

# Using One-Class SVMs for Relevant Sentence Extraction

Canasai Kruengkrai and Chuleerat Jaruskulchai  
Intelligent Information Retrieval and Database Laboratory  
Department of Computer Science, Faculty of Science  
Kasetsart University, Bangkok, Thailand  
Email: g4364115, fscichj@ku.ac.th

## Abstract

*In this paper, we present a novel approach to relevant sentence extraction, especially using only positive examples for training. Our approach applies a methodology of Support Vector Machines (SVMs) for one-class classification called one-class SVMs. The idea is to transform the data into the feature space corresponding to the kernel, and then separate them from the origin with maximum margin. We also examine a method for analyzing on a subset of features that appears to be the best discriminate indicators. Experiments on two different text corpora, including newspaper articles and technical papers, show that our approach gives reasonable results.*

## 1. Introduction

Relevant sentence extraction can be of great value for automatic text summarization. Traditionally, text summarization is the problem of condensing a source text into a shorter version preserving its information content. It can be broadly classified into two approaches: abstraction and extraction. In contrast to abstraction that requires using heavy machinery from natural language processing (NLP), including grammars and lexicons for parsing and generation [7], extraction can be easily viewed as the process of selecting relevant excerpts (e.g., sentences, paragraphs, etc.) from the original document and concatenating them into a shorter form. Thus, most of recent works in this area are based on extraction [5]. A wide variety of research has considered the use of statistical learning to extract text for summarization [12][4]. These approaches learn by combining statistics from a training data.

In a supervised learning task, we can address text summarization in the simplest form of classification problem that determines whether a given sentence is a member of relevant or non-relevant sentences. This task needs a number of labeled training examples to learn accurately. One can identify the positive examples (relevant sentences) that can be manually produced by humans or automatically generated by a sentence alignment method [2], but it would be hard to identify representative negative examples (non-relevant sentences). Unfortunately, using the remaining sentences that are not relevant sentences as the negative examples can cause the unbalanced training data problem on some data sets. Work by [14] employs a simple strategy by selecting a random subset of negative examples to form the balanced training data set. However, randomly selecting the negative examples may hurt classifier performance if the representative negative examples are removed. More sophisticated

approaches for this problem are suggested in [11].

In this paper, we present a novel approach in order to extract relevant sentences, especially using only positive examples for training. We also examine a method for analyzing on a subset of features that appears to be the best discriminate indicators. Our approach applies a methodology of Support Vector Machines (SVMs) for one-class classification [17]. We refer to this approach as one-class SVMs. The idea of one-class SVMs is to transform the data into the feature space corresponding to the kernel, and then separate them from the origin with maximum margin. Recently, this idea have been successfully applied to other tasks, such as text categorization [13] and gene prediction [10]. An earlier approach to sentence extraction with SVMs was proposed by Hirao et al. [8], which is quite different from our work. Whereas we consider sentence extraction as one-class classification that uses only positive examples for training, they consider the same problem as two-class classification that trains a classifier by using both positive and negative examples. They conducted experiments with Japanese text documents, which is extremely difficult to compare with our approach.

The remainder of the paper is organized as follows. Section 2 reviews some important concepts of one-class SVMs. In Section 3, we describe how to represent sentences in the form feature vectors, and our feature combination strategy are presented. Section 4 provides details of our experiments, including data sets and evaluation metrics. In Section 5, we discuss experimental results. Finally, conclusion and future work are given in Section 6.

## 2. One-Class Support Vector Machines

In this section, we describe some important concepts of one-class SVMs. Based on the structural risk minimization principle from statistical theory [22], SVMs have been applied to a wide range of real-world tasks. The formulation of SVMs can be considered as a simple linear classifier, normally using both negative and positive examples for training. In this situation, they separate a given set of binary training data with a maximal margin hyperplane. They can perform with nonlinear separation by using the technique of kernels, which realizes a nonlinear mapping to a feature space. Schölkopf et al. [17] have extended SVMs to one-class classification problems. Their approach is to construct a hyperplane that is maximally distant from the origin with all data lying on the opposite side from the origin. Another approach to one-class classification problem has also been proposed by Tax and Duin [18]. Their approach is called

the support vector data description (SVDD). The idea is to find a hypersphere to enclose a data set in feature space with minimum volume. Here we limit ourselves to Schölkopf et al.’s approach. We now give details of the algorithm for training one-class SVMs.

Given training data  $\mathbf{x}_1, \dots, \mathbf{x}_l \in \mathfrak{R}^N$ , where  $\mathbf{x}_i$  is a feature vector, we would like to estimate a function that takes the value +1 in a “small” region capturing most of the data points, and -1 elsewhere [13]. Formally, we write the function as follows:

$$f(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{x} \in S, \\ -1 & \text{if } \mathbf{x} \in \bar{S}, \end{cases} \quad (1)$$

where  $S$  and  $\bar{S}$  are a simple subset of the input space and its complement, respectively. Let  $\Phi: \mathfrak{R}^N \rightarrow F$  be a nonlinear mapping that maps the training data from  $\mathfrak{R}^N$  to a feature space  $F$ . To separate the data set from the origin, we solve the following primal optimization problem [17]:

$$\text{minimize: } \mathbf{V}(\mathbf{w}, \xi, \rho) = \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{\nu l} \sum_{i=1}^l \xi_i - \rho \quad (2)$$

$$\text{subject to: } (\mathbf{w} \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad (3)$$

where  $\nu \in (0, 1)$  is a parameter for controlling the tradeoff between the number of outliers and the model complexity, and  $\rho$  is the margin. When we solve the problem, we can obtain  $\mathbf{w}$  and  $\rho$ . Given a new data points  $\mathbf{x}$  to classify, a label is assigned according to the decision function that can be expressed as:

$$f(\mathbf{x}) = \text{sgn}((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) - \rho). \quad (4)$$

Introducing Lagrange multipliers  $\alpha_i$  and using the Kuhn-Tucker condition, we set the derivatives with respect to the primal variables equal to zero, and then we can get:

$$\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i), \quad (5)$$

where only a subset of points  $\mathbf{x}_i$  that lies closest to the hyperplane has nonzero values  $\alpha_i$ . These points are called support vectors. Instead of solving the primal optimization problem directly, one can consider the following dual program:

$$\text{maximize: } \mathbf{W}(\alpha) = \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (6)$$

$$\text{subject to: } 0 \leq \alpha_i \leq \frac{1}{\nu l}, \quad \sum_i \alpha_i = 1. \quad (7)$$

$K(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j))$  are the kernels (the dot products between mapped pairs of input points). They allow much more general decision functions when the data are not linearly separable, and the hyperplane can be represented in a feature space. The kernels frequently used are polynomial kernels  $K(\mathbf{x}_i, \mathbf{x}_j) = ((\mathbf{x}_i \cdot \mathbf{x}_j) + 1)^d$  and Gaussian kernels  $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)$ . From equation (4) and (5), we can eventually write the decision function as:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) - \rho\right). \quad (8)$$

### 3. Sentence Representation

#### 3.1 Feature Vector

As we consider sentence extraction as a classification problem, we must decide how to represent an individual sentence

in terms of a feature vector. The goal is to find features that will help discriminate between relevant sentences and non-relevant sentences as much as possible. We use the  $N$ -dimensional binary vector,  $\mathbf{x} = (x_1, \dots, x_N)$ , for representing a specific sentence. In our current work, we employ seven common features. These features are outlined as follows.

**Absolute Location:** Teufel [19] notes that the structure of an article has the absolute spatial organization of information. To extract this feature, the document is evenly partitioned into 20 segments. Then these segments are collapsed, yielding in 10 differently sized segments. The segmentation technique tries to follow the structure of ideal documents. As a result, the segment size is smaller in the beginning and end of the document, while the segment size is larger in the middle. Each feature  $x_i$  for  $i = 1, \dots, 10$  equals 1 if the sentence occurs in the corresponding segments, respectively.

**Section Structure:** Sentences in a section (e.g., Introduction or Conclusion) are distinguished according to their occurrences. Each feature  $x_i$  for  $i = 11, 12$ , or 13 equals 1 if the sentence occurs in first, middle or last third of section, respectively.

**Paragraph Structure:** Similar to the section structure, each feature  $x_i$  for  $i = 14, 15$ , or 16 equals 1 if the sentence occurs in first, middle or last third of paragraph, respectively.

**Cue Phrases:** This feature is depend on the document genre, which can be extracted from a training corpus. The technique for gathering the cue phrases was discussed in [20]. In our work, we only used the fixed cue phrases that occurred in gold standard sentences of the training data. On the corpus of technical papers, the cmp-lg, we found a number of frequently occurred phrases in the training data, which were extracted as the cue phrases. For examples, these phrases are “In this paper”, “The paper presents”, “In summary”, etc. However, on the corpus of newspaper articles, the Ziff-Davis, we did not found useful phrases. Consequently, this feature did not be used on the Ziff-Davis data set. The feature  $x_{17}$  gets score 1 if the fixed cue phrase occurs in the sentence, and otherwise 0.

**TFIDF:** In information retrieval, TFIDF (Term Frequency Inverse Document Frequency) is a well-known term weight technique measuring [16]. The TFIDF weight can be calculated as follows:

$$w_{i,k} = tf_{i,k} \cdot idf_k = tf_{i,k} \cdot \log(D/n_k), \quad (9)$$

where  $tf_{i,k}$  is the frequency of occurrence of term  $k$  in document  $i$ ,  $n_k$  is the total number of documents with term  $k$  assigned, and  $D$  is the total number of documents in the corpus. We selected top 10 scored terms in our experiments. Given a set of ranking sentences by TFIDF weighting, the feature  $x_{18}$  of top 40 sentences gets score 1, and otherwise 0.

**Title Terms:** Terms occurring in the title usually reveal the specific concept of an article. After removing stop words, each sentence is assigned the score according to the frequency of the title term occurrences. The feature  $x_{19}$  of top 15 sentences receives score 1, and otherwise 0.

**Sentence Length:** Given a threshold (20 tokens including punctuation), The feature  $x_{20}$  gets score 1 for all sentences longer than the threshold, and otherwise 0.

Using all of these features for representing each sentence, we finally obtain 20-dimensional binary vector, and the classifier can be performed.

Property	Ziff-Davis	cmp-lg
number of docs	6942	179
average length of docs (word)	970	3947.8
average length of abstracts (word)	123	120.1
average length of extracts (word)	283	196.4
compression rate (abstract/doc)	17.86%	4%
compression rate (extract/doc)	34.70%	6%

Table 1: Properties of data sets.

### 3.2 Feature Combination

In [1], a batch feature combination technique was used to identify potential contributing features. They experimented extensively to obtain the best feature combination. The process of finding the optimal feature combination can be very expensive. For this reason, our technique here is based on Recursive Feature Elimination (RFE) for feature selection with SVMs. More details of RFE can be found in [6]. The idea of RFE is to train the classifier, compute the ranking criterion for all features, and remove the feature with smallest ranking criterion. By iterating this procedure, we generate nested subsets of features that provide the best possible separation.

For the case of linear SVMs, the ranking criterion of the features can be estimated from the weight  $\mathbf{w}$  multiplying the inputs of a given classifier. The RFE method can also be extended to the nonlinear case. Given a data set consisting of features  $\{x_1, \dots, x_k, \dots, x_N\}$ , we start by computing its objective function. Defining the matrix  $\mathbf{H}$  as  $(\mathbf{H})_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ , equation (6) can be written as  $\mathbf{W}(\alpha) = (1/2)\alpha^T \mathbf{H} \alpha$ . We first find the change in the objective function  $\mathbf{H}(-k)$  caused by removing each feature  $x_k$ . We then calculate its ranking criterion by the following equation:

$$\Psi(x_k) = \frac{1}{2}(\alpha^T \mathbf{H} \alpha - \alpha^T \mathbf{H}(-k) \alpha). \quad (10)$$

The feature corresponding to the smallest difference  $\Psi(x_k)$  will be removed. The RFE procedure can be processed to carry out ranking criterion by keeping the  $\alpha$  unchanged.

## 4. Experiments

In this section, we describe in detail two different text corpora: the Ziff-Davis and the cmp-lg data sets. The typical approach for testing a summarization system is to create an “ideal” summary, either by professional abstractors or merging summaries provided by multiple human subjects using methods such as majority opinion, union, or intersection [9]. This approach is known as intrinsic method. However, identifying important units in documents by humans is a time-consuming process. There have been several works presenting algorithms for automatic producing gold standard sentences [2][15]. These gold standard sentences called extracts can be used for training and evaluating the summarization systems.

### 4.1 Data Sets and Experimental Setup

The Ziff-Davis data set is a collection of 6942 newspaper articles announcing computer products, marking important units in each document by Marcu [15]. We used an individual sentence that contains the important unit as an extract. The annotation scheme is based on aligning clauses in a hand-written abstract with clauses in that text. To

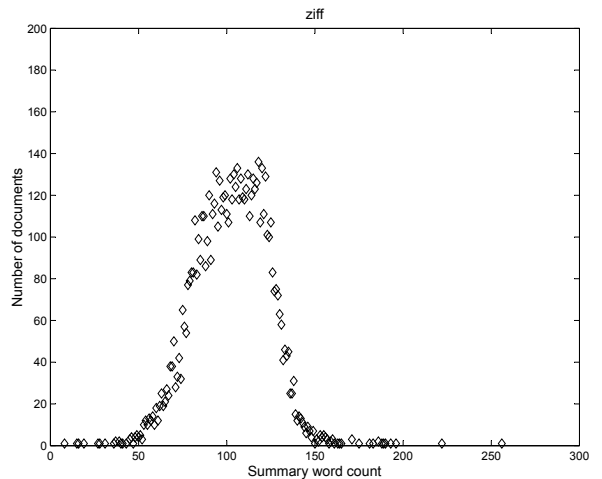


Figure 1: Distribution of abstract word length on the Ziff-Davis data set.

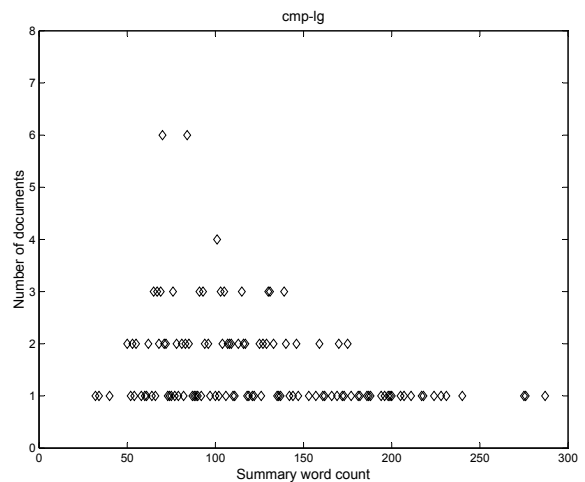


Figure 2: Distribution of abstract word length on the cmp-lg data set.

determine these clauses, the rhetorical structure theory is applied. The idea is to delete the clauses from the text and also preserve the correlation between the abstract and the text based on cosine similarity measure. This process converges when it can no longer delete the clauses without decreasing the similarity of the current set of extracts with the abstract. It finally cleans up the generated extracts using the rhetorical status of the clauses.

The cmp-lg data set is a collection of 183 papers from computational linguistics with abstracts written by the authors [21]. The paper which has no abstract was removed resulting 179 papers consisting of abstract-text pairs. We also removed the reference (or bibliography) section of the papers, and replaced mathematical equations and headers by placeholders. In the cmp-lg data set, we produced extracts from hand-written abstracts by using Banko et al.’s algorithm [2]. The algorithm first finds all the open class words, which is the strongly indicative words. Then it tags

all the proper nouns and common nouns in the text, and measures the overlap score between a sentence in the abstract and sentences in the text. The overlap score gains the extra points if a matched token is a proper noun or a common noun, which is capable of distinguishing between two sentences much more than a normal token. Finally, the algorithm selects a sentence from the text that has maximum overlap score. These steps are iterated until all the sentences in the abstract are read.

Table 1 shows properties of the data sets. The average abstract lengths on the data sets are nearly the same number, while the average document lengths are quite different, which leads to the smaller compression rates in the longer documents. This trend was also observed in [5], suggesting that the summary (or abstract) length is independent of the document length. Figure 1 and 2 illustrate the distributions of the abstract length on the data sets. It can be seen that the abstract length is narrowly distributed around 100-130 words on the Ziff-Davis data set, but it is broadly distributed around 70-160 words on the cmp-lg data set. The average ratio between the extract length and the abstract length is roughly 2.3 for the Ziff-Davis data set, and 1.6 for the cmp-lg data set.

Labeling the data sets was straightforward. We transformed the entire sentences in the text corpus into feature vectors. The feature vectors corresponding to extracts were labeled as positive examples, and the remaining vectors in that document were labeled as negative examples. On the Ziff-Davis data set, this yielded 272784 feature vectors. After removing duplicate vectors, the training data contained 1415 unique vectors. We were finally left with 698 positive examples and 717 negative examples. On the cmp-lg data set, we obtained 17254 feature vectors. After removing duplicate vectors, the training data contained 1435 unique vectors. We were finally left with 368 positive examples and 1067 negative examples. It can be seen that on some data sets, like the cmp-lg, the case of unbalanced training data occurs. One significant benefit of using one-class SVMs for our task is that it merely uses the positive examples for training, so the redundancy of the negative examples does not affect learning process at all. However, the case of unbalanced data is still the challenge in classification, since the classifier has to tackle the noise in negative examples. The experiments presented in this paper were performed using LIBSVM [3], which can solve the one-class SVM algorithm based on Schölkopf et al.’s approach.

## 4.2 Evaluation Metrics

We evaluate our algorithm using a ten-fold cross-validation methodology. We conduct ten runs in which we train the classifier using 9/10 of our data, and test the classifier using the remaining data. During testing, a test example always falls into one of the following 4 categories: false positive (*FP*) if the classifier labels it as a positive while it is a negative; false negative (*FN*) if the classifier labels it as a negative while it is a positive; true positive (*TP*) and true negative (*TN*) if the classifier correctly predicts the label. We then refer to recall (*R*) of the classifier as the fraction between the number of true positive predictions *TP* and the number of positive examples in the test set:

$$R = \frac{TP}{TP + FN} , \quad (11)$$

precision (*P*) as the fraction between the number of true positive predictions *TP* and the number of positively identified examples during testing:

$$P = \frac{TP}{TP + FP} , \quad (12)$$

In order to get a single measure of effectiveness, we employ  $F_1$  that is a combination of recall and precision calculated as follows:

$$F_1 = \frac{2 \cdot R \cdot P}{R + P} . \quad (13)$$

## 5. Results and Discussion

We first consider the effect of training with different kernels. Experiments were run using all the features as described in Section 3.1. One each cross-validation run, we used default parameters  $\sigma$  and  $d$  of LIBSVM for the Gaussian and the polynomial kernels, respectively. We varied the parameter  $\nu$  in the range [0.05, 0.50]. All results presented are the averages of ten runs using the micro-average. Table 2 and 3 show results of using different kernels on two data sets. Notice that for all four types of kernels (linear, polynomial, Gaussian, and sigmoid), the one-class SVM algorithm performs well with the small values of  $\nu$ . Since the smaller values of  $\nu$  correspond to the smaller number of outliers, this leads to the larger region capturing most of the training points. It can be seen that the results are very sensitive to the parameters. On the Ziff-Davis data set, the one-class SVM algorithm works quite well with the linear and the sigmoid kernels. However, it yields poor results for the Gaussian kernel. Further experiments are needed for tuning the parameter  $\sigma$  of the Gaussian kernel on this data set. On the cmp-lg data set, the one-class SVM algorithm gives the best results with the Gaussian kernel. The results are slightly different for other kernels.

We now move on to different feature combinations. In our experiments, we employed a greedy sub-optimal RFE method that generates nested subsets of features as described in Section 3.2. Results were carried out using the Gaussian kernel and  $\nu = 0.10, 0.20$ . On each iteration of RFE runs, we used a nested subset of features as a feature combination. Let  $\mathbb{F}$  be the set of the entire features. We denote by  $\mathbb{F} \setminus \{.\}$  a feature which is removed from  $\mathbb{F}$  based on its ranking criterion. On both data sets, the TFIDF feature was eliminated in the first iteration, since it yielded the smallest difference of the objective function. After the second and the third iteration, the title term (TT) and cue phrase (CP) features were removed on the cmp-lg data set, respectively. For the Ziff-Davis data set, the title term feature was also removed in the second iteration. We then stopped the RFE procedure, because we did not use the cue phrase feature on this data set. This yields four different feature combinations. Table 4 shows results of using different feature combinations. On the Ziff-Davis data set, classifier performance decreases when we use other feature combinations. Using all the features ( $\mathbb{F}$ ) provides the best results. However, on the cmp-lg data set, the one-class SVM algorithm excluding TFIDF from the features can slightly increase classifier performance.

To compare the one-class SVM algorithm with other supervised learning algorithm, we chose a method in [12] as the baseline method. This method is based on the independent assumption of Bayes’ Rule. First, it determines the char-

$\nu$	Linear			Polynomial			Gaussian			Sigmoid		
	$P$	$R$	$F_1$	$P$	$R$	$F_1$	$P$	$R$	$F_1$	$P$	$R$	$F_1$
0.05	0.489	0.926	0.640	0.484	0.704	0.573	0.452	0.439	0.445	0.497	0.962	0.655
0.10	0.491	0.901	0.635	0.474	0.829	0.603	0.457	0.416	0.436	0.507	0.943	0.659
0.20	0.491	0.714	0.582	0.459	0.586	0.514	0.412	0.355	0.381	0.503	0.835	0.627
0.30	0.488	0.699	0.575	0.453	0.567	0.503	0.373	0.309	0.338	0.505	0.765	0.609
0.40	0.493	0.670	0.568	0.440	0.513	0.474	0.340	0.241	0.282	0.515	0.646	0.573
0.50	0.496	0.509	0.502	0.500	0.504	0.502	0.312	0.213	0.253	0.505	0.510	0.508

**Table 2: Results of the one-class SVM algorithm on the Ziff-Davis data set.**

$\nu$	Linear			Polynomial			Gaussian			Sigmoid		
	$P$	$R$	$F_1$	$P$	$R$	$F_1$	$P$	$R$	$F_1$	$P$	$R$	$F_1$
0.05	0.277	0.937	0.427	0.312	0.801	0.449	0.351	0.686	0.464	0.275	0.937	0.425
0.10	0.308	0.872	0.455	0.317	0.773	0.450	0.354	0.691	0.468	0.309	0.874	0.456
0.20	0.331	0.781	0.465	0.333	0.730	0.457	0.377	0.689	0.487	0.329	0.781	0.464
0.30	0.337	0.686	0.452	0.331	0.628	0.434	0.391	0.637	0.484	0.340	0.686	0.455
0.40	0.351	0.596	0.442	0.318	0.462	0.376	0.387	0.642	0.483	0.352	0.596	0.443
0.50	0.342	0.486	0.402	0.308	0.391	0.345	0.408	0.443	0.425	0.346	0.500	0.409

**Table 3: Results of the one-class SVM algorithm on the cmp-lg data set.**

acteristic properties of features from a text corpus. Then, each sentence in a test document obtains scores for each of the features used to estimate the sentence’s probability to be included in the summary given its feature values. With this approach, it affects the evaluation that the number of sentences selected from a test document equals to the number of extracts. Thus, precision, recall, and  $F_1$  are always identical.

For one-class SVMs, we have to choose a subset of predicted examples in order to compare results with the naive Bayes. Consequently, the predicted examples must be ranked according to their distances from the hyperplane. We can test each of the predicted examples to get its signed distance from the optimal hyperplane, which can be computed as a by-product of evaluating candidates for including in the classes (the argument of the  $sgn$  in the decision function  $f(\mathbf{x}_i)$ ). We just rank them in decreasing order and select top  $r$  examples, where  $r$  is the number of the extracts in the test set. Table 5 presents the performance of the classifiers obtained by using two feature combinations ( $\mathbb{F}$  and  $\mathbb{F} \setminus \{\text{TFIDF}\}$ ), and fixing  $\nu = 0.20$ . We see that the performance of the one-class SVM algorithm slightly drops on this evaluation metric. On the Ziff-Davis data set, the one-class SVM algorithm outperforms the Naive Bayes in all cases. On the cmp-lg data set, it yields much better results using only  $\mathbb{F} \setminus \{\text{TFIDF}\}$ .

## 6. Conclusion and Future Work

This paper has examined the one-class classification based on SVMs for relevant sentence extraction. Experimental results indicate that the use of one-class SVM algorithm has distinct advantages, especially in the case of the unbalanced training data set. We show that the one-class SVM algorithm with appropriate parameters can give reasonable performance. We also apply the RFE method to the feature combination problem. Our results demonstrate that it is possible to improve the performance of the classifiers on by combining a set of features. We believe that our approach

is capable of performing on real-world data. In future work, we are interested to explore other features, such as parts of speech and name entities.

## Acknowledgments

This research was supported by the grant of the National Research Council of Thailand, 2002. We thank Daniel Marcu for providing us the Ziff-Davis data set with marking important units in all documents.

## 7. References

- [1] C. Aone, M. Okurowski, J. Gorfinsky, and B. Larsen. A trainable summarizer with knowledge acquired from robust NLP techniques. In I. Mani, M. Maybury (Eds.), *Advances in automatic text summarization*. MIT Press, 1999.
- [2] M. Banko, V. Mittal, M. Kantrowitz, and J. Goldstein. Generating extraction-based summaries from hand-written summaries by aligning text spans. In *Proceedings of PACLING '99*, 1999.
- [3] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. 2002. The software is available at: <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [4] W. T. Chuang and J. Yang. Extracting sentence segments for text summarization: A machine learning approach. In *Proceedings of the 23rd ACM SIGIR*, pages 152–159, 2000.
- [5] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell. Summarizing text documents: Sentence selection and evaluation metrics. In *Proceedings of the 22nd ACM SIGIR*, pages 121–128, 1999.
- [6] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1/3):389–422, 2000.

Feature	ziff						cmp-lg					
	$\nu = 0.1$			$\nu = 0.2$			$\nu = 0.1$			$\nu = 0.2$		
	$P$	$R$	$F_1$	$P$	$R$	$F_1$	$P$	$R$	$F_1$	$P$	$R$	$F_1$
$\mathbb{F}$	0.457	0.416	0.436	0.412	0.355	0.381	0.354	0.691	0.468	0.377	0.689	0.487
$\mathbb{F} \setminus \{\text{TFIDF}\}$	0.388	0.297	0.337	0.365	0.274	0.313	0.393	0.652	0.490	0.414	0.633	0.501
$\mathbb{F} \setminus \{\text{TFIDF}, \text{TT}\}$	0.308	0.218	0.255	0.270	0.141	0.185	0.414	0.540	0.469	0.417	0.523	0.464
$\mathbb{F} \setminus \{\text{TFIDF}, \text{TT}, \text{CP}\}$	-	-	-	-	-	-	0.392	0.408	0.400	0.394	0.417	0.405

**Table 4: Results of the one-class SVM algorithm using different feature combinations with the Gaussian kernel.**

Data Set	Feature	Naive Bayes	One-Class SVM			
			Linear	Polynomial	Gaussian	Sigmoid
ziff	$\mathbb{F}$	0.397	0.509	0.486	0.403	0.500
	$\mathbb{F} \setminus \{\text{TFIDF}\}$	0.346	0.469	0.523	0.386	0.457
cmp-lg	$\mathbb{F}$	0.379	0.374	0.380	0.421	0.374
	$\mathbb{F} \setminus \{\text{TFIDF}\}$	0.388	0.412	0.405	0.468	0.409

**Table 5: Comparisons of the one-class SVM algorithm and the Naive Bayes.**

- [7] U. Hahn and I. Mani. The challenges of automatic summarization. *IEEE Computer*, 33(11):29–35, 2000.
- [8] T. Hirao, H. Isozaki, E. Maeda, and Y. Matsumoto. Extracting important sentences with support vector machines. In *Proceedings of COLING 2002*, pages 342–348, 2002.
- [9] H. Jing, R. Barzilay, K. McKeown, and M. Elhadad. Summarization evaluation methods: Experiments and analysis. In *AAAI Intelligent Text Summarization Workshop*, pages 60–68, 1998.
- [10] A. Kowalczyk and B. Raskutti. One class svm for yeast regulation prediction. In *SIGKDD Explorations*, volume 4, pages 99–100, 2002.
- [11] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *Proc. 14th International Conference on Machine Learning*, pages 179–186, 1997.
- [12] J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proceedings of the 18th ACM SIGIR*, pages 68–73, 1995.
- [13] L. M. Manevitz and M. Yousef. One-class svms for document classification. *Journal of Machine Learning* 2, pages 139–154, 2001.
- [14] I. Mani and E. Bloedorn. Machine learning of generic and user-focused summarization. In *Proceedings of the Fifteenth National Conference on AI*, pages 821–826, 1998.
- [15] D. Marcu. The automatic construction of large-scale corpora for summarization research. In *Proceedings of the 22nd ACM SIGIR*, pages 137–144, 1999.
- [16] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
- [17] B. Schölkopf, J. C. Platt, J. ShaweTaylor, A. J. Smola, and R. C. Williamson. Estimating the support of a highdimensional distribution. Technical report, Microsoft Research, MSRTR9987, 1999.
- [18] D. M. Tax and R. P. Duin. Outliers and data descriptions. In *Proceedings of the Seventh Annual Conference of the Advanced School for Computing and Imaging*, 2001.
- [19] S. Teufel. rgumentative zoning: Information extraction from scientific articles. Ph.D. Thesis, University of Edinburgh, 1999.
- [20] S. Teufel and M. Moens. Sentence extraction as a classification task. In *ACL/EACL-97 Workshop on Intelligent and Scalable Text Summarization*, 1997.
- [21] TIPSTER Text Summarization Evaluation Conference (SUMMAC). May 1998. The data set is available at: [http://www.itl.nist.gov/iaui/894.02/related\\_projects/tipster\\_summac/cmp\\_lg.html](http://www.itl.nist.gov/iaui/894.02/related_projects/tipster_summac/cmp_lg.html).
- [22] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, United Kingdom, 1998.