A Statistical Algorithm for Linguistic Steganography Detection Based on Distribution of Words

CHEN Zhi-li, HUANG Liu-sheng, YU Zhen-shan, LI Ling-jun, YANG wei
Department of Computer Science and Technology, University of Science and Technology of China, National High Performance Computing Center(Hefei), Hefei, Anhui 230027, China
zlchen3@mail.ustc.edu.cn

Abstract

In this paper, a novel statistical algorithm for linguistic steganography detection, which takes advantage of distribution of words in the text segment detected, is presented. Linguistic steganography is the art of using written natural language to hide the very presence of secret messages. Using the text data, which is the foundational media in internet communications, as its carrier, linguistic steganography plays an important part in Information Hiding (IH) area. The previous work was mainly focused on linguistic steganography and there were few researches on linguistic steganalisys. We attempt to do something to help to fix this gap. In our experiment of detecting the three different linguistic steganography methods: NICETEXT, TEXTO and Markov-Chain-Based, the total accuracies on discovering stego-text segments and normal text segments are found to be 87.39%, 95.51%, 98.50%, 99.15% and 99.57% respectively when the segment size is 5kB, 10kB, 20kB, 30kB and 40kB. Our research shows that the linguistic steganalysis based on distribution of words is promising.

1. Introduction

Nowadays, Internet information plays a central role in people's lives and text-based information dissemination media, such as email, blogs and text messaging is rising. As a result, the importance and size of text data is increasing at an accelerating pace. This increase in the significance of digital text in turn creates increased concerns about the usage of text media as a covert channel of communication. One of such covert means of communication is known as linguistic steganography. Linguistic steganography aims to use written natural language to conceal secret messages. The whole idea is to hide the very existence of the real message. Linguistic steganography algorithms embed a message into a cover text in a covert manner such that the presence of the embedded message in the resulting stego-text cannot be easily discovered by anyone except the intended recipient.

This work was supported by the National Natural Science Foundation of China (No. 60773032), the Major State Basic Research Development Program of China (No. 2006CB303006), the Ph.D. Program Foundation of Ministry of Education of China (No. 20060358014), and the Natural Science Foundation of Jiangsu Province of China (No. BK2007060).

The previous work was mainly focused on linguistic steganography. The simplest method of modifying text for embedding a message is to substitute selected words by their synonyms so that the meaning of the modified sentences is preserved as much as possible. One steganography approach that is based on synonym substitution is the system proposed by Winstein[1]. There are some other approaches. Among them NICETEXT and TEXTO are most famous.

NICETEXT[2][3] system generates natural-like cover text using the mixture of word substitution and Probabilistic Context-free Grammars (PCFG). There are a dictionary table and a style template in the system in use. The style template can be generated using PCFG or a sample text. The dictionary is used to randomly generate sequences of words, while the style template selects natural sequences of parts-of-speech when controlling generation of word, capitalization, punctuation, and white space. NICETEXT system was intended to protect the privacy of cryptograms to avoid detection by censors.

TEXTO[4] is a text steganography program designed for transforming uuencoded or pgp ascii-armoured ascii data into English sentences. It was written to facilitate the exchange of binary data, especially encrypted data. TEXTO works just like a simple substitution cipher, with each of the 64 ascii symbols used by pgp ascii armour or uuencode from secret data replaced by an English word. Not all of the words in the resulting text are significant, only those nouns, verbs, adjectives, and adverbs used to fill in the preset sentence structures. Punctuation and "connecting" words (or any other words not in the dictionary) are ignored.

Markov-Chain-Based is another approach proposed by [5]. It builds a state transfer chart of Markov signal source from a sample text. A part of state transfer chart tagged with equal probabilities that are represented with one or more bit(s) is illustrated by Fig. 1. Then the algorithm uses the chart to generate cover text according to secret messages.

The linguistic steganography algorithms: NICETEXT, TEXTO and Markov-Chain-Based are of much higher hiding capacity and execution efficiency but lower security than other methods such as synonym substitution. They are still the main methods for concealing the communication in

0-7695-3102-4/08 $25.00 © 2008 IEEE
DOI 10.1109/ARES.2008.61

case that a great deal of encrypted data is being transmitted. With the use of these algorithms, attackers may carelessly take the cover text data for normal text data, and it is more innocuous to transfer the encrypted data.

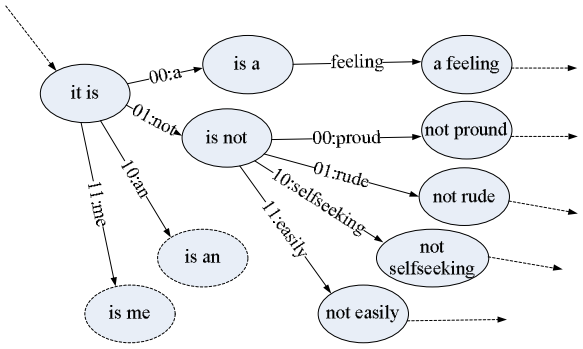

Figure 1. A part of tagged state transfer chart

However, there are some drawbacks in the above approaches discussed. For example, the first approach sometimes replaces words synonyms that do not agree with correct English usage or the genre and the author style of the given text. And the later three approaches are detectable by a human warden and are probably used in communication channels where only computers act as attackers.

A few detecting algorithms has been proposed. The paper [6] brought forward an attack against systems based on synonym substitution, especially the system presented by Winstain. The 3-gram language model was used in the attack. The experimental accuracy of this method on classification of steganographically modified sentences was 84.9% and that for unmodified sentences was 61.4%. Another detection algorithm was enlightened by the design ideas of conception chart proposed by the paper [7], which made use of the measurement of correlation between sentences. The accuracy of the simulation detection using this algorithm was 76%. The two methods fell short of accuracy that the practical employment of detection requires. In addition, the first method required a great lot of computation to calculate a large number of parameters of the 3-gram language model while the second one requires a database of rules consuming a lot of work.

This research examines drawbacks of the last three linguistic steganography approaches, aiming to accurately detect the employment of them in given text. We have used the statistical characteristics of distribution of words in the given text segments to distinguish between stego-text segments and normal text segments.

2. Important notions

2.1 Word Location

In our research, we look upon any set of consecutive sentences in the same article as a text segment. That is, we can regard a part of article or a whole article as a text segment. In our experiment, all the operations we have done to text are based on a text segment. We formalize a text segment as:

$$S = \{w_0, w_1, w_2, ..., w_{n-1}\} \quad (1)$$

Where, w_i, $0 \le i \le n-1$, is the (i+1)th word of the text segment. The word location of w_i is defined as:

$$wl_i = \frac{i}{n} \quad (2)$$

Obviously, there is $0 \le wl_i < 1$.

In fact, a text segment S probably just has m distinct words, and each word $w_k^{'}$ has n_k occurrences, there is an equation:

$$\sum_{k=0}^{m} n_k = n \quad (3)$$

Suppose the word $w_k^{'}$ has n_k occurrences in text segment S, that is to say, it has a set of word locations. A location set of $w_k^{'}$ is denoted by $LS(w_k^{'})$:

$$WL(w_k^{'}) = \{wl_{k0}, wl_{k1}, wl_{k2}, ..., wl_{k,n_k-1}\} \quad (4)$$

Subject to $wl_{k0} < wl_{k1} < wl_{k2} < ... < wl_{k,n_k-1}$

Where $\left|WL(w_k^{'})\right| = n_k$

Let $S^{'}$ denote the set of distinct words, WL denote the set of all $WL(w_k)$. That is to say

$$S^{'} = \{w_0^{'}, w_1^{'}, w_2^{'}, ..., w_{m-1}^{'}\} \quad (5)$$

$$WL = \{WL(w_k^{'}) \mid for\ all\ w_k^{'} \in S^{'}\} \quad (6)$$

We represent a text segment as another form, where

$$S = < S^{'}, WL > \quad (7)$$

2.2 Statistical Variables of Word Location

In the natural text, certain repeating word usually has an unbalanced distribution. That is, the word repeats frequently in some places, but rarely in others. We define Spread Degree (SD) of a word as its variance of word location. That is

$$SD(w_k^{'}) = \frac{1}{n_k}\sum_{i=0}^{n_k}(wl_{ki} - Avg(w_k^{'}))^2 \quad (8)$$

Where $Avg(w_k^{'}) = \frac{1}{n_k}\sum_{i=0}^{n_k} wl_{ki}$

Therefore, SD can be used as a measurement of the unbalance of the word distribution.

Formulation (7) shows that a text segment constitutes of a set of distinct words and word location set of each word. We can measure the distributions of words in the text

segment by examine the distribution of *SD*. Two statistical variables are defined as following.

The average of *SD* is

$$\overline{SD} = \sum_{i=0}^{m} SD(w_i^{'}) \frac{n_i}{n} \quad (9)$$

The variance of *SD* is

$$Var(SD) = \sum_{i=0}^{m} (SD(w_i^{'}) - \overline{SD})^2 \frac{n_i}{n} \quad (10)$$

The values of $\overline{SD}$ and $Var(SD)$ indicate the distribution characteristics of the words in a text segment. They can be used as the rule to classify the text segment to normal text or stego-text.

2.3 Support Vector Machine (SVM) Classifier

SVM is a popular technique for classification [8]. A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training data set contains one class label and several features and that in the testing data set contains only the features. The goal of SVM is to produce a model which predicts the class label of data instances in the testing data set.

In this paper, we use an existent SVM classifier [8] to classify our experiment data without getting to the bottom of how SVM classifier works.

3. Method

In natural language, normal text has many inherent statistical characteristics that can't be provided by text generated by the linguistic steganography algorithms that we investigate. In this paper, we try to find out the statistical characteristics of the word locations in a text segment. Our observation shows that words in generated text are distributed more equally than that of normal text. The probable reason is that the linguistic steganography algorithms chose words from candidate word sets in dictionary table randomly to generate a large part of the generated text's words, while the normal text involves its words in a certain subject and therefore some words will repeat frequently in particular places but rarely in others.

In our experiment, experimental data set contains training data set and testing data set, both of which consists of good data set and bad data set. The training data set is used to train the detection algorithm to well distinguish between the "good" data and "bad" data. The testing data set is the data we want to detect. We built a corpus out of the novels written by Charles Dickens, a great English novelist in the Victorian period. We named it Charles-Dickens-Corpus. We built another corpus out of novels written by a few novelists whose last name began with letter "s" and named it S-Corpus. Finally, we built a third corpus from the cover text generated by the linguistic steganography algorithms we investigated: NICETEXT, TEXTO and Markov-Chain-Based, calling it Bad-Corpus. We then built the training data set by building the good data set out of Charles-Dickens-Corpus and building the bad data set out of Bad-Corpus. The testing data set was built in the same way with its good data set built out of S-Corpus.

Comment: *Computing location set* $WL(w_k)$ *of each distinct word in S. Here,* $w_i^{'} = w_k$ *and* $w_i^{'} \in S^{'}$.

$S^{'} \leftarrow \phi$

For all $w_k \in S$ do

  If $w_k \notin S^{'}$ then

    $S^{'} \leftarrow S^{'} \cup \{w_k\}$

    $WL(w_i^{'}) \leftarrow \phi$

  End if

  $WL(w_i^{'}) \leftarrow WL(w_i^{'}) \cup \{\frac{k}{n}\}$

End for

(a)

Comment: *Computing SD of all words in* $S^{'}$

For all $w_k^{'} \in S^{'}$ do

$$Avg(w_k^{'}) \leftarrow \frac{1}{n_k} \sum_{i=0}^{n_k} wl_{ki}$$

End for

For all $w_k^{'} \in S^{'}$ do

$$SD(w_k^{'}) \leftarrow \frac{1}{n_k} \sum_{i=0}^{n_k} (wl_{ki} - Avg(w_k^{'}))^2$$

End for

(b)

Comment: *Computing average and variance of the first K SDs*

Sort all $SD(w_k^{'})$ where $w_k^{'} \in S^{'}$ according to the value of $n_k = |WL(w_k^{'})|$. Suppose the index of $w_k^{'}$ remain unchanged.

$$\overline{SD_K} = \sum_{i=0}^{K} SD(w_i^{'}) \frac{n_i}{n^{'}}$$

$$Var_K(SD) = \sum_{i=0}^{K} (SD(w_i^{'}) - \overline{SD_K})^2 \frac{n_i}{n^{'}}$$

Where $n^{'} = \sum_{i=0}^{K} n_i$

(c)

Figure 2. The algorithm of detection

In the detection, there are three main processes employed: training, testing and classifying. We apply the following procedures to both training and testing processes to abstract the classification features from the processed text segment:

First, the detection program reads the given text segment, parsing it, splitting it into words, obtaining the

word set S by formulation (1). Then the program finds the distinct words in S, adding them to S', and calculates word location of each word, adding it to corresponding word location set. At the end of this step, we get a distinct word set S' and the set of word location sets WL. See Figure 2(a).

Second, the detection program calculates SDs from S' and WL using formulation (8). In this step, we calculate SD of each word under a precondition that every occurrence of the word has same possibility. These SDs indicate the distribution characteristics of each repeating word. See Figure 2(b).

Third, the detection program sorts all SDs according to the sizes of their word location sets. As some words in the text segment may not repeat or repeat few times and they show little in the characteristic of the word distribution, we cut off these words. So the program just picks up the first K SDs to calculate a partial average and a partial variance of the SDs: $\overline{SD_K}$ and $Var_K(SD)$. The former is a total estimation of the distribution characteristics of words in the processed text segment; the latter is an estimation of the variance of distinct words distributions. Both of them are used to classify the processed text segment. See Figure 2(c).

Going through these steps described above, the program produces two files containing training result data and testing result data respectively. Each item of the training result data contains the class label and the two class features, that is $\overline{SD_K}$ and $Var_K(SD)$.The items of the testing result data contain the same thing except for the meaningless class label. These two files then delivered to the SVM classifier to complete the classification: the classifier uses the training result data file to train the classifier model and output a model file; the classifier uses the model file and the testing result data file to decide the class every testing data item represents for, outputting a file containing the classification results. The flow of the detection procedure is illustrated as Figure 3.

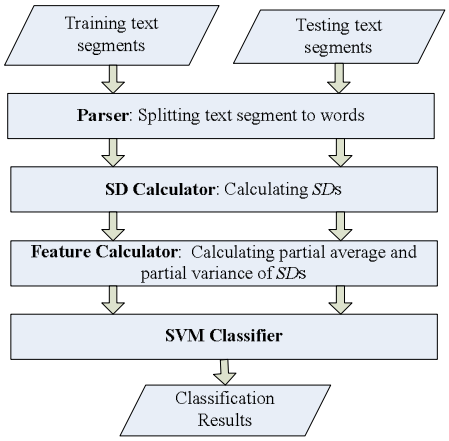

Figure 3. The flow of the detection

Figure 3 shows us that the detection procedure has a simple flow, and the training and testing process can progress in parallel. Further more, once the training process is executed, the training result data can be used repeatedly unless some parameters of the program are changed.

4. Results and Discussion

As what we have explained, the experimental data set in our experiment is made up of training data set and testing data set. Both data sets contain good data set and bad data set. The composing of the experimental data set is showed in Table 1.

Table 1 the composing of experimental data set

Text Data Set	Data Type	Segment Number	Sum
Training Set	Good Set	117	217
	Bad Set	100	
Testing Set	Good Set	146	468
	Bad Set	322	

In the training data set, the 117 text segments in good set come from Charles-Dickens-Corpus and the 100 text segments in bad set come from Bad-Corpus. In the testing data set, the 146 text segments in good set come from S-Corpus while the 322 text segments in the bad set also come from Bad-Corpus without overlapping with text segments in training data set.

In the training process, detection program processes the text segments in training data set to obtain their $\overline{SD_K}$ s and $Var_K(SD)$ s. That is to say, each text segment in training data set has an output containing its $\overline{SD_K}$ and $Var_K(SD)$, which will be used as classification features in the SVM classifier. These outputs are stored to a training result file. In the testing process, the same thing is done to text segments in testing data set with their outputs stored to a testing result file. In the classification process, the SVM classifier uses the training result file to train its classification model and then uses the testing file to classify the text segments in testing data set.

In the experiment, text segments with a size of 5kB, 10kB, 20kB, 30kB and 40kB are detected. The detection results are listed in Table 2.

Table 2 Detection results with different segment size

Text segment size	Success	Fail	accuracy
5kB	409	59	87.39%
10kB	447	21	95.51%
20kB	461	7	98.50%
30kB	464	4	99.15%
40kB	466	2	99.57%

In Table 2, we can see that the accuracy increases along with the increase of segment size. The accuracies are pretty high when the segment size is not less than 10kB.

Surprisingly, the accuracies exceed 99% when the segment size is equal to or more than 30kB. Above all, the table shows that the detection algorithm works very well even when the segment size is 5kB.

The figures from Figure 4 to Figure 13 show the distribution of classification features of training and testing text segments with different size of text segment. Each point in the figures with Avg(SD) ($=\overline{SD_K}$) as X-axis value and Var(SD) ($=Var_K(SD)$) as Y-axis value represents a normal text segment when colored green, a stego-text segment when colored red. When the size of text segments varies from 5kB to 40kB, we find that the red points shrink to the left-upper corner while the green points still spread to the right-lower corner.

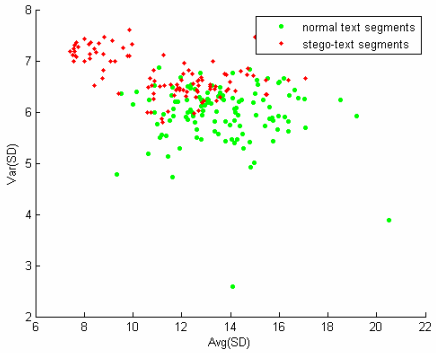

Figure 4. Training results (segment size = 5kB)

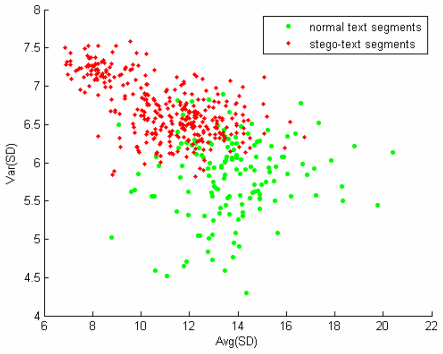

Figure 5. Testing results (segment size = 5kB)

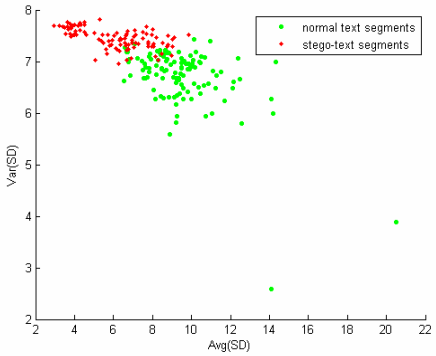

Figure 6. Training results (segment size = 10kB)

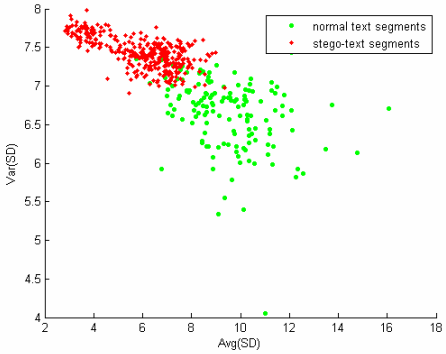

Figure 7. Testing results (segment size = 10kB)

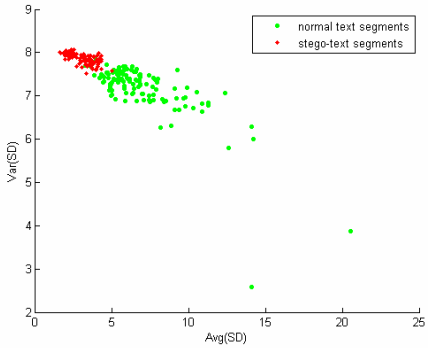

Figure 8. Training results (segment size = 20kB)

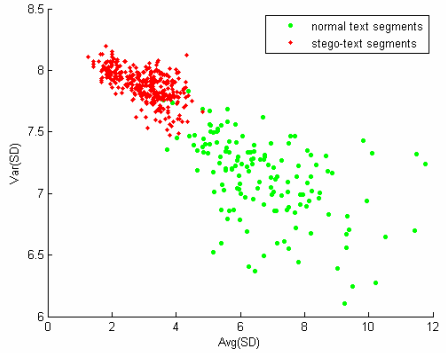

Figure 9. Testing results (segment size = 20kB)

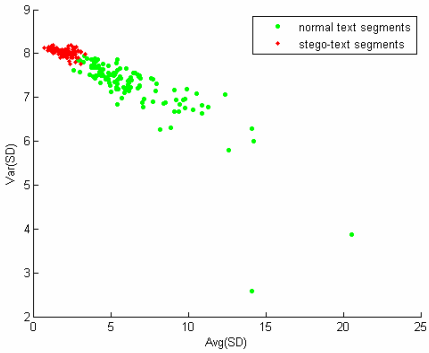

Figure 10. Training results (segment size = 30kB)

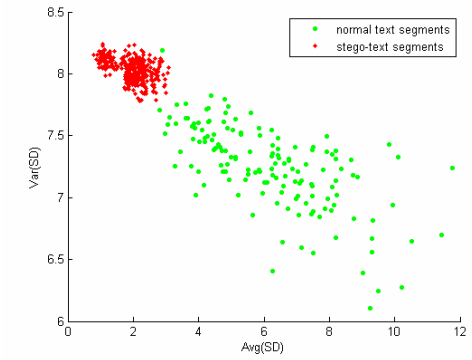

Figure 11. Testing results (segment size = 30kB)

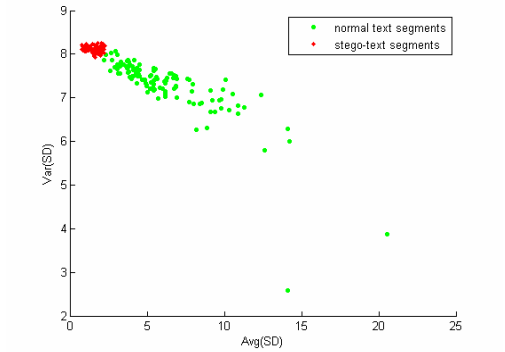

Figure 12. Training results (segment size = 40kB)

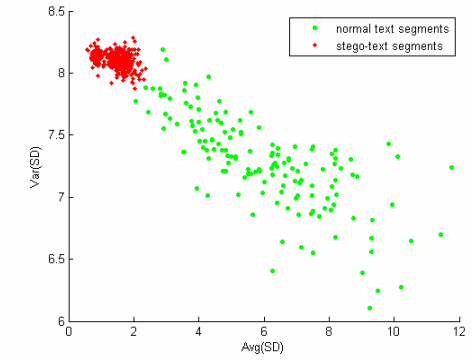

Figure 13. Testing results (segment size = 40kB)

The change of the figures according to the size of text segment shows that the generated text segments has a lower Avg(SD) and has a higher Var(SD), that is to say, a certain word in the generated text segment is distributed more equally than that in the normal text segment, while the variance in distinct word distributions in the generated text segment is more apparent than that in the normal text segments.

5. Conclusion

In this paper, a statistical algorithm for linguistic steganography detection has been presented. The algorithm takes advantage of distribution of words in the text segment detected. In the experiment of detecting the three different linguistic steganography methods: NICETEXT, TEXTO and Markov-Chain-Based, the total accuracies on discovering stego-text segments and normal text segments are found to be 87.39%, 95.51%, 98.50%, 99.15% and 99.57% respectively when the segment size is 5kB, 10kB, 20kB, 30kB and 40kB. Our initial results show that the linguistic steganalysis based on distribution of words is promising.

Many interesting and new challenges are involved in the analysis of linguistic steganography algorithms, which is known as linguistic steganalysis that have little or no counterpart in other media domains, such as images or video. Linguistic steganalysis performance strongly depends on many factors such as the length of the hidden message and the way to generate cover text. However, our algorithm shows that a method of blind detection of linguistic steganography algorithms that generating cover text is feasible.

6. References

[1] Winstein, Keith. Lexical steganography through adaptive modulation of the word choice hash. http://alumni.imsa.edu/~keithw/tlex/lsteg.ps. Ms.

[2] Chapman, Mark. Hiding the Hidden: A Software System for Concealing Ciphertext as Innocuous Text. http://www.NICETEXT.com/NICETEXT/doc/thesis.pdf. 1997.

[3] Chapman, Mark, George Davida and Marc Rennhard. A Practical and Effective Approach to Large-Scale Automated Linguistic Steganography. Lecture Notes in Computer Science, Volume 2200, Springer-Verlag: Berlin Heidelberg. Jan 2001. 156-167.

[4] K. Maher. TEXTO. URL:ftp://ftp.funet.fi/pub/crypt/steganography/texto.tar.gz

[5] WU Shu-feng, HUANG Liu-sheng. Research on Information Hiding. Degree of master, University of Science and Technology of China, 2003.

[6] CM Taskiran, U Topkara, M Topkara et al. Attacks on lexical natural language steganography systems. Proceedings of SPIE, 2006.

[7] ZHOU Ji-jun, YANG Zhu, NIU Xin-xin et al. Research on the detecting algorithm of text document information hiding. Journal on Communications. Dec. 2004 Vol.25, No. 12, 97-101.

[8] CW Hsu, CC Chang, CJ Lin. A Practical Guide to Support Vector Classification. 2003. http://www.csie.ntu.edu.tw/～cjlin.