Probabilistic Sentence Reduction Using Support Vector Machines

M.L. Nguyen, A. Shimazu, S. Horiguchi, B.T. Ho and M. Fukushi

Japan Advanced Institute of Science and Technology
1-8, Tatsunokuchi, Ishikawa, 923-1211, JAPAN
nguyenml@jaist.ac.jp

Abstract

This paper investigates a novel application of support vector machines (SVMs) in sentence reduction. Furthermore, we propose a new probabilistic sentence reduction method based on support vector machine learning. Experimental results show that the proposed methods outperform earlier methods in term of sentence reduction performance.

1 Introduction

The most popular methods of sentence reduction for text summarization are corpus based methods. Jing (Jing 00) studied a method to remove extraneous phrases from sentences by using multiple sources of knowledge to decide which phrases could be removed. However, while this method exploits a simple model for sentence reduction by using statistics computed from a corpus, a better model can be obtained by using a learning approach.

Knight and Marcu (Knight 02) proposed a sentence reduction based on corpus using machine learning technique. They discussed a noisy-channel based approach and a decision tree based approach to sentence reduction. Their algorithms provide the best way to scale up the full problem of sentence reduction using available data. However, these algorithms involve the constraint that the word order of a given sentence and its reduced sentence are the same. Nguyen and Horiguchi (Nguyen 03a) presented a new sentence reduction technique based on a decision tree model without the constraint. They also indicated that semantic information is useful for sentence reduction tasks.

The major drawback of previous works on sentence reduction is that those methods are likely to output local optimal results, which may have lower accuracy. This problem is caused by the inherent sentence reduction model; that is, only a single reduced sentence can be obtained. As pointed out by Lin (Lin 03), the best sentence reduction output for a single sentence is not approximately best for text summarization. This means that local optimal refer to the best reduced output for a single sentence and the best reduced output for the whole text is global optimal. Therefore, it is very valuable if the sentence reduction task can generate multiple reduced outputs and select the best one using the whole text document. However, such a sentence reduction method has not yet been proposed.

Support Vector Machines (Vapnik 95), on the other hand, are strong learning methods in comparing with decision tree learning and other learning methods (Sholkopf 97). The aim of this paper is to illustrate the potential of SVMs in enhancing the accuracy of sentence reduction in comparing with previous works. Accordingly, a novel deterministic method for sentence reduction using SVMs and a two-state method using pairwise coupling (Hastie 98) are described. Furthermore, to solve the problem of generating multiple best outputs, we propose a probabilistic sentence reduction model, in which a variant of probabilistic SVMs using pairwise coupling and two-stage method is discussed.

The remaining of this paper will be organized as follows: Section 2 introduces the Support Vector Machines learning. Section 3 presents the previous works on sentence reduction and our deterministic sentence reduction using SVMs. We also discuss remained problems of deterministic sentence reduction. Section 4 presents a probabilistic sentence reduction using support vector machine to solve this problem. Section 5 presents implementation and experimental results; Section 6 gives our conclusions and presents some remained problems to be solved in our future work.

2 Support Vector Machines

Support vector machine (SVM)(Vapnik 95) is a technique of machine learning based on statistical learning theory. Suppose that we are given l training examples (x_i, y_i), $(1 \leq i \leq l)$, where x_i is a feature vector in n dimensional feature space, y_i is the class label {-1, +1 } of x_i. SVM finds a hyperplane $w.x + b = 0$ which correctly separates training examples and has maximum margin which is the distance between two hyperplanes $w.x + b \geq 1$ and $w.x + b \leq -1$. The optimal hyperplane with maximum margin can be obtained by slowing the following quadratic programming.

$$\begin{array}{ll} \min & \frac{1}{2}\|w\| + C_0 \sum_{i}^{l} \xi_i \\ s.t. & y_i(w.x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 & \end{array} \tag{1}$$

where C_0 is the constant and ξ_i is called a slack variable for the non-separable case. In final, the optimal hyperplane is formulated as follows:

$$f(x) = \text{sign}\left(\sum_{1}^{l} \alpha_i y_i K(x_i, x) + b\right) \tag{2}$$

where α_i is the Largrange multiple, and $K(x', x'')$ is called a kernel function, it calculates similarity between two arguments x' and x''. For instance, the Polynomial kernel function is formulated as follow:

$$K(x', x'') = (x'.x'')^p \tag{3}$$

SVMs estimate the label of an unknown example x whether sign of $f(x)$ is positive or not.

3 Deterministic Sentence Reduction Using SVMs

3.1 Problem Description

In the corpus based decision tree approach, a given input sentence is parsed into a syntax tree and the syntax tree is then transformed into a small tree to obtain a reduced sentence.

Let t and s be syntax trees of the original sentence and a reduced sentence, respectively. The process of transforming syntax tree t to small tree s is called a rewriting process (Knight 02), (Nguyen 03a). To transform the syntax tree t to the syntax tree s, some terms and five rewriting actions are defined.

An *Input list* consists of a sequence of words subsumed by the tree t where each word in the Input list is labelled with the name of all syntactic constituents in t. Let CSTACK be a stack that consists of sub trees in order to rewrite a small tree. Let RSTACK be a stack that consists of sub trees, which are removed from the Input list in the rewriting process.

- SHIFT action transfers the first word from the Input list into CSTACK. It is written mathematically and given the label SHIFT.
- REDUCE(lk, X) action pops the lk syntactic trees located at the top of CSTACK and combines them into a new tree, where lk is an integer and X is a grammar symbol.
- DROP X action moves subsequences of words that correspond to syntactic constituents from the Input list to RSTACK.
- ASSIGN TYPE X action changes the label of trees at the top of the CSTACK. These POS tags might be different from the POS tags in the original sentence.
- RESTORE X action takes the X element in RSTACK to remove that element into the Input list, where X is a subtree.

For convenience, let *configuration* be a status of Input list, CSTACK and RSTACK. Let *current context* be the important information of a configuration. The important information are defined as a vector of features using heuristic methods as in (Knight 02), (Nguyen 03a).

The main idea behind the deterministic sentence reduction is that it uses a rule in the current context of the initial configuration to select a distinct action in order to rewrite an input sentence into a reduced sentence. After that, the current context is changed to a new context and the rewriting process is repeated for selecting an action that corresponds to the new context. The rewriting process is finished when it meets a termination condition. Here, one rule corresponds to the function that maps the current context to a rewriting action. These rules are learned automatically from the corpus of long sentences and their reduced sentences (Knight 02), (Nguyen 03a).

3.2 Example

Figure 3 shows an example of applying a sequence of actions to rewrite the input sentence (a, b, c, d, e), where each character is a word. The structure of the Input list, two stacks, and the term of a rewriting process based on the actions mentioned above are illustrated in Figure 3. For example, in the first row, DROP H deletes the sub-tree with its root node H in the Input list and stored it in the RSTACK. The reduced tree s can be obtained after applying a sequence of actions as follows: DROP H; SHIFT; ASSIGN TYPE K; DROP B; SHIFT; ASSIGN TYPE H; REDUCE 2 F; RESTORE H;

SHIFT; ASSIGN TYPE D; REDUCE 2G. In this example, the reduced sentence is (b, e, a).

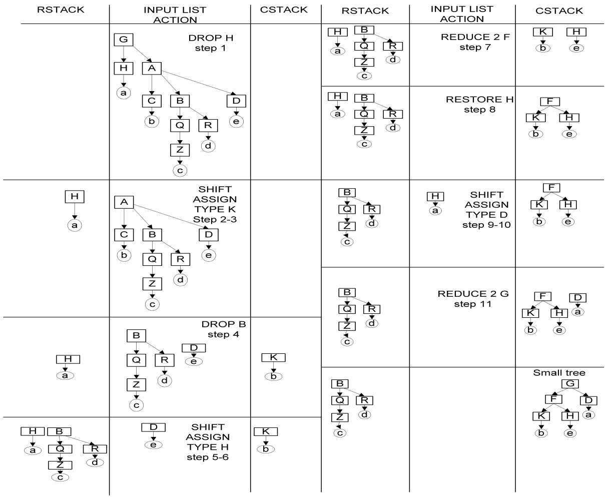

Figure 1: An Example of Rewriting Process

3.3 Learning Reduction Rules Using SVMs

As mentioned above, the action for each configuration can be guessed by using a learning rule, which map a context to an action. To obtain such rules, the configuration is represented by a vector of features with a high dimension. After that, we estimate the training examples by using several support vector machines to deal with the multiple classification problem in sentence reduction.

3.3.1 Features

One of the important task in applying SVMs to text summarization is to define features. Here, we describe features used in our sentence reduction models.

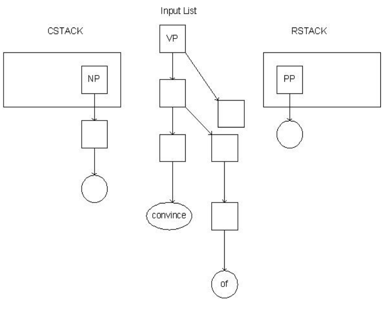

Figure 2: Example of Configuration

The features are extracted based on the current context. As it can be seen in Figure 4, a context includes the status of the Input list and the status of CSTACK and RSTACK. We define a set of features for a current context as follows:

Operation feature

The operation features are selected as follows.

- These features reflect the number of trees in CSTACK and RSTACK, and the type of last five actions.
- We also used the features represents for the information of two stacks, as the information denotes the syntactic category of the root nodes of the partial trees built up to certain time. We considered the ten last partial trees in CSTACK and RSTACK for obtaining syntactic category of their root nodes, and the POS tag of a last word in CSTACK is also considered as a feature.

Original tree feature

These features denote the syntactic constituents that start with the first unit in the Input list.
For example, in Figure 4 the syntactic constituents are labels of the current element in the Input list from "VP" to the verb "convince".

Semantic features

The followings are used in our model as semantic information.

- Semantic information about current words within the Input list; these semantic types are obtained by using the named entities such as Location, Person, Organization and Time within the input sentence. To define these name entities, we use the method described in (Borthwick 99).
- Semantic information about whether or not the word in the Input list is a *head word*.
- The word relations such as whether or not a word is in a relation with other words in the sub-categorization table. These relations and the sub-categorization table are obtained by using the Commlex database (Macleod 95).

Using the semantic information, we are able to avoid deleting important segments within the given input sentence. For instance, the *main verb*, the *subject* and the *object* are essential and for the noun phrase, the head noun is essential, but an adjective modifier of the head noun is not. For example, let us consider that the verb "convince" was extracted from the Commlex database as follows.

convince
NP-PP: PVAL ("of")
NP-TO-INF-OC

This entry indicates that the verb "convince" can be followed by a noun phrase and a prepositional phrase starting with the preposition “of”. It can be also followed by a noun phrase and a to-infinite phrase. This information shows that we cannot delete an “of” prepositional phrase or a to-infinitive that is the part of the verb phrase.

3.3.2 Two-state SVMs Learning using the Pairwise Coupling

Using these features we can extract training data for SVMs. Here, a sample in our training data consists of a pairs of a feature vectors and a action. The algorithm to extract training data from the training corpus are modified using the algorithm described in our pervious work (Nguyen 03a).

Since the original support vector machine (SVM) is a binary classification method while the sentence reduction problem is formulated as multiple classification, we have to find a method of adapting support vector machines to this problem. For multi-class SVMs, one can use the strategies such as *one-vs all*, *pairwise comparison* or *DAG graph* (Hsu 02). In this paper, we use the pairwise strategy that constructs a rule for discriminating pairs of classes and then selecting the class with the most winning among two class decisions.

To boost the training time and the reduction performance, we propose a two-stage SVM described below.

Suppose that the examples were divided into five groups $m_1, m_2, ..., m_5$., Let Svmc be multi-class SVMs and let Svmc-i be multi-class SVMs for a group m_i. We use one Svmc classifier to recognize to which group a given example e should be belong. Assume that e has been belonged to the group m_i. The classier Svmc-i is then used to recognize a specific action for the example e. The five classifiers Svmc-1, Svmc-2,..., Svmc-5 are trained by using those examples which have actions belonging to SHIFT, REDUCE, DROP, ASSIGN TYPE and RESTORE.

Table 1 shows the distribution of examples on five data groups.

3.4 Disadvantage of Deterministic Sentence Reductions

The idea of the deterministic algorithm is used the rule for each current context to select the action and so on. The process terminates when a stop condition meet. If early steps of this algorithm fail to select the best actions, then the possibility of obtaining a wrong reduced output becomes high.

Table 1: Distribution of example data on five data groups

Name	Number of examples
SHIFT-GROUP	13363
REDUCE-GROUP	11406
DROP-GROUP	4216
ASSIGN-GROUP	13363
RESTORE-GROUP	2004
TOTAL	44352

One way to solve this problem is to select multiple actions that corresponds to the context at each step in the process of rewriting. However, the question that emerges here is how to determine criteria to use in selecting multiple actions for a context. If this problem can be solved, then multiple best reduced outputs can be obtained for each input sentence and the best one will be selected by using the whole text document.

We propose a model for selecting multiple actions for a context in sentence reduction as a probabilistic sentence reduction and present a variant of probabilistic sentence reduction in the next section.

4 Probabilistic Sentence Reduction Using SVM

4.1 The Probabilistic SVMs Models

Let A be a set of k actions $A = \{a_1, a_2...a_i, ..., a_k\}$ and C be a set of n contexts $C = \{c_1, c_2...c_i, ..., c_n\}$. A probabilistic model α for sentence reduction will select an action $a \in A$ for the context c with probability $p^{\alpha}(a|c)$. The $p^{\alpha}(a|c)$ can be used to score action a among possible actions A depending the context c that is available at the time of decision. There must be several methods to estimate such scores. We called these probabilistic sentence reduction methods. The conditional probability $p^{\alpha}(a|c)$ are estimated using a variant of probabilistic support vector machine, which is described in following sections.

4.1.1 Probabilistic SVMs using the Pairwise Coupling

For convenience, we denote $u_{ij} = p(a = a_i | a = a_i \vee a_j, c)$. Given a context c and an action a, we assume that the estimated pairwise class probabilities r_{ij} of u_{ij} are available. Here r_{ij} can be estimated by some binary classifiers. For instance, we could estimate r_{ij} by using the SVM binary posterior probabilities as described in (Plat 2000). Then, the goal is to estimate $\{p_i\}_{i=1}^k$, where $p_i = p(a = a_i|c)$, $i = 1, 2, ..., k$. For this propose, a simple estimate of these probabilities can be derived by using the following voting method:

$$p_i = 2 \sum_{j:j \neq i} I_{\{r_{ij} > r_{ji}\}} / k(k-1)$$

where I is an indicate function and $k(k-1)$ is the number of pairwise classes. However, this model is too simple and we can obtain a better model by using the following method.

Assume that u_{ij} are pairwise probabilities of the model subject to the condition that $u_{ij} = p_i/(p_i+p_j)$. In (Hastie 98), the authors proposed to minimize the Kullback-Leibler (KL) distance between the r_{ij} and u_{ij}

$$l(p) = \sum_{i \neq j} n_{ij} r_{ij} log \frac{r_{ij}}{u_{ij}} \quad (4)$$

where r_{ij} and u_{ij} are the probabilities of a pairwise a_i and a_j in the estimated model and our model, respectively, and n_{ij} is the number of training data where their classes are a_i or a_j. To find the minimizer of equation (6), they first calculate

$$\frac{\partial l(p)}{\partial p_i} = \sum_{i \neq j} n_{ij} \left(-\frac{r_{ij}}{p_i} + \frac{1}{p_i + p_j}\right).$$

Thus, letting $\Delta l(p) = 0$, they proposed to find a point satisfying

$$\sum_{j:j \neq i} n_{ij} u_{ij} = \sum_{j:j \neq i} n_{ij} r_{ij}, \qquad \sum_{i=1}^{k} p_i = 1,$$

where $i = 1, 2, ...k$ and $p_i > 0$.

Such a point can be obtained by using the algorithm described elsewhere in (Hastie 98). It is applied to obtain a probabilistic SVM model for sentence reduction using a simple method as follows. Assume that our class labels belong to l groups: $M = \{m_1, m_2 ... m_i, ..., m_l\}$, where l is a number of groups and m_i is a group e.g., SHIFT, REDUCE ,..., ASSIGN TYPE. Then the probability $p(a|c)$ of an action a for a given context c can be estimated as follows.

$$p(a|c) = p(m_i|c) \times p(a|c, m_i) \quad (5)$$

where m_i is a group and $a \in m_i$. Here, $p(m_i|c)$ and $p(a|c, m_i)$ are estimated by the method in (Hastie 98).

4.2 Probabilistic sentence reduction algorithm

After obtained a probabilistic model p, we then use this model to define function score, by which the search procedure ranks derivation of incomplete and complete reduced sentences. Let $d(s) = \{a_1, a_2, ...a_d\}$ be the derivation of a small tree s, where each action a_i belong to a set of possible actions. The score of s is the product of the conditional probabilities of the individual actions in its derivation.

$$Score(s) = \prod_{a_i \in d(s)} p(a_i|c_i) \quad (6)$$

where c_i is the context in which a_i was decided. The search heuristic attempts to find the best reduced tree s^* as follow:

$$s^* = \underbrace{\text{argmax}}_{s \in \text{tree}(t)} \quad Score(s) \quad (7)$$

where $tree(t)$ are all the complete reduced trees from the tree t of the given long sentence. Assume that for each configuration the actions $\{a_1, a_2, ...a_n\}$ are sorted in the decreasing order according to $p(a_i|c_i)$, in which c_i is the context of that configuration. Algorithm 1 shows a probabilistic sentence reduction using the top K-BFS search algorithm. This algorithm uses a breadth-first search which does not expand entire frontier, but instead expands at most the top K scoring incomplete configurations in the frontier and terminated it when finding M completed reduced sentences (CL is a list of reduced trees), or when all hypotheses have been exhausted. A configuration is completed if and only if the Input list is empty and there is one tree in the CSTACK. Note that, the function *get-context(h_i, j)* obtained the current context of the j^{th} configuration in h_i, where h_i is a heap at step i. The function *Insert(s,h)* ensures that the heap h is sorted according to the score of each element in h. Essentially, in implementation we can use a dictionary of contexts and actions observed from the training data in order to reduce the number of actions to explore for a current context.

5 Experiments and Discussion

We used the same corpus as described in (Knight 02), which includes 1067 pairs of sentences and their reductions. To evaluate sentence reduction algorithms, we randomly selected 32 pairs of sentences from our parallel corpus, which is refereed to as the test corpus. The training corpus of 1035 sentences extracted 44352 examples, in which each training example is corresponded to an action. The SVM tool, the LibSVM (Chang 01) is applied to train our model. The training examples were divided

Algorithm 1 A probabilistic sentence reduction algorithm

```
$CL$={Empty};
$i = 0$; $h_0$={ Initial configuration}
while $|CL| < M$ do
  if $h_i$ is empty then
    break;
  end if
  $u$ =min($|h_i|, K$)
  for $j = 1$ to $u$ do
    $c$=get-context($h_i, j$)
    Select $m$ so that $\sum_{i=1}^{m} p(a_i|c) < Q$ is maximal
    for $l$=1 to $m$ do
      parameter=get-parameter($a_l$);
      Obtain a new configuration $s$ by performing action $a_l$
      with parameter
      if Complete($s$) then
        Insert($s, CL$)
      else
        Insert($s, h_{i+1}$)
      end if
    end for
  end for
  $i = i + 1$
end while
```

into SHIFT, REDUCE, DROP, RESTORE and ASSIGN groups. To train our support vector model in each group, we used the pairwise method with the polynomial kernel function, in which the parameter p in (3) and the constant C_0 in equation (1) is 2 and 0.0001, respectively.

The algorithms (Knight 02) and (Nguyen 03a) were served as the baseline1 and the baseline2 to compare with the proposed algorithms. The deterministic sentence reduction using SVM and the probabilistic sentence reduction is named as SVM-D and SVMP, respectively. For convenience, the ten top reduced outputs using SVMP is called SVMP-10. We used the same evaluation method as described in (Knight 02) to compare the proposed methods with pervious methods. For this propose, we presented each original sentence in the test corpus to three judges who are specialist in English, together with three sentence reductions: the human generated reduction sentence, the outputs of the proposed algorithms, and the output of the baseline algorithms.

The judges were told that all outputs were generated automatically. The order of the outputs was scrambled randomly across test cases. The judges participated in two experiments. In the first, they were asked to determine on a scale from 1 to 10 how well the systems did with respect to selecting the most important words in the original sentence. In the second, they were asked to determine the grammatical criteria of reduced sentences.

Table 2 shows the results of sentence reduction using support vector machine in comparing with the baseline methods and against human reduction for English language. Table 2 shows *compression rates*, and *mean* and *standard deviation* results across all judges, for each algorithm. The results show that the length of the reduced sentence using decision trees is shorter than using SVMs, and indicate that our new methods outperform the baseline algorithms in grammatical and importance criteria.

Table 2: Experiment results with Test Corpus

Method	Comp	Gram	Impo
Baseline1	57.19%	8.60 ± 2.8	7.18 ± 1.92
Baseline2	57.15%	8.60 ± 2.1	7.42 ± 1.90
SVM-D	57.65%	8.76 ± 1.2	7.53 ± 1.53
SVMP-10	57.51%	8.80 ± 1.3	7.74 ± 1.39
Human	64.00%	9.05 ± 0.3	8.50 ± 0.80

Table 2 shows that the first 10 reduced sentences yielded by the reduction process using SVMP-10 (the SVM probabilistic model) obtained highest performances. We also compared the computation time of sentence reduction using support vector machine with the pervious works. Table 3 shows that the computation time of SVM-D and SVMP-10 are slower than baseline, but this value are acceptable. We also investigated how sensitive the proposed algorithms are with respect to the training data by carrying out the same experiment on sentences of a different genre. We took the test corpus by selecting sentences in the web-site of the Benton Foundation (http://www.benton.org). The leading sentences in each news were selected as the most relevant sentences to the summary of the news. We obtained 32 leading long sentences and 32 headlines for each news above. The 32 sentences are used as a second test for our methods. We use a simple ranking criteria that the more words in the reduced sentence overlap with the words in the headline, the more

Table 3: Computation times of performing reductions on test-set. Average sentence length was 21 words.

Method	Time (sec)
Baseline	138.25
SVM-D	212.46
SVMP-10	1030.25

important sentence is. A sentence satisfying this criteria is called a *relevant candidate*.

For a given sentence, we use a simple method, namely SVMP-R to obtain a reduced sentence by selecting a *relevant candidate* among the ten top reduced outputs using SVMP-10.

Table 4 depicts the experiment results of the baseline methods, SVM-D, SVMP-R and SVMP-10, respectively. The results shows that, when applied to sentence of a different genre, the performance of SVMP-10 degrades smoothly, while the performance of the deterministic sentence reductions (the baselines and SVM deterministic) drops sharply. This indicates that the probabilistic sentence reduction using support vector machine is more stable.

Table 4 shows that the performance of SVMP-10 is also closed to the human reduction outputs and outperform the performance of previous works. In addition, the performance of SVMP-R outperforms the deterministic sentence reduction algorithms and the differences of SVMP-R's results and SVMP-10's results are small. This indicates that we can obtain reduced sentences which are relevant to the headline, while ensure the grammatical criterion and the importance criterion compared to the original sentences.

Table 4: Experiment results with Benton Corpus

Method	Comp	Gramm	Impo
Baseline1	54.14%	7.61 ± 2.10	6.74 ± 1.92
Baseline2	53.13%	7.72 ± 1.60	7.02 ± 1.90
SVM-D	56.64%	7.86 ± 1.20	7.23 ± 1.53
SVMP-R	58.31%	8.25 ± 1.30	7.54 ± 1.39
SVMP-10	57.62%	8.60 ± 1.32	7.71 ± 1.41
Human	64.00%	9.01 ± 0.25	8.40 ± 0.60

6 Conclusions

We have presented a new probabilistic sentence reduction approach that enables a long sentence to be rewritten into reduced sentences based on support vector models. Our methods achieves a better performance in comparing with earlier methods. The proposed reduction approach can generate multiple best outputs. Experimental results showed that the running time of reducing sentences is reasonable and the top of 10 reduced sentences returned by the reduction process might yield accuracies dramatically higher than previous works. We believe that a good ranking method might improve the sentence reduction performance further.

References

A. Borthwick, “A Maximum Entropy Approach to Named Entity Recognition”, Ph.D thesis, Computer Science Department, New York University (1999).

C.-C. Chang and C.-J. Lin, “LIBSVM: a library for support vector machines”, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

H. Jing, “Sentence reduction for automatic text summarization”, In Proceeding of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics NAACL-2000.

T.T. Hastie and R. Tibshirani, “Classification by pairwise coupling”, The Annals of Statistics, 26(1): pp. 451-471, 1998.

C.-W. Hsu and C.-J. Lin, “A comparison of methods for multi-class support vector machines”, IEEE Transactions on Neural Networks, 13, pp. 415-425, 2002.

K. Knight and D. Marcu, “Summarization beyond sentence extraction: A Probabilistic approach to sentence compression”, Artificial Intelligent 139: pp. 91-107, 2002.

C.Y. Lin, “Improving Summarization Performance by Sentence Compression — A Pilot Study”, Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages, pp.1-8, 2003.

C. Macleod and R. Grishman, “COMMLEX syntax Reference Manual”; Proteus Project, New York University (1995).

M.L. Nguyen and S. Horiguchi, “A new sentence reduction based on Decision tree model”, Proceedings of 17th Pacific Asia Conference on Language, Information and Computation, pp. 290-297, 2003

V. Vapnik, “The Natural of Statistical Learning Theory”, New York: Springer-Verlag, 1995.

J. Platt,“ Probabilistic outputs for support vector machines and comparison to regularized likelihood methods,” in Advances in Large Margin Classifiers, Cambridege, MA: MIT Press, 2000.

B. Scholkopf et al, “Comparing Support Vector Machines with Gausian Kernels to Radius Basis Function Classifers”, IEEE Trans. Signal Procesing, 45, pp. 2758-2765, 1997.