Attacks on Lexical Natural Language Steganography Systems

Cuneyt M. Taskiran[a], Umut Topkara[b], Mercan Topkara[b], and Edward J. Delp[c]

[a]Motorola Labs, Multimedia Research Lab, Schaumburg, Illinois 60196
[b]Center for Education and Research in Information Assurance (CERIAS)
Purdue University, West Lafayette, Indiana, 47907
[c]Video and Image Processing Laboratory (*VIPER*)
School of Electrical and Computer Engineering, Purdue University, Indiana, 47907

ABSTRACT

Text data forms the largest bulk of digital data that people encounter and exchange daily. For this reason the potential usage of text data as a covert channel for secret communication is an imminent concern. Even though information hiding into natural language text has started to attract great interest, there has been no study on attacks against these applications. In this paper we examine the robustness of lexical steganography systems.In this paper we used a universal steganalysis method based on language models and support vector machines to differentiate sentences modified by a lexical steganography algorithm from unmodified sentences. The experimental accuracy of our method on classification of steganographically modified sentences was **84.9%**. On classification of isolated sentences we obtained a high recall rate whereas the precision was low.

Keywords: steganalysis, lexical steganography, natural language steganography, universal steganalysis, statistical attacks

1. INTRODUCTION

The importance and size of text data is increasing at an accelerating pace, spurred both by the central role Internet information plays in people's lives and by the rise of text-based information dissemination media, such as email, blogs and text messaging. This increase in the significance of electronic text in turn creates increased concerns about the usage of text media as a covert channel of communication. These concerns are especially urgent for text media since it is easier for non-tech-savvy users to modify text documents compared to other types of multimedia documents, such as images and video. Such covert means of communication is known as steganography. Steganographic methods aim to embed a message in a cover object in a covert manner such that the presence of the embedded message in the resulting stego-object cannot be easily discovered by anyone except the intended recipient. Steganographic applications require the flexibility to alter cover object in a stealthy way to be able to embed the hidden information.

A typical scenario for steganography is the case of two parties who exchange digital objects through a public communication channel. They also desire to exchange secret messages; however, they do not want the existence of this secret communication to be noticed by others. They also do not want to achieve confidentiality through encryption, because the exchange of encrypted messages would reveal the existence of their secret communication. For this reason, they use a steganographic algorithm to embed secret messages into cover objects to obtain *stego-objects*, and exchange these stego-objects through the public communication channel. While traversing the public communications channel, the stego-objects may come under intentional or unintentional attacks. Examples of unintentional attacks are transmission errors, lossy compression, and changing the visual properties of the stego-document. Intentional attacks, on the other hand, are deliberate attempts to distinguish stego-objects from unmodified objects and thus detect the presence of covert communication. Attack methods generally exploit the fact that embedding information usually changes the statistical properties of the objects compared to typical unmodified objects.

Natural language (NL) processing based information hiding techniques aim to embed information in text documents by manipulating their lexical, syntactic, or semantic properties[1] while preserving the meaning as much as possible. These techniques are more robust than methods that modify the appearance of text elements such as fonts or inter-line spacing, in resisting attacks to remove or obtain the information hidden in a text

document. Compared to methods developed for image, video, and audio domains, NL information hiding is still a new area that has its unique challenges. A small number of NL watermarking and steganography methods have been described in the literature. However, since the theory and practice of NL information hiding is still in the process of being developed, there has been little emphasis in previous literature on testing the security, stealthiness and robustness of the proposed methods using various attacks.

As mentioned above NL steganography methods may employ lexical, syntactic, or semantic linguistic transformations to manipulate cover text and embed a message. In this paper we will focus on methods that perfom *lexical steganography*, which is based on changing the words and other tokens in the cover text. We test the stealthiness of lexical steganography systems by developing an attack method that determines whether the choice of lexical tokens in a given text has been manipulated to embed hidden information. To the best of authors' knowledge, this is the first study that assesses the robustness of existing lexical steganography systems against statistical attacks based on text analysis.

Our approach relies on the fact that the text manipulations performed by the lexical steganography system, though they may be imperceptible, nevertheless change the properties of the text by introducing language usage that deviates from the expected characteristics of the cover text. Our method may be summarized as follows: First, we capture cover-text and stego-text patterns by training language models on unmodified and steganographically modified text. Second, we train a support vector machine (SVM) classifier based on the statistical output obtained from the language models. Finally, we classify a given text as unmodified or steganographically modified based on the output of the SVM classifier. Our choice of the SVM classifier was motivated by the facts that they were used successfully for text classification[2] and that were proven to be effective as a universal steganographic attack when images were used as cover objects.[3,4] We demonstrate the performance of our approach on a lexical steganography system proposed by Winstein.[5]

The organization of the paper is as follows: In Section 2 we provide a brief survey of the methods perviously proposed in NL steganography. Section 3 describes in detail the lexical steganography system that we have used in our experiments. Section 4 introduces the language modeling scheme used by our system. In Section 5 we present the results of our steganography detection experiments. Finally, conclusions are presented in Section 6.

2. PREVIOUS APPROACHES TO NATURAL LANGUAGE STEGANOGRAPHY

Compared to similar work in the image and video domains, work in natural language (NL) steganography and watermarking has been scarce. The previous work in information hiding into natural language text was mainly focused on steganography, this is probably due to the fact that it is challenging to derive robust watermarking methods for text. In this section we review the previous work done in NL steganography.

The simplest method of modifying text for embedding a message is to substitue selected words by their synonyms so that the meaning of the modified sentences are preserved as much as possible. One steganography approach that is based on synonym substitution is the system proposed by Winstein,[5] which we will describe in detail in Section 3. The other two approaches to NL steganography are based on generating a random cover text, and easily detectable by a human warden. These last two approaches are explained briefly below.

2.1. Using Probabilistic Context-Free Grammars to Generate Cover Text

A *probabilistic context-free grammar* (PCFG) is a commonly used language model where each transformation rule of a context-free grammar has a probability associated with it.[6] A PCFG can be used to generate word sequences by starting with the root node and recursively applying randomly chosen rules. Conversely, a word sequence belonging to the language produced by a PCFG can be parsed to reveal a possible sequence of possible rules that can produce it.

In the mimicry text approach described in[7] a cover text is generated using a PCFG that has statistical properties close to normal text. This is achieved by assigning a Huffman code to each grammar rule based on the probability of the rule. The payload string is then embedded by choosing the grammar rule whose code corresponds to the portion of the message being embedded. An example sentence generated by this technique is illustrated in Figure 1. The PCFG and the corresponding rule probabilities are learned using a corpus.

Rule #	Rule	code	prob.
(1)	$S \Rightarrow AB$	0	0.5
(2)	$S \Rightarrow CB$	1	0.5
(3)	$A \Rightarrow$ She	00	0.25
(4)	$A \Rightarrow$ He	01	0.25
(5)	$A \Rightarrow$ Susan	10	0.25
(6)	$A \Rightarrow$ Alex	11	0.25
(7)	$B \Rightarrow$ likes D	0	0.5
(8)	$B \Rightarrow$ detests D	10	0.25
(9)	$B \Rightarrow$ wants D	110	0.125
(10)	$B \Rightarrow$ hates D	111	0.125
(11)	$C \Rightarrow$ Everybody	0	0.5
(12)	$C \Rightarrow$ The cleaning lady	10	0.25
(13)	$C \Rightarrow$ A nice kid	11	0.25
(14)	$D \Rightarrow$ milk.	00	0.25
(15)	$D \Rightarrow$ apples.	01	0.25
(16)	$D \Rightarrow$ pumpkin pie.	10	0.25
(17)	$D \Rightarrow$ cookies.	11	0.25

(a)

Position	Prefix	Rule	output string
•1011001	1	2	CB
1 • 011001	0	11	Everybody B
10 • 11001	110	9	Everybody wants D
10110 • 01	01	15	Everybody wants apples.

(b)

Figure 1. Using a probabilistic context-free grammar to generate cover text for the secret payload 1011001. (a) A very simple probabilistic context-free grammar. The Huffman code corresponding to each rule is also listed. (b) Generation of cover text using the rules determined by the payload.

The problem with this method is that even within limited linguistic domains, deriving a PCFG that models natural language is a daunting task. Furthermore, some aspects of language cannot be modeled by context-free grammars. Because of these reasons, cover text produced by PCFGs tend to be ungrammatical and nonsensical. This makes it easy for native speakers to detect such texts, which defeats the steganographic purpose of the method. Therefore, this method can only be used in communication channels where only computers act as attackers.

2.2. Generating Cover Text Using Hybrid Techniques

The *NICETEXT* system[8,9] for the generation of natural-like cover text according to a given message uses a mixture of the method discussed above and synonym substitution. The system has two components: a dictionary table and a style template. The dictionary table is a large list of (*type,word*) pairs where the *type* may be based on the part-of-speech[8] of *word* or its synonym set.[9] Such tables may be generated using a part-of-speech tagger or WordNet, as will be discussed in the next section. The dictionary is used to randomly generate sequences of words. The style template, which is conceptually similar to the PCFG of Section 2.1, improves the quality of the cover text by selecting natural sequences of parts-of-speech while controlling word generation, capitalization, punctuation, and white space generation. An example of a simple dictionary and how the style template affects the generated text is illustrated in Figure 2. A dictionary containing more than 200,000 words categorized into more than 6,000 types was used in.[8] Different style templates, such as Federal Reserve Board meeting minutes or Aesop's Fables, were learned using online text collections and employed in the NICETEXT system.

3. INFORMATION EMBEDDING THROUGH SYNONYM SUBSTITUTION

Synonym substitution is the most widely used linguistic transformation employed to modify text for information hiding. In synonym substitution, the message is embedded by selecting words from the cover text and replacing them by one of their synonyms according to a message encoding rule. In this section we describe one such system, the Tyrannosaurus Lex (T-Lex) system proposed by Winstein[5] and discuss some of its drawbacks that makes it susceptible to universal statistical steganalysis attacks. By universal attack we mean that our method does not take into consideration any particular aspects of the T-Lex system. Therefore, although our discussion

Type	Code	Word
name–male	0	ned
name–male	1	tom
name–female	0	jody
name–female	1	tracy

(a)

Style	Payload	Output string
name–male name–male name–male	011	ned tom tom
name–male name–male name–female	011	ned tom tracy
name–male name–female name–male	011	ned tracy tom
name–male name–female name–female	011	ned tracy tracy
name–female name–male name–male	011	jody tom tom
name–female name–male name–female	011	jody tom tracy
name–female name–female name–male	011	jody tracy tom
name–female name–female name–female	011	jody tracy tracy

(b)

Figure 2. Example of a simple dictionary and how the style template affects the output for the *NICETEXT* system.[8] (a) A simple dictionary with two types, name–male and name–female. (b) Using a style and the dictionary in (a) to generate text corresponding to a payload string.

and experiments focus on the T-Lex system, we believe our method is applicable to other synonym substitution based information hiding techniques.

3.1. The T-Lex Lexical Steganography System

The basic problem that face synonym substitution is that, in order to preserve the meaning of the sentences that are being manipulated as much as possible, synonyms that are substituted should have the same senses as the words to be replaced. However, determining the correct sense of a given word in a given context, known as the *word sense disambiguation* task in NLP, is a hard problem, since it is hard to even derive a general definition of the word sense concept.[10] For example the word "bank" may have the senses financial institution, river edge, or something to sit on, depending on the particular context it is used.

One approach to select synonyms with correct sense is to use WordNet.[11] WordNet is an electronic dictionary that groups English words into sets of synonyms called *synsets*, provides short definitions, and lists semantic relations between these synonym sets. In the T-Lex system, a database of synonyms is first generated using the information provided in WordNet. Not all the synonyms listed in WordNet are included in T-Lex's synonym database. In order to ensure that only words with the close senses are replaced with each other, only the words that caryy a particular synset pattern are taken into account. For example, assume that words w_1, w_2, w_3 all have more than one sense and belong to the synsets $S_1 : \{w_1, w_2\}$, $S_2 : \{w_1, w_2, w_3\}$. In this case, even though words w_1 and w_2 have more than one sense, they can still be interchanged in all contexts without damaging the semantic structure. Applying considerations such as the one described above, Winstein obtained synsets that contain approximately 30% of WordNet's 70,803 single word entries as T-Lex's synonym database. The mean synset size for the database was 2.56 words while the maximum synset size was 13.

A given message is embedded into the cover text using the synset database as follows. First, the letters of the message text are Huffman coded according to English letter frequencies. Then, the Huffman code binary string is expressed in mixed radix form according to the current state of the embedding algorithm. As a simple example,[12] assume that the string to be embedded is $(101)_2$ and that currently the following sentence is being considered.

$$\texttt{San Jose is a} \left\{ \begin{array}{cc} & \textbf{excellent} \\ 0 & \textit{decent} \\ 1 & \textit{fine} \\ 2 & \textit{great} \\ 3 & \textit{wonderful} \end{array} \right\} \texttt{little} \left\{ \begin{array}{cc} & \textbf{city} \\ 0 & \textit{metropolis} \\ 1 & \textit{town} \end{array} \right\}.$$

In this example the boldface words are the original words that will be substituted by a word from their synsets shown below them. Note that words in a synset are indexed according to their alphabetical order. In mixed

radix form each digit may have a different base determined by the size of the synset at that location. For the above example we have

$$\begin{pmatrix} a_1 & a_0 \\ 4 & 2 \end{pmatrix} = 2a_1 + a_0 = 5,$$

with the constraints that $0 \leq a_1 < 4$ and $0 \leq a_0 < 2$. Thus, we obtain the values $a_1 = 2$ and $a_0 = 1$ which indicates that the boldface words should be replaced by the words *great* and *town*.

3.2. Drawbacks of the T-Lex System

We have embedded a short message into Jane Austen's novel *Pride and Prejudice* obtained from the Gutenberg Project. Four examples of the changes made by the T-Lex system are shown below, where the first sentence fragment is the original version and the second is the steganographically modified version.

...I can tell you, to be	making	new acquaintances every day...
...I can tell you, to be	fashioning	new acquaintances every day...
An invitation to dinner was soon	afterwards	dispatched;
An invitation to dinner was soon	subsequently	dispatched;
...and make it still better, and say	nothing	of the bad–belongs to you alone.
...and make it still better, and say	nada	of the bad–belongs to you alone.
Bingley likes your sister	undoubtedly;	
Bingley likes your sister	doubtless;	

The above examples illustrate two shortcomings of the T-Lex system. First, it sometimes replaces words with synonyms that do not agree with correct English usage, as seen in the phrase *soon subsequently dispatched*. Second, T-Lex also substitutes synonyms that do not agree with the genre and the author style of the given text. It is clear that the word *nada* does not belong to Jane Austen's style. Furthermore, the string *say nada of* is not part of typical English usage.

Both types of errors made by the T-Lex system are caused by the fact when choosing synonyms from synsets, important factors such as genre, author style, and sentence context are not taken into account. Synonyms that deviate from common usage can be detected using language models trained on a collection of typical text that has the same genre and style as the one being analyzed. This shortcoming is not unique to the T-Lex system but is a problem with all synonym substitution methods. One can argue that these systems may be improved by making use of information derived from language models while synonyms are being chosen during the embedding process. However, such synonym substitution methods would have high computational complexity.

4. TRAINING STATISTICAL LANGUAGE MODELS

A language model (LM) is a statistical model that estimates the prior probabilities of n-gram word strings.[13] An n-gram LM models the probability of the current word in a text based on the $n-1$ words preceding it; hence, an n-gram model is a $n-1^{th}$ order Markov model, where, given the probability of a set of n consecutive words, $W = \{w_1, \ldots, w_n\}$, the LM probability is calculated using

$$P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_0, \ldots, w_{i-1}), \tag{1}$$

where the initial condition $P(w_1|w_0)$ is chosen suitably. We have used trigram models in our experiments, i.e. LMs with $n = 3$. In practice, these LM probabilities are estimated from a set of training text data.

One approach to estimate n-gram probabilities from training text is to count the number of n-grams occurring in the text and then define the probability as the maximum likelihood estimate.

$$P(W) = r(W)/N \tag{2}$$

where $r(W)$ is the frequency of the n-gram W. This simple approach has a big drawback: Since the number of possible n-grams grows exponentially with increasing n, no matter how large a training text collection is used, there will be many n-grams that will not be observed. The maximum likelihood approach in this case leads to two related problems: First, too much probability will be assigned to n-grams that are observed and none to the ones that are not observed. Second, many n-grams will get assigned a probability of zero, this is referred to as the "zero frequency problem".

In order to solve the first problem, a method to adjust observed n-gram frequencies is used. One such method is the well-known Good-Turing estimator, which adjusts the observed n-gram frequencies using the following definition before using Equation 2.[14]

$$r^* = (r+1)\frac{E(N_{r+1})}{E(N_r)} \tag{3}$$

where r is the frequency, r^* is the adjusted frequency, N_r is the number of n-gram-types that occur r times, and $E(N_r)$ is the expected value of N_r. Another discounting method is the Witten-Bell method where the first occurrence of each word is taken to be a sample for the "unseen" event. In this approach the amount of discounting for each word is relative to the number of distinct word types that follow it. For the special case of bigrams (i.e., n=2), partitioning the vocabulary relative to the current bigram W, the adjustment equation is given as

$$r^* = \begin{cases} \frac{T(W)}{Z(W)}\frac{N}{N+T(W)}, & r = 0 \\ r\frac{N}{N+T}, & r > 0 \end{cases} \tag{4}$$

In the above equation $Z(W)$ is the number of word types *not* seen after bigram W and $T(W)$ is the number of word types seen after W.

In order to address the second problem, "zero frequency problem", typically model smoothing is employed. One common model smoothing technique is the Katz's back-off rule, which states that the modeling algorithm estimates n-gram probabilities when there is "enough" data, otherwise tries to estimate probabilities for $n-1$-grams. If necessary, this backing off process is repeated .

The goodness-of-fit for a LM is usually measured by a quantity called *perplexity* in the natural language processing field, rather than using model entropy, as is common in signal processing. The perplexity for a LM is calculated using

$$\text{perplexity(LM)} = 2^{-\frac{1}{N}\sum \log_2 P(data|model)} \tag{5}$$

In our experiments, we have used Stanford Research Institute Language Modeling (SRILM) Toolkit[15] to train LMs that model language usage patterns for unmodified and steganographically modified text. SRILM supports LM creation and evaluation, where LM creation entails the estimation of model parameters from a collection of training text data and LM evaluation refers to calculating the probability of a given piece of text according to the model. SRILM provides a large number of parameters for LM estimation and evaluation. Most important parameters are:

- the order of n-grams to use
- the type of discounting algorithm to use. Supported methods include Good-Turing, absolute, Witten-Bell, and modified Kneser-Ney[14]
- an optional predefined vocabulary
- whether to discard "unknown words" or treat them as special tokens
- whether to collapse case distinctions in the input text

SRILM uses Katz's back-off model as the default for smoothing of language models.

5. EXPERIMENTS AND RESULTS

In order to obtain our text data collection we have employed the following procedure. First, we processed text from the Reuters news corpus in Government/Social (GCAT) topic category.[16] We have used the Stanford parser[17] to obtain accurate sentence boundaries. The parser output also contains part of speech (POS) tags. Using POS tags we selected only those sentences that contain at least one word tagged as *Verb*. This was done in order to obtain sentences that are syntactically as typical as possible. 40,000 resulting sentences were then selected as our data set. No further processing was performed on the text, that is, numbers, special markers, and punctuation were left in.

We then trained trigram models for sentences using SRILM Toolkit. Alternating values were assigned to three of SRILM's important parameters in order to obtain 8 different language models. These parameters were:

- Vocabulary: closed or open
- Model order: regular or skip
- n-gram frequency cutoff: include or exclude n-grams observed once

The output of a closed vocabulary language model for the sentence

`Manfred Bender scored from a header in the 80th minute, four transactions after coming on.`

is given below. Note that the probability assigned to the proper name is 0, since this name is not in the vocabulary of the model.

$p(< unk > | < s >)$ $= [OOV]$ 0 $[-inf]$
$p(< unk > | < unk > ...)$ $= [OOV]$ 0 $[-inf]$
$p(scored| < unk > ...)$ $= [1gram]$ $5.82849e-05$ $[-4.23444]$
$p(from|scored...)$ $= [2gram]$ 0.012987 $[-1.88649]$
$p(a|from...)$ $= [2gram]$ 0.0458446 $[-1.33871]$
$p(header|a...)$ $= [2gram]$ $8.36651e-05$ $[-4.07746]$
$p(in|header...)$ $= [1gram]$ 0.0101141 $[-1.99507]$
$p(the|in...)$ $= [2gram]$ 0.227927 $[-0.642205]$
$p(80th|the...)$ $= [1gram]$ $4.98111e-07$ $[-6.30267]$
$p(minute|80th...)$ $= [1gram]$ $2.26727e-05$ $[-4.6445]$
$p(,|minute...)$ $= [2gram]$ 0.0617284 $[-1.20951]$
$p(four|,...)$ $= [2gram]$ 0.000479659 $[-3.31907]$
$p(transactions|four...)$ $= [1gram]$ $4.33264e-06$ $[-5.36325]$
$p(after|transactions...)$ $= [1gram]$ 0.00122369 $[-2.91233]$
$p(coming|after...)$ $= [2gram]$ 0.00244599 $[-2.61155]$
$p(on|coming...)$ $= [3gram]$ 0.2 $[-0.69897]$
$p(.|on...)$ $= [2gram]$ 0.00166324 $[-2.77904]$
$p(< /s > |....)$ $= [3gram]$ 0.827586 $[-0.0821868]$
17 words, 2 OOVs
0 zeroprobs, logprob= -44.0975, ppl= 570.284, ppl1= 870.624

where the last two quantities listed are perplexities of the model for the sentence suing two different methods.

For each of the 8 language models we extract a feature vector for each sentence in our training set using the following features obtained from model outputs (quantities in parentheses are the values for the above example:

- Word statistics (Number of words (12), Out Of Vocabulary (OOV) words (2), Zero probability words (0))
- Mimimum n-gram context length matchihg the model (1), and maximum (3) respectively
- Statistics of model probability estimations for n-grams in the sentence (mean, min, max) (0.165, 0, 0.827586)

- Model statistics for log probability of n-grams in the sentence (min, max) (-inf, -0.0821868)
- Total probability of the sentence (-44.0975)

These features are then concatenated to obtain one feature vector for each sentence.

We partitioned the 40,000 sentences in our data set into two sets, a 30,000 sentence set, on which we trained out 8 LMs by alternating the values for the parameters listed above, and a 10,000 sentence set, on which we performed our classification experiments. We inserted a 781-bit long plain text message into the text consisting of 10,000 sentences using the T-Lex system. This resulted in 1169 steganographically modified sentences and 8831 unmodified sentences. We then extracted feature vectors from each of these 10,000 sentences using the features described above.

A support vector machine (SVM) classifier was trained on the features of randomly selected 500 unmodified sentences and 500 steganographically modified sentences. We used the `libsvm` library[18] to train the SVM classifier. Using the SVM classifier so obtained, we classified the remaining 669 steganographically modified and 8331 unmodified sentences. The accuracy on steganographically modified sentences was found to be **84.9%** and that for unmodified sentences to be **38.6%**. Note that the classification of each sentence is performed independently. In practice whole paragraphs would be steganographically modified, so we believe that classification on multiple consecutive sentences picked from a text will give much better results. This may easily be achieved by using the output of the SVM classifier to train a second SVM classifier for the whole text, rather than on a sentence by sentence basis.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented, to the best of authors' knowledge, the first work on performing text steganalysis. We used a steganalysis method based on language models and support vector machines to differentiate sentences modified by a lexical steganography algorithm from unmodified sentences. Our method is independent from the specifics of the steganography algorithm, and is universal in this sense. However our method is only applicable to languages for which statistical n-gram language models are effective. On classification of isolated sentences we obtained a high recall rate whereas the precision was low. We are planning to extend our feature set to include other text statistics at the level of groups of sentences as well as individual sentences. Our technique is heavily based on the presence of a lexicon, hence we expect its performance to suffer for steganalysis applications to inflectional and compounding languages such as German, Finnish and Turkish. We should also point out that the newswire domain that we performed our experiments on is one of the most challenging domains for statistical text analysis due to the fact that its vocabulary is very large and it contains many named entities appearing in different news stories.

Many interesting and new challenges are involved in natural language steganalysis that have little or no counterpart in other media domains, such as images or video. Building comprehensive language models is difficult. Steganalysis performance strongly depends on many factors such as author writing styles, genre, intended audience, and the particular content of the text. However, we believe that our initial results show that the universal steganalysis approach that was previously applied to image steganalysis[4] is promising in the text domain, too.

REFERENCES

1. M. Topkara, C. M. Taskiran, and E. Delp, "Natural language watermarking," *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents VII*, 2005.
2. T. Joachims, "Transductive inference for text classification using support vector machines," *Proceedings of 16th International Conference on Machine Learning*, 1999, Bled, SL, pp. 200–209.
3. I. Avcibas, N. Memon, and B. Sankur, "Steganalysis of watermarking and steganographic techniques using image quality metrics," *IEEE Transactions on Image Processing*, vol. 2, no. 12, pp. 221–229, 2003.
4. S. Lyu and H. Farid, "Detecting Hidden Messages using Higher-Order Statistics and Support Vector Machines," *Proceedings of the Fifth Information Hiding Workshop*, vol. LNCS, 2578, October, 2002, Noordwijkerhout, The Netherlands, Springer-Verlag.

5. "The tyrannosaurus lex system available at http://www.fb10.uni-bremen.de/anglistik/langpro/nlg-table/nlg-table-root.htm."
6. C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.
7. P. Wayner, "Mimic functions," *CRYPTOLOGIA*, vol. XVI, no. 3, pp. 193–214, July 1992.
8. M. Chapman and G. Davida, "Hiding the hidden: A software system for concealing ciphertext in innocuous text," *Proceedings of the International Conference on Information and Communications Security*, vol. LNCS 1334, 1997, Beijing, China.
9. M. Chapman and G. Davida, "Plausible deniability using automated linguistic stegonagraphy," *roceedings of the International Conference on Infrastructure Security*, October 1-3 2002, Bristol, UK, pp. 276–287.
10. N. Ide and J. Vronis, "Word sense disambiguation: The current state of the art," *Computational Linguistics*, vol. 24, no. 1, 1998.
11. C. Fellbaum, *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
12. R. Bergmair, "Towards linguistic steganography: A systematic investigation of approaches, systems, and issues.," tech. rep., University of Derby, August 2004.
13. A. Stolcke, "Srilm - an extensible language modeling toolkit," *Proceedings of International Conferrence on Spoken Language Processing*, 2002.
14. S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, 1996, Morgan Kaufmann Publishers, pp. 310–318.
15. S. R. Institute, "Stanford research institute language modeling toolkit," *http://www.speech.sri.com/projects/srilm/*.
16. "Reuters corpus," *http://about.reuters.com/researchandstandards/corpus/index.asp*.
17. D. Klein and C. D. Manning, "Accurate unlexicalized parsing," *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, 2003, Morristown, NJ, USA, Association for Computational Linguistics, pp. 423–430.
18. C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.