International Journal of Multimedia and Ubiquitous Engineering
Vol. 4, No. 2, April, 2009

Steganalysis of Synonym-Substitution Based Natural Language Watermarking

Zhenshan Yu, Liusheng Huang, Zhili Chen, Lingjun Li,
Xinxin Zhao, and Youwen Zhu
National High Performance Computing Center at Hefei
Department of Computer Science and Technology, USTC
Hefei, P.R.China
zsyu@mail.ustc.edu.cn

Abstract

Natural language watermarking (NLW) is a kind of digital rights management (DRM) techniques specially designed for natural language documents. Watermarking algorithms based on synonym substitution are the most popular kind, they embeds watermark into documents in linguistic meaning-preserving ways. A lot of work has been done on embedding, but only a little on steganalysis such as detecting, destroying, and extracting the watermark. In this paper, we try to distinguish between watermarked articles and unwatermarked articles using context information. We evaluate the suitability of words for their context, and then the suitability sequence of words leads to the final judgment made by a SVM (support vector machine) classifier. IDF (inverse document frequency) is used to weight words' suitability in order to balance common words and rare ones. This scheme is evaluated on internet instead of in a specific corpus, with the help of Google. Experimental results show that classification accuracy achieves 90.0%. And further analysis of several influencing factors affecting detection effects is also presented.

1. Introduction

Protection of intellectual product rights is the foundation of human technologic development, especially for civilian technologies. Digital rights management (DRM) which concerns access control technologies used by publishers and copyright holders to limit usage of digital media or devices, is always a hot research topic. Most researches concern non-text media like image, video and audio[1]; though natural language documents are major digital data everyday people meet. Compared to other types of carrier, text is low in redundancy, and natural language rules, for instance, spelling, word-building, syntax, grammar, semantics, etc; limit manipulation of text[2]. So they are both great challenges to conceal watermark in text properly and to detect such concealment.

Natural language watermarking (NLW) embeds watermark into documents to limit usage of these documents. Different from format-changing methods which modify format of documents such as word-spacing or line-spacing, NLW modifies the content of documents using linguistic methods while preserving meaning and style. Because NLW methods are immune to re-typesetting attacks, they are considered as promising and dominant ways in the field of digital document rights management.

Though a lot of ingenious NLW algorithms have been designed varying from syntactic ways to semantic ways, only a little work has been done on steganalysis of these schemes. In

this paper, we analyze one kind of natural language watermarking schemes that use synonym substitution. Hiding algorithm replaces changeable words (called keywords) in cover-text with their synonyms according to designed rules in order to embed messages. Because all substitutions are done between synonyms, meaning of the text and syntax of sentences are preserved. Algorithms based on synonym substitution are widely used in NLW.

We try to distinguish between watermarked articles and unwatermarked articles. Concretely, we focus on the context suitability which may be significantly changed after synonym substitution. In our detecting scheme, every keyword plays a game called ” word election” with its synonyms, its percentage of vote corresponds to its suitability for the context. Finally, after reading through the whole article, value of suspicion is computed from the sequence of the keywords' percentage of vote. If the suspicion is above a threshold, we estimate it as a watermarked article, otherwise it is unwatermarked.

How to evaluate the suitability of words for a context is the central issue. A rare word may be more suitable for a context than a common word. But because of its rareness, it seldom appears in corpus. We use IDF to balance common words and rare ones. IDF is an important concept in information retrieval. Details will be shown in section4.

We did our experiments on internet instead of in a single corpus, because it is very difficult to get useful context information in a million words corpus. There are billions of documents on internet, weighted by influence naturally. With the help of search engine or other tools, people can get lots of meaningful information. We use Google to get context information over internet in our experiment.

Some factors can affect the effect of detection, such as capacity, embedding ratio, and dimension of feature vector. Steganalysis experiments show that with the same embedding ratio and file size high capacity lead to high detection precision, with the same capacity and file size, high embedding ratio lead to high detection precision. Higher dimensional feature vector may lead to higher precision, but tradeoff between precision and efficiency requires consideration. Lower dimensional feature vector can also be used as a part of blind detecting vector further.

Rest sections of the paper are arranged as follows: in section2, we present the communication model, briefly survey previously proposed natural language watermarking and steganalysis algorithms; in section3, we describe the framework of synonym substitution algorithms and introduce T-Lex; in section4, we give some notation definitions and introduce IDF; in section5, we describe the details of our scheme, including the weighted suitability, the word election and the detecting algorithm; in section6, we show our experimental results and steganalyze influence of several factors; conclusions and future work are presented in section7.

2. Related work

2.1. Communication model

The communication model for digital watermarking is shown in Fig1. The hider who is usually a publisher holding copyright, embeds watermark into cover text to produce watermarked text. An attacker may be a pirate who wants to remove or modify the watermark, so he corrupts the watermarked text in all possible ways while tries to preserve

the value of the carrier document as much as possible. At last, an extractor who may be a judge try to extract watermark from corrupted text to get copyright information as evidence in the court.

As a steganalyst, we play the role of an attacker to evaluate the security of NLW schemes. In this paper, we focus on detecting watermark and steganalyzing such detection. So watermarked documents won't be changed, no corruption is done.

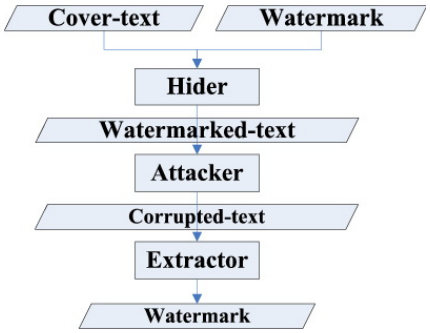

Figure 1. Communication Model for NLW

2.2. Security vulnerability of hiding methods

Depending on how to manipulate the cover text, text watermarking can be divided into two kinds: reformatting watermarking and natural language watermarking.

Reformatting means don't change the content of an article. They only change some format style. For example, shifting a word a bit left means 0 and a bit right means 1; characters in one font mean 0 and in another font mean 1, etc[3][4]. Taking advantage of human vision system (HVS) characteristics, well designed reformatting algorithms will not let a human adversary be conscious of the existence of embedding. But re-typesetting attacks can easily remove the embedded strings.

Depending on how to manipulate the content of documents, natural language watermarking algorithms can be briefly divided into two groups: syntactic hiding and semantic hiding.

Syntactic hiding methods modify the syntactic structure of sentences to embed secrets. Change the voice between active and passive (e.g. "A rogue hit my mage" to "My mage was hit by a rogue" or vice versa); use formal subject to reverse sentence (e.g. "We talked about Ming" to "It is Ming we talked about"); or generally modify syntactic trees of the sentences in cover-text[5]. All these methods are more robust than reformatting ones, but they cost lots of time in parsing sentence structure. Attacks also need base on parsing.

Semantic ways are usually linguistic robust. Synonym substitution schemes which don't need sentence parsing are the most popular and widely used in natural language watermarking[6][7]. A synonym lexicon composed of synonym sets is needed. In each set, words are interchangeable with each other. The simplest cases of synonyms are: words with different spellings, e.g. "unpracticed" and "unpractised"; full name vs abbreviation, e.g. "cuttle" short for "cuttlefish". Words in a synonym set are coded in order to carry bits. Hiding algorithms use the synonym lexicon to decide which and how words are substituted.

A concrete example is given in section3 talking about T-Lex, which is a synonym substitution scheme and is used in our experiment.

2.3. Steganalysis using language model

Cuneyt Taskiran and Umut Topkara et al. had used language models to analyze lexical watermarking systems[8]. Their target hiding scheme in experiment is T-Lex. Trigram models for sentences are trained using SRILM Toolkit, then a five dimensional feature vector is extracted from the output of the language models. Feature vectors of all sentences are passed to a classifier. Their experimental results show 84.9% accuracy on modified sentences and 38.6% on unmodified sentences. Their technique heavily depends on a lexicon.

3. Synonym substitution algorithms

3.1. Framework

In this section, we describe the framework of synonym substitution algorithms. Difficulties in designing replacing manners come from restrictions in natural language, and security is related to distortion of meaning, style, fluency, etc, which are introduced by substitution.

Suppose a synonym substitution algorithm H. It needs four inputs: a cover-text T, a secret s, a lexicon D, and a secret key k. H outputs watermarked text T':

$$H(T, s, D, k) = T'$$

Receiver use corresponding extracting algorithm E to extract watermark s' from T' with the help of D and k:

$$E(T', D, k) = s'$$

A properly designed protocol should ensure that $s' = s$.

Suppose keyword sequence in T is $seq(T) = w_1, w_2 \dots w_n$. After embedding, this sequence become $seq(T') = w'_1, w'_2 \dots w'_n$, w_i is substituted by w'_i or keep unchanged up to its carrying bits. A concrete example algorithm is given below.

3.2. Introduction to T-LEX

Tyrannosaurus Lex is a famous open-source synonym substitution scheme designed by Keith Winstein[9]. He extracted a synonym lexicon of approximate 20 thousand words from WordNet[10]. These words are grouped into disjoint sets, words in a set is interchangeable with each other. The average number of words in a set is 2.56, varying from 2 to 13.

An example of encoding with T-Lex is given in fig2. Suppose the original sentence in the cover-text is ”Her criticizers dismiss her as a country rube with rough features”. In this sentence, word ”rube” is in the synonym lexicon, so it is a keyword. Rube’s synonym sets are {bumpkin, rube, chawbacon, yokel} and these four words are coded to {00, 01, 10, 11} sequentially in the synonym lexicon. If bits to be carried are ”11”, the encoder replaces ”rube” with ”yokel”. When the receiver gets the watermarked-text and sees the encoded sentence ”Her criticizers dismiss her as a country yokel with rough features”, he will find the keyword ”yokel” and its code ”11” by looking up the synonym lexicon. So the secret transmits successfully.

Further details about T-Lex can be found in [9].

$$\text{Her criticizers dismiss her as a country} \left\{ \begin{array}{l} \text{00 bumpkin} \\ \text{01 rube} \\ \text{10 chawbacon} \\ \text{11 yokel} \end{array} \right\} \text{with rough features.}$$

Figure 2. Example of Encoding with T-Lex

4. Preliminaries

4.1. What is IDF

We have mentioned that IDF is used in our scheme to balance common words and rare ones. Here we introduce IDF minutely.

The TF-IDF weight (term frequency-inverse document frequency) is considered as one of the most important concepts in information retrieval. Usually, this weight is used to measure how important a term is to a document. TF is the number of occurrence of specified term divided by the number of occurrence of all terms in one document. IDF was first proposed by Karen Sparck Jones in 1972[11]. If n is the number of documents containing the term, N is the number of all documents in the corpus, IDF of that term should be:

$$w_{\text{idf}} = \ln \frac{N}{n} \tag{1}$$

where ln is the natural logarithm, and keyword's TF is revised as

$$w_{\text{tf-idf}} = w_{\text{tf}} * w_{\text{idf}} \tag{2}$$

TF indicates the importance of a term to a document is proportional to the occurrence of the term normalized by the document length. IDF indicates if the term appears more often in this document than in others. IDF balances common terms and rare ones.

For example, in a 1,000 words document, word "watermarking" appears 3 times, word "system" appears 7 times; in the corpus containing 1,000,000 documents, "watermarking" appears in 1,000 documents, "system" appears in 200,000 documents. If only TF is used, $(watermarking)_{tf} = 0.003$, $(system)_{tf} = 0.007$, seems that "system" is more related than "watermarking" to the core idea of the article, contradict to our intension. After weighted by IDF, $(watermarking)_{tf-idf} = 0.0207$, $(system)_{tf-idf} = 0.0113$. So "watermarking" is more important than "system" to this document, as we wish.

In our scheme, IDF help to adjust word's suitability for the context. Rare words can beat common words in the "word election" according to the IDF-revised results. We will show these in detail in section5.

4.2. Definitions

We present some definitions in order to simplify our description in the following.

Definition 1: Context of a keyword is the set of words neighboring it. Context of a keyword with size of 2n refers to the word set including every word which is not farer than n form the keyword in the article.

For example, in sentence ”synonym sets do not intersect with each other”, if ”intersect” is a keyword, then context of ” intersect” with size 4 is the set {do, not, with, each}.

Definition 2: $CF(w)$ is called the collection frequency of a word w if it represents the number of documents containing this word in the corpus. $CF(S)$ is called the collection frequency of a word set S if it represents the number of documents containing all words in S in the corpus.

Collection frequency reflects customs of natural language usage, which are useful in our detection.

5. Our scheme

5.1. Weighted suitability

In this section, we describe our suitability evaluation, the word election, detecting algorithm and explain the reason why we choose internet instead of traditional corpora.

Neighboring words in the context of a keyword are considered as the most related words to the keyword. If a keyword is suitable for the context, it should co-appear with all the words in the context frequently in the corpus. Oppositely, if a keyword is not suitable for the context, it should seldom co-appear with the words in the context in the corpus.

Suppose two words w_1 and w_2 are evaluating their suitability for the context C, we may meet the following four kinds of cases:

Case 1: we have almost the same chance to meet two words in documents (which means $CF(w_1)$ and $CF(w_2)$ are close in value), then their suitability are mainly determined by which one is more often seen with the context (compare between $CF(w_1, C)$ and $CF(w_2, C)$);

Case 2: we have almost the same chance to meet two words with the same context (which means $CF(w_1, C)$ and $CF(w_2, C)$ are close in value), but one word is common while another is rare (which means $CF(w_1)$ and $CF(w_2)$ span several orders of magnitude). In such a case, rare word and context is more like a collocation than common word and context. So rare word is more suitable;

Case 3: rare word is more often seen with the context than common word, clearly rare one is more suitable;

Case 4: common word is more often seen with the context than rare word, we need a rule to decide which one is more suitable;

Evaluation function for suitability should accord with case1-3 and is a decent balance point in case4. We define the suitability function $ST()$ as follows:

$$ST(w, C) = \ln \frac{N}{CF(w)} * CF(w, C) \quad (3)$$

In equation3, suitability $ST(w, C)$ is the product of collection frequency of (w, C) and IDF of w. $CF(w, C)$ is dominant, which means the more often a word w is seen with a context C,

the more suitable it is for C. And $CF(w)$ also affect the suitability. Sometimes, a word w and a context C co-appear just because w is common instead of w and C is a collocation.

So the commoner a word w is, the smaller its ST value is.

We give two examples here to illustrate our suitability function definition. Suppose the number of documents in the corpus is 19,620,000,000.

Example1: in original text ”which are nearest the capital”, ”nearest” is a keyword, its synonym set is {nearest, nighest}, and context is {which, are, the, capital}. By looking up the corpus, we find that $CF(nearest, C) = 1,290,000$, $CF(nighest, C) = 637$, $CF(nearest) = 84,300,000$, $CF(nighest) = 27,800$, so $ST(nearest, C) = 7,030,400$, $ST(nighest, C) = 8,578$.

Because $CF(nearest, C)$ is more than three orders of magnitude lager than $CF(nighest, C)$, ”nearest” is much more suitable for C than ”nighest”.

Example2: in original text ”whilst corn, cattle and horses”, ”cattle” is a keyword, its synonym set is {cattle, cow, kine, oxen}, and context is {whilst, corn, and, horses}. Let us focus on comparison between ”cattle” and ”cow”. By looking up the corpus, we find that $CF(cattle, C) = 654,000$, $CF(cow, C) = 549,000$, $CF(cattle) = 43,800,000$, $CF(cow) = 92,700,000$, so $ST(cattle, C) = 3,992,455$, $ST(cow, C) = 2,939,860$. In this example, $CF(w_1, C)$ and $CF(w_2, C)$ are close in value, and $CF(w_1)$ and $CF(w_2)$ are the same order of magnitude. IDF helps ”cattle” gain more advantage over ”cow”.

The original sentences above is excerpted from ”Essay on the Nature of Commerce in General” written by Richard Cantillon. All frequencies are gotten from internet using Google.

5.2. Word election

After the ST values have been evaluated, the keyword can begin the game called ”word election”. This election is between the keyword and its synonyms, percentage of vote of each word is in direct proportion to its ST value. We define a VP function to compute the percentage of vote as follows:

$$VP(w, C, S) = \frac{ST(w, C)}{\sum_{w_i \in S} ST(w_i, C)}, \text{ if } S \text{ is the synonym set of } w. \qquad (4)$$

In equation 4, the percentage of vote of a word w in the word election is determined by the ST values of all words in the synonym set. Obviously, the sum of VP values of all words in a synonym set is 1.

5.3. Detecting algorithm

We mainly detect two kinds of distortion introduced by synonym substitution. First, use improper word in the context. Second, rare word improperly replaces common word. In our detecting, occurrence of a keyword with large VP value in the document is legal and positive. Oppositely, occurrence of a keyword with small VP value is doubtable and negative.

Our detecting scheme is described in table1.

Table 1. Detecting algorithm

1. Read from the beginning of a document;
2. Find next keyword w in the document. If meet the end of the document, jump to step5;
3. Find the synonym set S for the keyword w and record context C of w with size of 4 in the document at this occurrence;
4. Compute the VP value of keyword w, append this VP value to a VP value sequence; jump to step2;
5. Compute the expectation and variance of the VP value sequence; these two values compose a two dimensional feature vector of the checked document.
6. Pass the feature vector to a pre-trained SVM classifier and the last decision about watermarked or not is made.

Keyword sequence is extracted from suspicious document along with their context. Every occurrence of any keyword is mapped to a VP value, so the keyword sequence is mapped to a VP value sequence. Expectation and variance of the VP value sequence compose a two dimensional feature vector which is passed to statistical classifier.

SVM (support vector machine) is a set of related supervised learning methods used for classification. Each document is mapped to a feature vector. SVM uses these vectors as input. We first train a SVM classifier, and then use it to judge test documents.

In step3, we set the size of context window as 4. Larger size means more strict for the word to match the context, which lead to significant reduce in $CF(w, C)$; while smaller size means less related for the word to the context. Both direction mean tradeoff between strictness and relativity.

5.4. Limitations of traditional corpora

Before we present our experimental result, we have to explain the reasons why we compute CF function value on internet instead in a traditional corpus.

Achilles' heel of a traditional corpus is the data sparseness problem. No matter how many documents it contains, it is a sparse data set compared to the natural language texts exist in our real life. This disadvantage leads to many issues, one of them is collection frequency of word set which has several members is unpredictable. CF values got from a corpus are meaningless in our detecting.

Another problem may have been ignored for long time. Different documents in one corpus are regarded as same influential. Obviously this is not true-life. Articles in Harry Potter's books are much more influential than other ordinary fantasy novels.

On internet, things are perfectly arranged. More influential documents appear in more web pages, the whole internet is an influential weighted huge corpus.

In our experiments, we check frequency on the internet by Google. Google searches for appropriate web pages for searching keywords. We consider one web page as an independent document. Searching result returns the number of accordant web pages. This number is treated as the collection frequency. Number of pages containing the word ”a” is treated as the total number of documents.

6. Experimental results

6.1. Classifier performance

Detecting algorithm and SVM classifier can be considered as a combined classifier which determines the test document is watermarked or not. Each document is mapped to one element of the set {P, N}, P means positive class label while N means negative class label. The classifier maps documents to predicted classes whose labels are {y, n}.

Given a classifier and a test document, there are four possible outcomes. If the document is positive and is predicted as positive, it is counted as a true positive; if it is predicted as negative, it is counted as a false negative. If the document is negative and is predicted as negative, it is counted as a true negative; if it is predicted as positive, it is counted as a false positive. A two-by-two confusion matrix can be constructed and performance metrics can be derived.

Table 2. Confusion matrix

		True class	
		P	N
Hypothesized Class	y	True Positive	False Positive
	n	False Negative	True Negative

We use three performance metrics in our detection: precision, recall rate and accuracy. They can be computed as follows:

$$precision = \frac{TP}{TP+FP}$$

$$recall = \frac{TP}{P}$$

$$accuracy = \frac{TP+TN}{P+N}$$

In ideally best situation, TP=P and TN=N so that precision, recall rate and accuracy are all 100%. In general situations, these three performance metrics describe the classifier from different aspect with different values.

6.2. Classification by SVM

We choose English articles from academic readings and English literature, including Shakespeare’s drama, Dickens’ novel, and so on. Articles are split to documents of 10K bits in the experiment for normalizing. For watermarking algorithm we choose tyrannosaurus lex

which is open-source and belongs to synonym substitution kind. Secret messages are generated randomly.

Size of context window is 4. We choose libsvm2.84[12] as our SVM classifier. 1080 documents are used, 720 for train and 360 for test. Among train documents and test documents, a third of each set are watermarked.

As show in fig3, VP value sequence varies after embedding. The abscissa denotes sequence number of keywords in the document, the ordinate denotes VP value of keywords. White bars correspond to keywords before embedding, black bars correspond to keywords after embedding. At some indexes, such as 7, 8, 9, white bar and black bar are equal in height, which means embedder didn't replace this keyword. At some indexes, such as 5, 6, 10, 12, high white bar is replaced by low black bar, which means embedder replaced high VP-valued keyword with low VP-valued synonym. Such substitutions are regarded as improper. Also seldom, low white bar is replace by high black bar as at keyword 2, which means detecting algorithm may be mislead. But such misleading situation is really rare. Though keywords 7, 8, 9, 13, 14, 16, 17 remain unchanged in embedding, if they were changed, more suspicion would be accumulated.

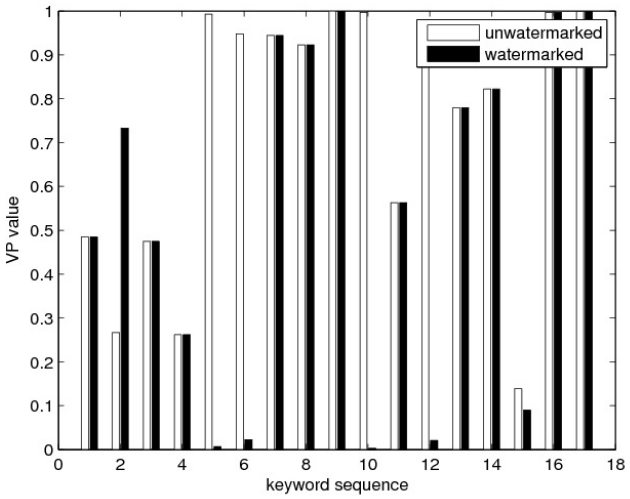

Figure 3. VP value before embedding vs. after embedding

In fig4, distribution of feature vectors of test documents is shown. The abscissa means expectation of VP value sequence of keywords in the document, the ordinate means variance of VP value sequence. White dots stand for feature vectors got from unwatermarked documents, while black stars stand for feature vectors got from watermarked documents. In general, black stars lie at the left part of the space while white dots lie at the right part of the space. This distribution indicates that embedding significantly brings down the expectation of VP value sequence, which means our detection is effective.

As shown in table3, our main experiment show 90.0% accuracy, 86.8% precision and 82.5% recall rate.

Table 3. Confusion matrix in main detection

		True class	
		P	N
Hypothesized Class	y	99	15
	n	21	225

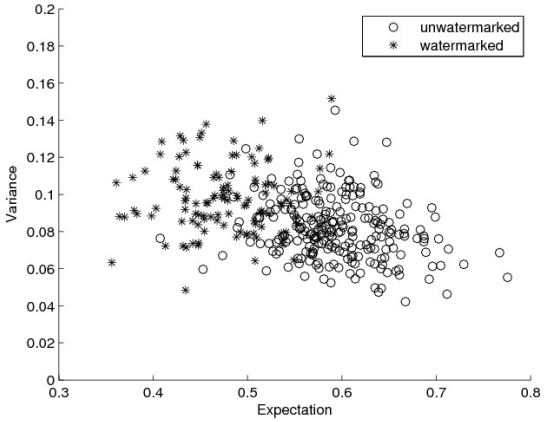

Figure 4. Distribution of feature vectors in test documents

6.3. Influence of capacity

Each document in the main experiment is same in size, but their capacities range between 20 bits to 64 bits. We depart documents in the main experiment into two sets by their capacity, each set have 540 documents. Capacities of documents in one set are below 44 bits and in another set are above 44 bits. Both set are divided by the same proportion as in main experiment, that is 360 (120 watermarked) for train and 180 (60 watermarked) for test. They respectively do the train and test work under SVM.

Table 4. Confusion matrix with low capacity

		True class	
		P	N
Hypothesized Class	y	48	13
	n	12	107

Table 5. Confusion matrix with high capacity

		True class	
		P	N
Hypothesized Class	y	51	5
	n	9	115

Table4 for low capacity show that 86.1% accuracy, 78.7% precision and 80.0% recall rate. Table5 for high capacity show that 92.2% accuracy, 91.1% precision and 85.0% recall rate. Compared to low capacity set, high capacity set exceeds in precision, recall rate and accuracy. High set's performance is even better than main experiment's.

This auxiliary steganalysis experiment show that with the same file size and the same embedding ratio, higher capacity introduce more chance to detect embedded watermark. Obviously, higher capacity introduces more modification to the cover text.

6.4. Influence of embedding ratio

During the experiments mentioned above, T-Lex watermarking embedding ratio is 100%, which means T-Lex use mod1 to encode the cover text, all the keywords participate in hiding. If we reduce embedding ratio to 50%, which means T-Lex use mod2 to encode the cover text, and only half of the keywords participate during hiding, our detecting scheme encounter more failure.

In this auxiliary steganalysis experiment, total number of documents reduces to 675, by the same proportion as in main detection, which means 450 (150 watermarked) for train and 225 (75 watermarked) for test. Table6 shows only 75.1% accuracy.

Lower embedding ratio helps increasing security of watermarking. It is a hard-to-choose tradeoff. Capacity is really precious in a cover text. Wasting capacity dilutes the distortion but increases the size of the cover document.

Table 6. Confusion matrix with 50% embedding ratio

		True class	
		P	N
Hypothesized Class	y	51	5
	n	9	115

6.5. Using only expectation

Dimension of the feature vector is very essential. Though higher dimensional feature vector may lead to higher precision, it also needs more manipulation and longer time. Tradeoff between precision and efficiency requires consideration. Further more, low dimensional feature vector can also be used as a part of a blind detecting vector which is designed for examining watermarked documents without knowing the hiding schemes.

As showed in fig4, watermarked and unwatermarked documents are roughly distributed left and right, which means expectation plays a central role in the two dimensional vector. We redo the main experiment by using one dimensional feature vector that contains only expectation. Table7 shows 87.7% accuracy, 77.9% precision and 88.3% recall rate. Compared to using expectation and variance, we get slightly lower accuracy and precision yet higher recall rate. This is a valuable tradeoff for designing feature vector in blind detection. Moreover, higher recall rate means lower rate of watermarked documents loss, which is significative in many application of NLW.

Table 7. Confusion matrix using only expectation

		True class	
		P	N
Hypothesized Class	y	51	5
	n	9	115

7. Conclusions and future work

In this paper, we steganalyze synonym-substitution based natural language watermarking schemes using context information, and present our experimental results gotten from internet instead of a single corpus. We suggest treat internet as a scale-large, up-to-time, influence-weighted corpus. Experimental results show 90.0% accuracy, 86.8% precision and 82.5% recall rate.

Auxiliary experiments also show that one dimensional vector using only expectation reaches considerable performance, which is very meaningful for designing blind detecting scheme in the future.

We use exactly the same synonym lexicon as in watermarking. There is chance for us to gain a similar synonym lexicon because natural language is shared with everyone. How the lexicon will affect the detection will be included in our future work.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (No. 60773032 & 60703071), the Ph.D. Program Foundation of Ministry of Education of China (No. 20060358014), the Natural Science Foundation of Jiangsu Province of China (No. BK2007060), and the Anhui Provincial Natural Science Foundation (No. 070411043).

References

[1] Z. Duric, M. Jacobs, and S. Jajodia, “Information Hiding: Steganography and Steganalysis,” Data Mining and Data Visualization, 2005.

[2] M. Topkara, G. Riccardi, D. Hakkani-Tuer, and M. Atallah, “Natural language watermarking: challenges in building a practical system,” Proceedings of SPIE, vol. 6072, pp. 106–117, 2006.

[3] T. Liu and W. Tsai, “A New Steganographic Method for Data Hiding in Microsoft Word Documents by a Change Tracking Technique,” Information Forensics and Security, IEEE Transactions on, vol. 2, no. 1, pp. 24–30, 2007.

[4] C. Chao, W. Shuozhong, and Z. Xinpeng, “Information Hiding in Text Using Typesetting Tools with Stego-Encoding,” Proceedings of the First International Conference on Innovative Computing, Information and Control-Volume 1, pp. 459–462, 2006.

[5] M. Atallah, V. Raskin, M. Crogan, C. Hempelmann, F. Kerschbaum, D. Mohamed, and S. Naik, “Natural Language Watermarking: Design, Analysis, and a Proof-of-Concept Implementation,” Information Hiding, 4th International Workshop, IHW, pp. 25–27, 2001.

[6] U. Topkara, M. Topkara, and M. Atallah, “The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions,” Proceedings of the 8th workshop on Multimedia and security, pp. 164–174, 2006.

[7] Y. Chiang, L. Chang, W. Hsieh, and W. Chen, “Natural language watermarking using semantic substitution for chinese text,” Digital Watermarking: Second International Workshop, IWDW, vol. 2939, pp. 129–140, 2003.

[8] C. Taskiran, U. Topkara, M. Topkara, and E. Delp, “Attacks on lexical natural language steganography systems,” Proceedings of SPIE, vol. 6072, pp. 97–105, 2006.

[9] “The tyrannousaurus lex system available at http://alumni.imsa.edu/ keithw/tlex/,”

[10] C. Fellbaum et al., WordNet: an electronic lexical database. Cambridge, Mass: MIT Press, 1998.

[11] K. Jones et al., “A statistical interpretation of term specificity and its application in retrieval,” Journal of Documentation, vol. 28, no. 1, pp. 11–21, 1972.

[12] C. Chang and C. Lin, “LIBSVM: a library for support vector machines,” Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm, vol. 80, pp. 604–611, 2001.