustering toolkit. Technical Report 02-017, University of Minnesota, Department of Computer Science, November 2003.

Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In *Proceedings of ICML-97, International Conference on Machine Learning*, 1997.

David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34, March 2002.

S.Dumais, J.Platt, D.Heckerman, and M.Sahami. Inductive learning algorithms and representation for text categorization. In *Proceedings of ACM-CIKM98*, 1998.

V. Vapnik. *Estimation of Dependencies Based on Empirical Data.* Springer-Verlag, 1992.

Erik Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR)*, pages 317–332, 1995.

Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of SIGIR-99*, 1999.

Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.