Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization

Hwee Tou Ng
DSO National Laboratories
20 Science Park Drive
Singapore 118230
nhweetou@dso.org.sg

Wei Boon Goh
Ministry of Defence
Gombak Drive
Singapore 669638

Kok Leong Low
Ministry of Defence
Gombak Drive
Singapore 669638

Abstract

In this paper, we describe an automated learning approach to text categorization based on perceptron learning and a new feature selection metric, called correlation coefficient. Our approach has been tested on the standard Reuters text categorization collection. Empirical results indicate that our approach outperforms the best published results on this Reuters collection. In particular, our new feature selection method yields considerable improvement.

We also investigate the usability of our automated learning approach by actually developing a system that categorizes texts into a tree of categories. We compare the accuracy of our learning approach to a rule-based, expert system approach that uses a text categorization shell built by Carnegie Group. Although our automated learning approach still gives a lower accuracy, by appropriately incorporating a set of manually chosen words to use as features, the combined, semi-automated approach yields accuracy close to the rule-based approach.

1 Introduction

We live in a world of information explosion. The phenomenal growth of the Internet has resulted in the availability of huge amounts of online information. Much of this information is in the form of natural language texts. Hence, the ability to catalog and organize textual information automatically by computers is highly desirable. In particular, a computer system that can categorize real-world, unrestricted English texts into a predefined set of categories would be most useful.

In this paper, we present an automated learning approach to building a robust, efficient and practical text categorization system, called CLASSI, using the perceptron learning algorithm. We also describe a new feature selection metric, called correlation coefficient, which yields considerable improvement in categorization accuracy. When tested on the standard Reuters text categorization collection, our approach outperforms the best published results on this Reuters corpus.

We also conducted a usability case study by comparing the performance of such an automated learning approach with the more traditional, rule-based "expert system" approach of building text categorization systems. In the rule-based expert system approach, the developer of the system manually codes up a set of rules to categorize texts. In contrast, our learning approach alleviates the knowledge acquisition bottleneck inherent in a rule-based approach.

As comparison, we use an existing text categorization system, TCS, developed using a text categorization shell built by Carnegie Group [Hayes *et al.*, 1990]. The input to TCS are newswire articles and the output categories form a tree. Our evaluation indicates that a completely automated learning approach still gives lower accuracy. However, by manually modifying and augmenting the set of words to be used as features in a topic categorizer, we achieve accuracy very close to the manual rule-based approach. This suggests that at present, a semi-automated approach is perhaps the best way to build a high performance text categorization system.

The rest of this paper is organized as follows. Section 2 gives a description of the text categorization task. Section 3 discusses the text representation and feature selection metric used. Section 4 describes the perceptron algorithm used. Section 5 presents the empirical results achieved by our approach on the standard Reuters corpus. Section 6 describes the case study conducted to compare our automated learning approach with TCS. This is followed by Section 7 on related work, and Section 8 gives the conclusion.

2 Task Description

The input to our text categorization system, called CLASSI (CLASsification System for Information), consists of unrestricted English texts. The system is also given a set of predefined categories. There is no restriction as to what can form a category. For example, a category can be about a particular country (like USA, Japan), a particular subject topic (like economics, politics), etc. One text can belong to more than one categories if it mentions multiple topics (like in a long text). Also, the categories need not be exhaustive – some text may belong to none of the pre-defined set of categories.

Unlike most existing work on text categorization, we allow the categories to form a tree. We will describe in greater detail how hierarchical categorization is achieved when we discuss the usability case study in Section 6.

Given an input text, a text categorization system assigns zero, one or more categories to the text.

3 Text Representation and Feature Selection

To use an automated learning approach, we first need to transform a text into a feature vector representation. This transformation process requires the appropriate choice of features to use in a feature vector. These feature vectors form the training examples. Feature vectors that are derived from the relevant texts of a category C form the positive training examples for the category, while the feature vectors derived from the irrelevant texts of category C form the negative examples. Next an automated learning algorithm learns the necessary association knowledge from the training examples to build a classifier for each category C. In this section, we focus on the text representation and feature selection issues, while the next section discusses the perceptron learning algorithm used.

We use single words as the basic units to represent text. A "word" is defined as a contiguous string of characters delimited by spaces. Specifically, each text is pre-processed in the following steps:

1. Punctuation marks are separated from words.
2. Numbers and punctuation marks are removed.
3. All words are converted to lower case.
4. Words like prepositions, conjunctions, auxiliary verbs, etc., are removed. These 290 stop words are those given in [Lewis, 1992].
5. Each word is replaced by its morphological root form. For example, plural nouns like "interests" are replaced with the singular form "interest", inflectional verb forms like "ate", "eaten", "eating", etc., are replaced with the infinitive form "eat", and so on.

We use the morphological routines from WordNet [Miller, 1990] to convert each word into its morphological root form. The preprocessing speed of Classi is fast – about 4,000 words per second on a Pentium personal computer.

The remaining words after preprocessing are potential candidates for use as features, with each word as one feature in the feature vectors. Feature selection refers to the process of choosing a subset of these remaining words to use as features to form the training examples.

Previous research on text categorization [Apte *et al.*, 1994] suggests two possible ways in which the words to be used as features can originate: from the relevant texts only (local dictionary), or from both the relevant and irrelevant texts (universal dictionary). Apte et al. reported results indicating that local dictionary gives better performance. In our work, we also found that local dictionary gives better accuracy. In particular, results from our case study show that local dictionary gives considerably higher performance (see Section 6). Hence, we only use the local dictionary method on the Reuters corpus.

We also require a word to occur at least five times in the training texts to be chosen as a feature. This measure is quite widely used, for example in the work of [Hearst *et al.*, 1996] and [Lewis and Ringuette, 1994]. This is beneficial since infrequent words are not reliable indicators for use as features.

After words are chosen according to the local dictionary method, and after eliminating words with infrequent occurrence, we select a set of n features by applying a feature selection metric. The features chosen are the top n features with the highest feature selection metric score. We experimented with three feature selection metrics: *correlation coefficient*, χ^2, and frequency.

We define the correlation coefficient $\mathcal{C}$ of a word w as:

$$\mathcal{C} = \frac{(N_{r+}N_{n-} - N_{r-}N_{n+})\sqrt{N}}{\sqrt{(N_{r+}+N_{r-})(N_{n+}+N_{n-})(N_{r+}+N_{n+})(N_{r-}+N_{n-})}}$$

where N_{r+} (N_{n+}) is the number of relevant (non-relevant) texts in which the word w occurs, and N_{r-} (N_{n-}) is the number of relevant (non-relevant) texts in which the word w does not occur.

Our correlation coefficient is a variant of the χ^2 metric used in [Schutze *et al.*, 1995], where $\mathcal{C}^2 = \chi^2$. $\mathcal{C}$ can be viewed as a "one-sided" χ^2 metric. The rationale behind the use of our new correlation coefficient $\mathcal{C}$ is related to the finding that local dictionary yields a better set of features as reported in [Apte *et al.*, 1994] (and confirmed in our own work). That is, we are looking for words that only come from the relevant texts of a category C and are indicative of membership in C. Words that come from the irrelevant texts or are highly indicative of *non-membership* in C are not as useful. The correlation coefficient $\mathcal{C}$ selects exactly those words that are highly indicative of membership in a category, whereas the χ^2 metric will not only pick out this set of words but also those words that are indicative of non-membership in the category. Our empirical results suggest that using words from the relevant texts and are indicative of membership in a category is better than using words that are indicative of membership as well as non-membership of a category.

The third feature selection metric we experimented is frequency, which selects words that occur most frequently in the training texts to use as features.

The value of a feature in a feature vector is the normalized frequency of the corresponding word in the training text. We use normalized frequencies so that training texts of different lengths are normalized to contribute equally during training.

Let $t_1, t_2, \ldots, t_n$ be the set of words chosen as features when building the classifier for a category C. Then

$$(t_1\ f_1)(t_2\ f_2)\ldots(t_n\ f_n)$$

is the derived training example, where f_i is a normalized frequency.

When training the classifier for a category C, we use all the texts in the training corpus that belong to C as the positive training texts. On the other hand, it is often the case that there are many non-relevant texts not belonging to category C in the training corpus. We employ the same technique described in [Hearst *et al.*, 1996] to select a subset of the non-relevant texts to use as the negative training texts. These texts are the most relevant non-relevant texts. First we form the vector sum of all the positive training vectors. Then the negative training vectors are ranked by their dot product score with the positive aggregate vector. The higher the dot product score, the more relevant the negative text is to the category.

4 Perceptron Learning

Having chosen a word-frequency list representation of text, we now consider the task of building a classifier for a category C. Let $t_1, t_2, \ldots, t_n$ be the set of words chosen as features based on a set of training texts for C, as described in the last section. Given a new text T to be classified, Classi first pre-processes the text as described earlier. The feature vector representation of the new text T is

$$(t_1\ f_1)(t_2\ f_2)\dots(t_n\ f_n)$$

where $f_1,\dots,f_n$ are the normalized frequencies of the word occurrences in T.

Our classifier arrives at a classification decision by finding an appropriate set of real-valued weights $w_0, w_1, \dots, w_n$ such that

$$\sum_{j=0}^{n} w_j * f_j \geq 0$$

if and only if T belongs to category C. (We let $f_0 = 1$ in computing the weighted sum of frequencies.)

As our classifier arrives at a decision by taking a linearly weighted summation, it functions as a *linear threshold unit* (LTU). That is, our classifier is a linear classifier. The perceptron learning algorithm (PLA) [Rosenblatt, 1958] is a well-known algorithm for learning such a set of weights for an LTU, and we use this algorithm in CLASSI.

Let $E_1, E_2, \dots, E_l$ be the positive examples derived from the relevant texts of category C, and let $E_{l+1}, E_{l+2}, \dots, E_m$ be the negative examples derived from the non-relevant texts not of category C. Let E_i be of the form

$$(t_1\ f_{i_1})(t_2\ f_{i_2})\dots(t_n\ f_{i_n})$$

We set $f_{i_0} = 1$ for $i = 1,\dots,m$. The goal of the perceptron learning algorithm is to find a set of weights $w_0, w_1, \dots, w_n$ such that

$$\sum_{j=0}^{n} w_j * f_{i_j} \geq 0 \qquad 1 \leq i \leq l$$

and

$$\sum_{j=0}^{n} w_j * f_{i_j} < 0 \qquad l+1 \leq i \leq m$$

Although PLA is a well-known algorithm, for completeness sake, we give here a formal description of PLA in Table 1. PLA is essentially a hill-climbing, gradient-descent search algorithm. It starts with a random set of weights and iteratively refines the weights to minimize the number of misclassified examples. η is a constant that controls the learning rate. We set $\eta = 0.35$ in all our evaluation runs. Since PLA may not converge or converge too slowly in practice, we also set the maximum number of iterations to 300 in all our evaluation runs.

Note that as part of the process of deciding whether a new text belongs to a category, the classifier computes a linearly weighted summation $\sum_{j=0}^{n} w_j * f_j$. This weighted sum can be taken as a measure of the degree of membership of a text in a category. As such, besides being able to decide whether a new text belongs to a category, we can also use this weighted sum to rank a set of new texts from the most closely matching text to the least matching text.

We define *recall* (R) as the ratio of truly relevant texts that are classified by CLASSI as relevant, and *precision* (P) as the ratio of texts classified as relevant by CLASSI that are truly relevant.

1. `Initialize the weights` $W = (w_0\ w_1\ \dots\ w_n)$ `to random real values.`

2. `Compute the weighted sum of frequencies for all training examples` E_i`:`

$$\sum_{j=0}^{n} w_j * f_{i_j}$$

3. `If all positive examples have non-negative sum and all negative examples have negative sum, then output the weights and stop.`

4. `Else compute the vector sum` S `of the misclassified examples. That is, if` E_i `is a positive example that is misclassified as negative, then`

$$S \leftarrow S + (f_{i_0}\ f_{i_1}\ \dots\ f_{i_n})$$

`Conversely, if` E_i `is a negative example that is misclassified as positive, then`

$$S \leftarrow S - (f_{i_0}\ f_{i_1}\ \dots\ f_{i_n})$$

5. `Update the weights as follows and go to step 2:`

$$W \leftarrow W + S \cdot \eta$$

`where` η `is a constant scale factor.`

Table 1: The perceptron learning algorithm

5 Reuters Test Corpus

In order to compare the performance of CLASSI with other state-of-the-art text categorization systems, we tested CLASSI on a standard test collection for text categorization used in the literature. This collection of texts, known as Reuters-22173, consists of Reuters newswire articles about financial categories [Lewis, 1992].[1] This text corpus has about 3.4 million words (occupying 23 MB) with 135 categories and 22,173 texts.

We used the same training/testing set split of [Apte *et al.*, 1994]. First, 723 texts used as testing in a separate study are removed from consideration. Of the remaining texts, some do not have any category assigned to them. After ignoring such texts, the training set consists of 10,666 texts (dated on or before 7 April 87) and the testing set consists of 3,679 texts (dated on or after 8 April 87). As in [Apte *et al.*, 1994], we consider only 93 categories which occur more than once in the training texts.

For the Reuters corpus, we choose the features using the local dictionary method, which was found to yield better results on this corpus by [Apte *et al.*, 1994]. That is, the words are only taken from the positive training texts. For each category C, we used all training texts belonging to C as the

[1] Available by anonymous ftp from ftp.cs.umass.edu at /pub/doc/reuters1, courtesy of Reuters, Carnegie Group, and David Lewis.

positive examples of C, and the top 3000 most relevant non-relevant texts as the negative examples of C. The technique to pick the most relevant non-relevant texts are described in Section 3.

The number of features chosen is an important parameter that affects the performance of CLASSI on the Reuters corpus. We experimented with several different number of features as listed in Table 2. We also show the effect of the three feature selection methods used. The accuracy measure in Table 2 is the micro-averaged break-even points, the same measure as used in [Lewis and Ringuette, 1994]. At a break-even point, recall and precision are the same. Micro-averaging combines the recall and precision values of all the categories by summing the true positive, true negative, false positive, and false negative counts across all categories.

From Table 2, it is evident that our new feature selection method based on correlation coefficient consistently outperforms the χ^2 and frequency selection method at all feature sizes. Also, performance improves as more features are used. We plan to investigate more thoroughly the relationship between feature set size and performance, and finding the optimal feature set size where the performance peaks. As the number of features used has a significant impact on accuracy, it may be beneficial to apply cross-validation techniques like those of [Kohavi and John, 1995] to automatically determine the best number of features to use for each category from the training examples.

Previous published test results on this training/testing set split of the Reuters corpus include the system SWAP-1 of [Apte *et al.*, 1994], RIPPER and EXPERTS of [Cohen and Singer, 1996], and an implementation of Rocchio's algorithm [Rocchio, 1971] by Cohen and Singer [Cohen and Singer, 1996].

We list in Table 3 the best micro-averaged break-even points achieved by CLASSI, RIPPER, SWAP-1, EXPERTS and Rocchio. The accuracy figures listed are based on representations that do not give special treatment to the headlines of a text. CLASSI outperforms all previously published results on the Reuters corpus.

Wiener et al. [Wiener *et al.*, 1995] also tested their neural network approach on the Reuters corpus. Although they reported break-even point of 0.820, the list of categories they consider is different from the 93 categories reported here and so the results are *not* directly comparable. For example, they consider categories like cbond, loan, ebond, gbond, tbill and tbond which are not among the 93 categories considered in our present study. (See the list of 135 categories in Figure 8.1, Chapter 8 of [Lewis, 1992] from which the 93 categories are chosen.)

6 A Usability Case Study

To evaluate the usability of our automated learning approach, we also conducted a case study by comparing the performance of our approach with an existing text categorization system. This system, called TCS, was built using a text categorization shell developed by Carnegie Group [Hayes *et al.*, 1990]. TCS was built using a rule-based expert system approach. The input to TCS are daily newswire articles. The output categories of TCS form the leaf nodes of a tree. A fragment of this hierarchical organization of the categories is shown in Figure 1. The first level denotes the division into various countries, such as USA, Japan, Australia, etc. The second level denotes the division into primary subject topics, such as economics, politics, etc. The third level denotes detailed subject topics, such as the subdivision of politics into law and political party. As an example, a text that talks about passing a legislative bill in the US Senate will come under the leaf category USA-politics-law. There are 160 leaf categories in TCS. It took about 1.5 person-years to develop the rules needed for categorization in TCS.

To achieve hierarchical categorization, CLASSI forms the internal, non-leaf categories. An internal, non-leaf category denotes the union of all its children categories. CLASSI builds one classifier for each category (leaf and non-leaf node) in the tree. The output categories of an input text can be zero, one or more leaf categories in the tree. When an input text is presented to CLASSI, it first checks for each country at the top level to see if the text belongs to any of the country category. If not, then the text does not belong to any category. However, if a text belongs to a country according to the classifier built for that country category, CLASSI then recursively checks for membership in the categories of the subtree rooted at that country category. If at any node, it is determined that the input text does not belong to any of its children categories, then categorization stops for that branch of the recursion. The recursive process terminates at a leaf category.

As the number of categories increases, this hierarchical approach to categorization is more efficient than a linear approach which considers every category sequentially. To our knowledge, no previous work on text categorization dealt with a tree or hierarchy of categories.

The positive training texts of a leaf category C are the relevant texts of category C. For a non-leaf, internal category C, the positive training texts are taken from the descendant leaf node categories under C, with an equal number of texts from each descendant leaf node category. We used 100 positive training texts for each category.

For negative training texts of a category C, we used the positive training texts that belong to the sibling categories of C. For example, for the category USA-economics, we used texts belonging to USA-politics-law, USA-politics-party, etc. as the negative training texts. We used 200 negative training texts for each category, selecting the most relevant non-relevant texts if more than 200 negative texts are available.

The total size of our training corpus is about 20MB. Since we do not have training texts with their assigned categories verified by human, the categories of the training texts that we use to train CLASSI are those assigned automatically by the TCS system. Since these training text categories are only about 75% accurate, the use of such noisy training corpus tends to lower the accuracy of CLASSI.

The number of features in the training examples we used for the first, second and third level of the category tree is 10, 200, and 200, respectively. We used only 10 features for a country category since it tends to have only a smaller number of indicative features.

To compare the accuracy of CLASSI versus TCS, we manually assigned the correct categories to a new, randomly chosen set of 350 texts not used in the training corpus. The size of this test corpus is about 100,000 words.

Table 4 lists the performance figures of successive versions of CLASSI as compared to TCS. The F-measure [Rijsbergen, 1979] is defined as

$$F = \frac{2PR}{P+R}$$

where P is the precision and R is the recall. We use the F-measure that gives equal weightage to both recall and precision.

Feature Selection	20 features	50 features	100 features	200 features
Correlation coeff.	0.784	0.792	0.799	0.802
χ^2	0.742	0.771	0.790	0.794
Frequency	0.717	0.763	0.778	0.785

Table 2: Effect of Feature Selection Method and Feature Set Size on Break-even point

System	Option	Break-even point
Classi	200 features	0.802
Ripper	negative tests	0.796
SWAP-1	80–100 freq feat.	0.789
Experts	3-words	0.759
Rocchio		0.745

Table 3: Results on the Reuters test corpus

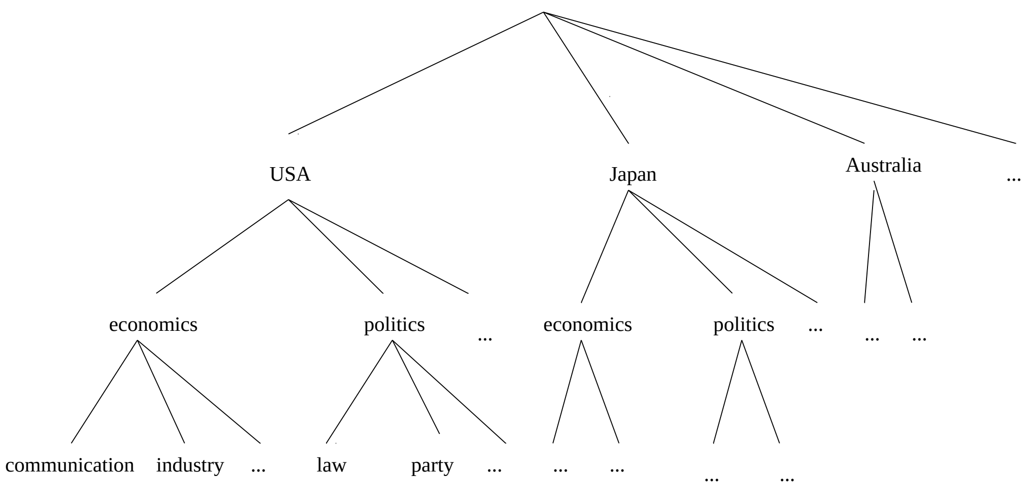

Figure 1: The tree of categories

Method	Recall	Precision	F-measure
(1) +,- texts; freq.	0.163	0.072	0.100
(2) + texts; freq.	0.467	0.203	0.283
(3) + texts; χ^2	0.452	0.427	0.439
(4) + texts; corr. coeff.	0.482	0.463	0.472
(5) words from children cat.	0.495	0.475	0.485
(6) two-cycle generation	0.533	0.512	0.522
(7) manual features	0.736	0.719	0.728
Tcs	0.778	0.694	0.733

Table 4: Successive improvements to Classi and Comparison with Tcs

Version (1) of CLASSI listed in Table 4 uses universal dictionary where the features come from both positive and negative training texts, and relies on frequency to select the features. Its F-measure performance is only 0.1. Versions (2) - (4) switch to using local dictionary where the features are taken from the positive training texts only. Versions (2), (3) and (4) use the frequency, χ^2, and correlation coefficient metric, respectively, to select the features. Again, correlation coefficient achieves higher accuracy compared with the χ^2 and frequency metrics.

In version (5) of CLASSI, we add to version (4) the following: for the second level categories (general subject topics) in the category tree, CLASSI uses the features from the leaf children categories of a subject category C as C's features. This results in a moderate improvement in accuracy.

From version (5), we added a new training method, called *two-cycle generation*, to yield version (6) of CLASSI. This new training method is related to the rationale of wanting to use only words that are indicative of membership of a category, and not words that are indicative of non-membership. This is the same rationale behind the use of local dictionary and our correlation coefficient. Basically, in two-cycle generation, after a set of features is chosen and a classifier is formed by the perceptron learning algorithm, we discard the features with negative weights assigned by the perceptron learning algorithm. The remaining set of features then form the final set of features used, and the perceptron algorithm is applied again to learn a new classifier. This two-cycle generation method yields considerable improvement in accuracy, as shown in Table 4.

Up till version (6), we have applied completely automated learning techniques, with no special use of domain-specific or manual efforts. We achieved an F-measure accuracy of 0.522, which is still substantially lower than the accuracy of 0.733 achieved by TCS. We now decide to use some manual engineering efforts. In particular, we incorporate a set of manually chosen words to use as features, but *only* for the country categories at the top level of the category tree. This manually chosen set of words is obtained by pruning some of the non-indicative words found by our automated learning method, and adding the words that were used in TCS for the country categories. Using these manually chosen words as features, and the same set of training texts as in previous versions, we obtained version (7) of CLASSI. The performance of CLASSI now reaches 0.728, which is only about 0.5% below the accuracy of 0.733 achieved by TCS. This result is encouraging, especially considering that the training texts used by CLASSI is noisy (since the training categories are assigned by TCS which contain mistakes).

Thus, our case study suggests that at present, a semi-automated approach is perhaps the best way to build a high performance text categorization system. Existing learning methods are good at tuning a set of weights, compared with manual engineering of weights. However, feature selection methods are still not good enough, especially for training sets of smaller size, in order for a useful set of features to be selected.

The categorization speed of CLASSI is quite fast. A daily collection of newswire articles, averaging about 2,000 texts and more than 500,000 words (more than 3.5MB) takes only about 40 minutes to be categorized on a Pentium PC. Hence, CLASSI is a practical system that runs efficiently.

7 Related Work

Many learning methods have been applied to text categorization [Schutze *et al.*, 1995], including decision rule induction [Apte *et al.*, 1994], decision tree induction [Lewis and Ringuette, 1994], nearest neighbor algorithms [Masand *et al.*, 1992], Bayesian classifiers [Lewis and Ringuette, 1994], discriminant analysis [Hull, 1994], neural networks [Schutze *et al.*, 1995; Wiener *et al.*, 1995; Lewis *et al.*, 1996], etc. Schutze et al. and Wiener et al. made use of non-linear neural networks, but reported only a slight improvement in accuracy over the use of linear neural networks. The activation function used in the linear neural network of [Schutze *et al.*, 1995; Wiener *et al.*, 1995; Lewis *et al.*, 1996] is the sigmoid activation function, while our perceptron uses a stepwise (0-1) activation function. To our knowledge, none of the previous work has used the perceptron learning algorithm in text categorization. We used the perceptron algorithm since it has been shown to achieve surprisingly high accuracy [Mooney *et al.*, 1989], and it has very fast training time (at least an order of magnitude faster compared with the backpropagation algorithm of non-linear neural networks), making it a good choice for building a practical text categorization system. However, we do not claim that the perceptron algorithm is the best learning algorithm to use for text categorization. More experimentation needs to be done to evaluate the relative strength of various learning algorithms.

Our new correlation coefficient is based on a variation of the χ^2 metric used in [Schutze *et al.*, 1995]. To our knowledge, none of the previous work has adopted the use of such a correlation coefficient for feature selection in text categorization. It appears that the improvement resulting from the use of better feature selection methods is at least as significant as the improvement achieved from better learning algorithms.

Finally, no previous work has reported the comparative accuracy of a semi-automated learning approach with the manual, rule-based expert system approach.

8 Conclusion

We have successfully built a robust, efficient and practical text categorization system, CLASSI, using the perceptron learning algorithm. Our evaluation has shown that CLASSI outperforms existing approaches on the standard Reuters corpus. The use of a new correlation coefficient in feature selection results in considerable improvement in categorization performance. We also conducted a case study which indicates that a semi-automated approach can achieve categorization performance close to the manual, expert system approach of building text categorization systems.

References

[Apte *et al.*, 1994] Chidanand Apte, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, July 1994.

[Cohen and Singer, 1996] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In *19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.

[Hayes *et al.*, 1990] P.J. Hayes, P.M. Andersen, I.B. Nirenburg, and L.M. Schmandt. TCS: A shell for content-based text categorization. In *Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications*, pages 320–326, 1990.

[Hearst *et al.*, 1996] Marti Hearst, Jan Pedersen, Peter Pirolli, Hinrich Schutze, Gregory Grefenstette, and David Hull. Xerox TREC4 site report. In *Proceedings of the Fourth Text Retrieval Conference TREC-4*, 1996.

[Hull, 1994] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *17th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[Kohavi and John, 1995] Ron Kohavi and George H. John. Automatic parameter selection by minimizing estimated error. In *Machine Learning: Proceedings of the Twelfth International Conference*, 1995.

[Lewis and Ringuette, 1994] David Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Symposium on Document Analysis and Information Retrieval*, 1994.

[Lewis *et al.*, 1996] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.

[Lewis, 1992] David Lewis. *Representation and Learning in Information Retrieval*. PhD thesis, Dept of Computer and Information Science, Univ of Massachusetts at Amherst, 1992.

[Masand *et al.*, 1992] Brij Masand, Gordon Linoff, and David Waltz. Classifying news stories using memory based reasoning. In *15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992.

[Miller, 1990] George A Miller. Five papers on WordNet. *International Journal of Lexicology*, 3(4), 1990.

[Mooney *et al.*, 1989] Raymond J. Mooney, Jude W. Shavlik, G. Towell, and A. Gove. An experiemental comparison of symbolic and connectionist learning algorithms. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 775–780, 1989.

[Rijsbergen, 1979] C. J. Van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[Rocchio, 1971] J. Rocchio. Relevance feedback information retrieval. In Gerard Salton, editor, *The Smart Retrieval System – Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[Rosenblatt, 1958] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[Schutze *et al.*, 1995] Hinrich Schutze, David A. Hull, and Jan O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *18th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.

[Wiener *et al.*, 1995] Erik Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Symposium on Document Analysis and Information Retrieval*, 1995.