Adjective Deletion for Linguistic Steganography and Secret Sharing

Ching – Yun Chang[1] *Stephen Clark*[1]
(1) University of Cambridge, Computer Laboratory, 15 JJ Thomson Avenue, Cambridge, UK
Ching-Yun.Chang@cl.cam.ac.uk, Stephen.Clark@cl.cam.ac.uk

ABSTRACT
This paper describes two methods for checking the acceptability of adjective deletion in noun phrases. The first method uses the Google n-gram corpus to check the fluency of the remaining context after an adjective is removed. The second method trains an SVM model using n-gram counts and other measures to classify deletable and undeletable adjectives in context. Both methods are evaluated against human judgements of sentence naturalness. The application motivating our interest in adjective deletion is data hiding, in particular linguistic steganography. We demonstrate the proposed adjective deletion technique can be integrated into an existing stegosystem, and in addition we propose a novel secret sharing scheme based on adjective deletion.

KEYWORDS: linguistic steganography, adjective deletion.

Proceedings of COLING 2012: Technical Papers, pages 493–510,
COLING 2012, Mumbai, December 2012.

1 Introduction

Linguistic steganography is a form of covert communication in which information is embedded in a seemly innocent cover text so that the presence of the information is imperceptible to an outside observer (human or computer) (Fridrich, 2009). An ideal linguistic stegosystem should fulfil two fundamental requirements: *high imperceptibility* and *high payload capacity*. The former aims at imposing minimum embedding distortion to the cover text so that the resulting stegotext in which a message is camouflaged is inconspicuous. The latter aims at providing sufficient embedding capacity in order to achieve efficient information transmission. There is a trade-off between imperceptibility and payload, since any attempt to embed additional information via changes to the cover text increases the chance of introducing anomalies into the text and thus raising the suspicion of an observer (Chang and Clark, 2010a).

Another cryptographic method is secret sharing. Secret sharing (Blakley, 1979; Shamir, 1979) refers to methods for distributing a secret amongst a group of n people, each of whom is allocated a *share* of the secret. Individual shares are of no use on their own; only when any group of t (for *threshold*) or more shares are combined together can the secret be reconstructed. Such a system is called a (t, n)-threshold scheme. For example, a simple (3,3)-threshold scheme for a secret number s can be achieved by splitting s into three numerical shares s_1, s_2 and s_3 such that $s = s_1 + s_2 + s_3$. Note that there is no way to recover the secret number by only using one or two of the shares; all shares are required for effective recovery.

There are some proposed (t, n)-threshold schemes where $t \neq n$. For example, Shamir's scheme (Shamir, 1979) allows that any t out of n shares may be used to recover the secret. This scheme relies on the idea that it takes t points to define a polynomial of degree t-1 (e.g. it takes two points to define a straight line, three points to define a quadratic, four points to define a cubic curve). The method first randomly creates a polynomial of degree t-1 with the secret number as the first coefficient. Next each of the n people is given a distinct point on the curve. Therefore, any t out of the n people can fit a (t-1)th degree polynomial using their points, where the first coefficient is the secret. For example, any three of the five points (1, 1494), (2, 1942), (3, 2578), (4, 3402) and (5, 4414) can fit the polynomial of degree two $f(x) = 1234 + 166x + 94x^2$ and reveal the secret as 1234.[1] From the above two secret sharing schemes we can see that the share can be in different forms, such as numbers and points, depending on the methods used.

In this paper, we propose a novel (2, 2)-threshold secret sharing method where the shares are presented as two comparable texts, as explained in Section 8. The proposed method exploits the adjective deletion technique to embed secret bitstrings of 0s and 1s in two texts. These two texts can then be combined to reveal the secret bitstring; but neither text by itself can reveal the bitstring. Hence the proposed method is a novel combination of secret sharing and linguistic steganography. In addition, we demonstrate the adjective deletion technique can be integrated into an existing linguistic stegosystem (Chang and Clark, 2010a).

We have identified adjectives as a potentially large source of deletable words, in the sense that adjectives can often be removed without significantly affecting the meaning or naturalness of the resulting text. For example, "he spent only his *own* money" and "he spent only his money" express the same meaning. In the extreme case, there are adjective-noun pairs in which the adjective is somewhat redundant, for example *unfair prejudice*, *horrible crime* and *fragile glass*.

We explore the identification of redundant adjectives in context for the applications of linguistic

[1] http://en.wikipedia.org/wiki/Shamir's_Secret_Sharing.

steganography and secret sharing. In order to generate unsuspicious stegotext and textual shares after adjective deletion, we propose two checking methods using the Google n-gram corpus and an SVM to certify the naturalness of the generated sentences. The methods are evaluated using human judgements of naturalness. Note that the evaluation is based on the sentence-level naturalness rather than the coherence of the whole document. Modeling the document-level coherence of modified text would be useful but is outside the scope of our study. The resulting precision can be seen as an indirect measure of the imperceptibility of the stegosystem since quality deletions are less likely to be seen as suspicious by the observer, whereas the recall can be seen as an indirect measure of the payload since deletable adjectives are detected where possible and therefore as much information as possible is embedded.

There are various practical security issues in the application of linguistic steganography and secret sharing systems that we have chosen to ignore or simplify in order to focus on the underlying NLP technology. For example, we assume the adversary is a human acting passively rather than actively. In other words, we have ignored the possibility of computational steganalysis and steganographic attacks, such as detecting, extracting and destroying the hidden message (Fridrich, 2009).

2 Related Work

2.1 Linguistic Transformations for Steganography

Existing studies have exploited different linguistic transformations for the application of steganography, such as lexical substitution (Chapman and Davida, 1997; Bolshakov, 2004; Taskiran et al., 2006; Topkara et al., 2006c; Chang and Clark, 2010b), phrase paraphrasing (Chang and Clark, 2010a), sentence structure manipulations (Atallah et al., 2001a,b; Liu et al., 2005; Meral et al., 2007; Murphy, 2001; Murphy and Vogel, 2007b; Topkara et al., 2006b) and semantic transformations (Atallah et al., 2002; Vybornova and Macq, 2007). For details of the transformations mentioned above, readers can refer to our previous papers: Chang and Clark (2010a) and Chang and Clark (2010b).

Another group of studies aim to embed information into translated text. Stutsman et al. (2006) use multiple translation systems to provide alternative candidates for a sentence. The secret information is then embedded into the choice of translation. Another recent work proposed by Venugopal et al. (2011) introduces a watermark as a parameter in the machine translation algorithm and probabilistically identifies the watermarked translation. The motivation of watermarking machine translation outputs is to distinguish machine and human generated translations so a machine translation system is unlikely to learn from self-generated data.

These transformations often rely on sophisticated NLP tools and resources. For example, a lexical substitution-based stegosystem may require synonym dictionaries, POS taggers, word sense disambiguation tools and language models; a syntactic transformation-based stegosystem may require syntactic or semantic parsers and language generation tools. However, given the current state-of-the-art, such NLP techniques cannot guarantee the transformation's imperceptibility. Hence it is important to evaluate the security level of a stegosystem.

2.2 Stegosystem Evaluations

A stegosystem can be evaluated from two aspects: the security level and the embedding capacity. The security assessment methods used so far can be classified into two categories: automatic evaluation and human evaluation. Topkara et al. (2006b) and Topkara et al. (2006a) used

machine translation evaluation metrics BLEU and NIST, automatically measuring how close a stego sentence is to the original. Topkara et al. (2006b) admitted that machine translation evaluation metrics are not sufficient for evaluating stegosystems; for example, BLEU relies on word sequences in the stego sentence matching those in the cover sentence and thus is not suitable for evaluating transformations that change the word order significantly.

The other widely adopted evaluation method is based on human judgements. Meral et al. (2007), Kim (2008), Kim (2009) and Meral et al. (2009) asked subjects to edit stegotext for improving intelligibility and style. The fewer edit-hits a transformed text received, the higher the reported security level. Murphy and Vogel (2007b) and Murphy and Vogel (2007a) first asked subjects to rate the acceptability (in terms of plausibility, grammaticality and style) of the stego sentences on a seven-point scale. Then subjects were provided with the originals and asked to judge to what extent meaning was preserved on a seven-point scale. Chang and Clark (2010a) asked subjects to judge whether a paraphrased sentence is grammatical and whether the paraphrasing retains the meaning of the original.

The other aspect of the stegosystem evaluation is to calculate the amount of data capable of being embedded in a stego text, which can be quantified in terms of bits per language unit, for example per word or per sentence. Payload measurements can be theoretical or empirical. The theoretical payload measurement only depends on an encoding method and is independent of the quality of a stego text; whereas the empirical measurement takes the applicability of a linguistic transformation, namely the security of a stego text, into consideration and measures the payload capacity while a certain security level is achieved. Most of the payload rates reported in existing work are based on empirical measurements, with typical payload rates between 0.5 and 1.0 bits per sentence.

Not only the linguistic transformation and the encoding method, but also the choice of cover text can affect the security level and the payload capacity of a stegosystem. For example, if a newspaper article were chosen as the cover text, then any changes could be easily found in practice by comparing the stego text with the original article, which is likely to be readily available. In addition, an anomaly introduced by a linguistic transformation may be more noticeable in a newspaper article than in a blog article. In terms of payload capacity, a synonym substitution-based stegosystem may find more words that can be substituted in a fairy tale than in a medical paper since there are usually many terminologies in a medical paper which cannot be changed or even cannot be found in a standard dictionary. To the best of our knowledge, there is no study on the practical issue of using different types of cover text for the steganography application.

2.3 Sentence Compression

Sentence compression, text simplification and text summarisation usually involve removing unimportant words in a sentence in order to make the text more concise. For example, Knight and Marcu (2002), Cohn and Lapata (2008), Filippova and Strube (2008) and Zhu et al. (2010) have used the word deletion operation in their systems. However, to our knowledge, there is no work looking at redundant adjectives in text in particular. The proposed adjective deletion methods can be applied before and/or after a sentence compression system. Deleting unnecessary adjectives before can help the system focus on other content of a sentence. Deleting unnecessary adjectives after can generate an even more concise sentence.

Sentence	Those awaiting execution spent their *last* days alone .
Supertags before deletion	NP[nb]/N N/N N (S[dcl]\NP)/NP NP[nb]/N *N/N* N NP\NP .
Supertags after deletion	NP[nb]/N N/N N (S[dcl]\NP)/NP NP[nb]/N N N NP\NP .
Sentence	We met in UK *last* time .
Supertags before deletion	NP S[dcl]\NP ((S\NP)\(S\NP))/NP N *((S\NP)\(S\NP))/((S\NP)\(S\NP))* (S\NP)\(S\NP) .
Supertags after deletion	NP S[dcl]\NP ((S\NP)\(S\NP))/NP N/N N .

Table 1: Comparing supertags before and after adjective deletion

3 Deletable Adjective Classification

In order for an adjective deletion to be acceptable according to our method, we use two checks: grammaticality and naturalness checks. In order to prevent an ungrammatical adjective deletion, we use the syntactic filter proposed in Chang and Clark (2010a) to certify the deletion grammaticality. This is only a preliminary grammaticality check and does not guarantee sentence fluency. For generating the modified sentence, we also use Minnen et al. (2001)'s tools for correcting the form of an indefinite. For example, after deleting *alternative*, the phrase "**an** alternative choice" would be modified to "**a** choice". The original and modified sentences are then parsed using a wide-coverage CCG parser (Clark and Curran, 2007). After parsing, each lexical token is associated with a syntactic description, called a lexical category, or supertag. With the significant amount of information included in supertags, comparing two sequences of supertags is similar to comparing two syntax trees. Thus we require a deletion to retain the same sequence of supertags as that of the original sentence in order to ensure grammaticality. Table 1 shows two adjective deletion examples and their supertags,[2] where *last* is the target adjective. The first deletion case passes the grammaticality check since all the supertags remain the same after deleting *last*; while in the second example, both *UK* and *time*'s supertags are changed after the deletion and thus, this deletion fails the check. Note that all the experiment data used in this paper pass the syntax check.

3.1 N-gram Count Method

Inspired by Chang and Clark (2010b), which used the Google n-gram corpus to check the applicability of a synonym in context based on Bergsma et al. (2009), we use a similar method to calculate a score based on the n-gram counts before and after a potential deletion, as demonstrated in Table 2. The Google n-gram corpus[3] is a large publicly available collection of bi-grams to five-grams generated from approximately 1 trillion tokens of online text. Only n-grams appearing more than 40 times are kept in the corpus.

For the example in Table 2 we first extract contextual bi- to five-grams containing the target adjective *alternative* as well as that across the target position with *alternative* removed. The Google n-gram corpus is then consulted to obtain their frequency counts. We sum up all the logarithmic counts[4] for the original and modified cases. The reason for using the logarithm count is that lower-order n-grams usually have much larger counts than higher-order n-grams so taking the logarithm may prevent the sum being dominated by lower-order n-gram counts. Since before the deletion there are more n-grams extracted, we divide the sum by the number

[2]There is a parse error in the first sentence, but it does not affect the supertag comparison.
[3]Available from the LDC as LDC2006T13.
[4]log(0) and division by zero are taken to be zero.

There is always an *alternative* choice in a mental situation.	
N-grams before the deletion (log *freq*)	N-grams after the deletion (log *freq*)
an *alternative* (15.5)	a choice (15.2)
alternative choice (9.8)	always a choice (8.8)
always an *alternative* (7.9)	a choice in (11.3)
an *alternative* choice (9)	is always a choice (8.3)
alternative choice in (6.2)	always a choice in (5.5)
is always an *alternative* (7.4)	a choice in a (7.6)
always an *alternative* choice (0)	There is always a choice (7)
an *alternative* choice in (5.5)	is always a choice in (4.3)
alternative choice in a (0)	always a choice in a (0)
There is always an *alternative* (6)	a choice in a mental (0)
is always an *alternative* choice (0)	
always an *alternative* choice in (0)	
an *alternative* choice in a (0)	
alternative choice in a mental (0)	
$Count_{average}^{Before} = 4.8$	$Count_{average}^{After} = 6.8$
$Score = \frac{Count_{average}^{After}}{Count_{average}^{Before}} = 1.4$	

Table 2: An example of the Google n-gram count method

of extracted n-grams and call the derived average value the $Count_{average}$. Finally, we use a *Score* function which is equal to $\frac{Count_{average}^{After}}{Count_{average}^{Before}}$ to measure how much the $Count_{average}^{Before}$ changes after deleting the target word *alternative*. In this example the *Score* for deleting *alternative* in this context is equal to 1.4 and will be determined as acceptable by a threshold with value below 1.4.

3.2 SVM Method

Since some n-grams may be more informative than others when deciding whether an adjective can be deleted, we combine the n-gram counts and other measures described in Section 4 to train a classifier. We use the LIBSVM (Chang and Lin, 2011) implementation of support vector machines (SVMs) for classification. As suggested by Hsu et al. (2010), we scale feature values to the range [-1, +1], and use the default radial basis function (RBF) kernel. The two parameters of the RBF kernel, C and γ, are determined by using the model selection tool grid.py provided from LIBSVM. After the best (C, γ) is found using the training data, the whole training set is used again to train the final classifier. In order to observe the trade-off between precision and recall, we use the probability estimate feature in LIBSVM and train the SVM model to output probabilities so users can decide the security level by varying a probability threshold.

4 Features for the SVM

4.1 N-gram Counts

The first set of features consists of logarithmic contextual bi- to five-gram counts. Before the deletion, there are 14 contextual n-grams; after the deletion, there are 10 contextual n-grams as shown in Table 2. If a contextual window is not available, for example if a window spans beyond

the current sentence, the n-gram count is set to zero. For each contextual window we provide an additional boolean feature to indicate whether a window is available. The second set of features consists of 5 score values. The first score is the *Score* function described in Section 3.1. The second to the fifth scores are the scores calculated by only considering a specific window size n, where $n = 2$ to 5, using the same method as for the *Score* function. Again, each score is provided with an additional boolean feature to indicate whether the $Count^{Before}_{average}$ is equal to zero. There are a total of 58 features contributed from the n-gram counts.

4.2 Lexical Association Measures

In addition to n-gram features, we exploit some standard lexical association measures to determine the degree of association between an adjective and a noun. Pointwise Mutual Information (PMI) (Church and Hanks, 1990) is roughly a measure of how much one word tells us about the other. In order to calculate PMI, we need the joint frequency of the noun-adjective pair, the frequency of the noun modified by any adjective and the frequency of the adjective modifying any noun.

We collect (adjective, noun) pairs and their frequency counts from grammatical relations (GRs). The GRs we use are derived by parsing a Wikipedia dump (dated October 2007) with Clark and Curran (2007)'s CCG parser. We first consider GRs having the pattern (ncmod _ *noun adjective*) and extract the (*adjective*, *noun*) pair. Next we extract pairs that match patterns (xcomp _ be *adjective*) and (ncsubj be *noun* _) in a sentence. For instance, the GRs of the sentence "The car is red" are (det car_1 the_0) (xcomp _ be_2 red_3) and (ncsubj be_2 car_1 _), and since *car* and *red* match the two patterns, (*red*, *car*) is seen as an eligible pair for our database. A total of 63,896,006 adjective-noun pairs are extracted form the parsed Wikipedia corpus which includes 832,320 noun types and 792,914 adjective types.

We also use the log likelihood ratio (LLR), an alternative to PMI, which is reported to handle rare events better (Dunning, 1993). Again, the contingency table for computing LLR can be derived from the parsed Wikipedia corpus described above. In the study of collocation extraction, both high PMI and LLR values are treated as evidence that the collocation components occur together more often than by chance. In this paper, we use PMI and LLR as features in the SVM.

4.3 Noun and Adjective Entropy

Suppose we observe a noun N_1 as being modified by adjective J_1 five times, J_2 twice and J_3 three times. The modifier entropy of N_1 is $H(N_1) = -((0.5\,log\,0.5) + (0.2\,log\,0.2) + (0.3\,log\,0.3)) = 1.5$. Now suppose there is a noun N_2 modified by J_4 nine times and J_5 once. The modifier entropy of N_2 is $H(N_2) = -((0.9\,log\,0.9) + (0.1\,log\,0.1)) = 0.5$. Thus we can conclude that the modifier of N_1 is more unpredictable than that of N_2. Similarly, we calculate an adjective's argument entropy based on the entropy of the noun given a fixed adjective.

We also observe the modification frequency of a noun using the parsed Wikipedia corpus. From the corpus, we obtain the frequency of a noun being modified by any adjective (mod_{adj}), the frequency of a noun being modified by something other than an adjective (mod_{other}), and the frequency of a noun not being modified at all (mod_{non}). The modification entropy of a noun is defined as: $M(N) = -(p(mod_{adj})\,log\,p(mod_{adj}) + p(mod_{non})\,log\,p(mod_{non}))$. Note that $p(mod_{other})$ is not included in the definition of $M(N)$ since we want to focus on the adjectival modification of a noun. Modifier entropy, argument entropy, modification entropy plus the modification probabilities $p(mod_{adj})$, $p(mod_{other})$ and $p(mod_{non})$ are used as SVM features.

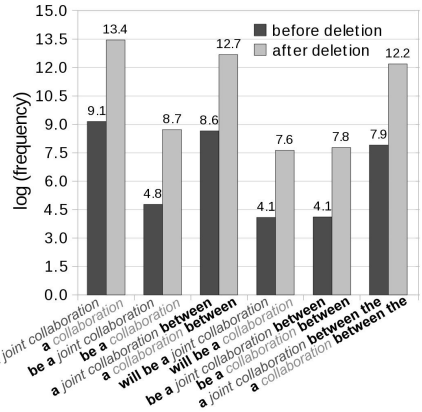


Figure 1: N-gram count distributions before and after deleting *joint*

4.4 Contextual α-Skew Divergence

We assume that if an adjective in a noun phrase is deletable, the noun should have a similar n-gram distribution to the original adjective-noun phrase across the various n-gram counts. Figure 1 shows the logarithmic n-gram counts of *joint collaboration* and *collaboration* being in the same context of the sentence "The task force will be a *joint collaboration* between the cities of Sterling Heights and Warren." In this example sentence, *joint* is determined as deletable. We can see that the counts have similar distributions before and after the deletion.

We use α-skew divergence (Lee, 1999) to calculate the n-gram distributional similarity between the original and the modified sentences. The α-skew divergence is a non-symmetric measure of the difference between two probability distributions P and Q. In our application, P is a probability vector containing normalised logarithmic counts derived from the contextual n-grams before removing the adjective, and Q is a probability vector obtained after deleting the adjective. The α-skew divergence measure is defined as:

$$S_\alpha(Q,P) = D(P\|\alpha{\cdot}Q + (1-\alpha){\cdot}P),$$

where $0 \leq \alpha \leq 1$ and D is the Kullback-Leibler divergence $D(P\|Q) = \sum_v P(v) log \frac{P(v)}{Q(v)}$. The α parameter is for avoiding the problem of zero probabilities, and in our system we use α=0.99. Under our assumption, a deletable adjective would have a smaller effect on the n-gram count distribution after deletion than an undeletable adjective and, therefore, a deletable adjective would have a smaller divergence value.

5 Data

5.1 Pilot Study Data

We first created a small dataset for a preliminary study. In order to experiment with redundant adjectives, we collected 90 sentences from the Internet, each of which contained an adjective-

noun pleonasm.[5] A pleonasm consists of two concepts (usually two words) that are mutually redundant: examples are *free gift*, *cold ice* or *final end*. In other words, pleonasms contain unnecessary words, and those words can be removed without affecting the meaning of the text.

Apart from positive data (deletable adjectives), we also need some negative data (undeletable adjectives) to test whether the n-gram count method and the SVM model have the ability to filter out bad deletions. We define an adjective as undeletable if the removal of an adjective in a noun phrase significantly affects the naturalness of the resulting sentence. The second author of this paper manually selected 76 undeletable cases from the British National Corpus (BNC) as the negative data.

Adjectives in pleonasms can be seen as extreme redundancies in text, and removing those redundancies would not reduce the level of security in terms of steganography. However, pleonasms are not general enough so might not be found frequently in text, which diminishes the amount of secret information which can be embedded in the text. Thus we collect more positive and negative data which are more frequent in text for training, developing and testing the n-gram count method and the SVM model. This additional set serves as our main data source (described in Section 5.2), with the pleonasm set serving as a useful pilot study.

5.2 Human Annotated Data

In order to have a labelled dataset for training and testing a classifier, we asked 30 native English speakers to judge whether the removal of an adjective in a noun phrase significantly affects the naturalness of the resulting sentence. The guideline is the same as that used for the pilot study data. Note that we only care about the naturalness of the resulting sentence rather than the meaning retention of the original sentence. In other words, the evaluation is based on the sentence-level naturalness rather than the coherence of the whole document. Table 3 shows the six examples that were used as part of the annotator instructions.

The sentences for creating the data were randomly selected from section A of the British National Corpus (BNC) with the constraint that each passed the syntax check as described in Section 3. We collected 1200 sentences, each of which contains one marked adjective to be annotated. In order to measure the inter-annotator agreement, 300 of the 1200 sentences were assessed by 3 different judges; the others were labelled only once. We calculated the inter-annotator agreement using Fleiss' kappa (Fleiss et al., 2003) scored between 0 and 1. Fleiss' kappa works for any fixed number of annotators and allows different items rated by different individuals. For the 300 multi-judged instances, the Fleiss' kappa is 0.49, which can be interpreted as *moderate agreement* according to Landis and Koch (1977).

The 300 multi-judged instances were labelled using the majority rule and were treated as the test set; the other 900 instances were randomly split into a 700-instance training set and a 200-instance development set. The ratio of the number of deletable adjectives to the number of undeletable adjectives is around 2:1 for all the datasets.

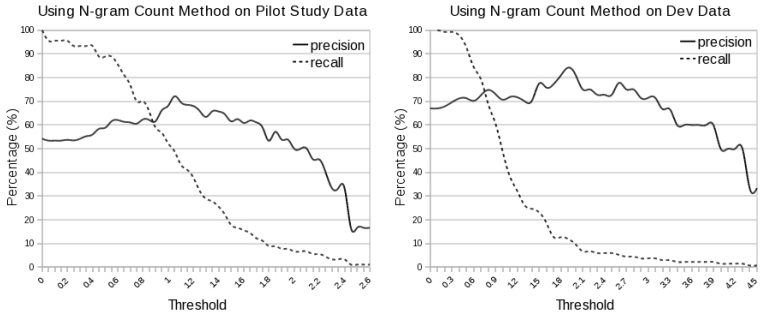
6 Experiments and Results

The performance on this adjective deletion task is measured in precision and recall on the positive deletable cases. From a steganography aspect, accuracy is not useful, while the trade-off between precision and recall is more relevant. A precision baseline is obtained by always saying

[5] A collection of pleonasms can be found at http://www.pleonasms.com.

Judgement	Example sentence
Deletable	He was putting on his *heavy* overcoat, asked again casually if he could have a look at the glass.
Deletable	We are seeking to find out what *local* people want, because they must own the work themselves.
Deletable	We are just at the beginning of the *worldwide* epidemic and the situation is still very unstable.
Undeletable	He asserted that a modern artist should be in tune with his times, careful to avoid *hackneyed* subjects.
Undeletable	With various groups suggesting police complicity in township violence, many blacks will find *little* security in a larger police force.
Undeletable	There can be *little* doubt that such examples represent the tip of an iceberg.

Table 3: Judgement examples given to annotators



(a) Results on the pilot study data

(b) Results on the development data

Figure 2: Performance of the n-gram count method

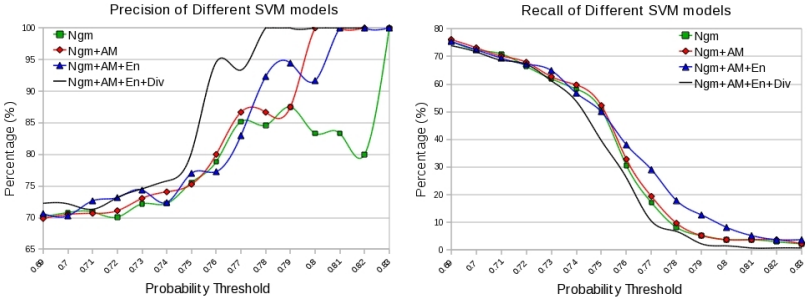
an adjective is deletable. The precision baselines in the pilot study data, development data and test data are 54.2%, 67.0% and 64.0%, respectively.

6.1 Experiments Using N-gram Count Method

We test the n-gram count method on the pilot study data and the development data. Figure 2(a) and Figure 2(b) show the precision and recall curves with respect to different thresholds for the pilot study data and the development data, respectively. For the pilot study data, the best precision 72.1% is achieved with a 48.9% recall by using a threshold equal to 1.05. For the development data, the best precision 84.2% is achieved using a threshold equal to 1.9. However, the recall value drops to 11.9% which means many deletable adjectives are being ignored.

6.2 Experiments using SVM

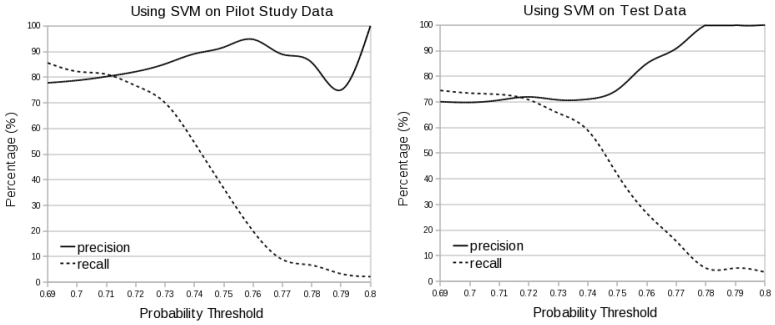
For the SVM learning approach, we first train models with different features and test the models on the development data. Figure 3(a) and Figure 3(b) show the precision and recall curves of the models with probability thresholds greater than 0.69 and lower than 0.83 (since these



(a) Precision curves of the models

(b) Recall curves of the models

Figure 3: Performance of SVM models using different features



(a) Results on the pilot study data

(b) Results on the test data

Figure 4: Performance of the Ngm+AM+En+Div model

values result in a reasonable precision range). In addition, we ignore results that have recall values lower than 10% even though a high precision is achieved. The SVM Ngm model is trained using the 58 features described in Section 4.1. Its best precision is 85.2% (with a recall greater than 10%) which is similar to that achieved by using the n-gram count method, but the corresponding recall is slightly improved to 17.2%. Next we add two association measures MI and LLR to the features and train the model Ngm+AM. The best precision of the Ngm+AM model is 86.7% and the corresponding recall is 19.4%. We then add features by including entropies and modification probabilities described in Section 4.3 and train the Ngm+AM+En model. This model achieves 92.3% precision with 17.9% recall. Finally, the Ngm+AM+En+Div model is trained with the divergence measure added to the features. The best precision of this model is 94.6% with 26.1% recall when the probability threshold 0.76 is used. Since the Ngm+AM+En+Div model achieves the best precision value among all the models, we further evaluate this model using the pilot study dataset and the test dataset.

Figure 4(a) shows the performance of the Ngm+AM+En+Div model on the pilot study data.

With 50% recall on the pilot study data, the SVM model achieves a precision of 90%, while the n-gram method only achieves 72.1% precision at this level of recall. We can see that there is a large improvement on classifying deletable adjectives from undeletable adjectives in the pilot study data compared to both the baseline and the n-gram count method. Finally, we use the probability threshold 0.76 that gives the best precision on the development set to evaluate the pilot study data and the test data. For the pilot study data, the classifier achieves a precision of 94.7% and a recall of 20%; for the test data, the classifier achieves a precision of 85% and a recall of 26.6%. Note that a precision of 100% is not necessarily required because the inter-annotator agreement on the collected human judgements is not 100% and therefore it is not clear whether the precision upper bound on this task is 100%.

Figure 4(b) shows the full range of precision-recall scores using different probability threshold values on the test data.[6] From this figure, we can clearly see the trade-off between precision and recall, which corresponds to the trade-off between imperceptibility and payload for the linguistic steganography application. In practice, steganography users can decide the threshold according to their requirements on the security level and embedding capacity. In addition, since the cover text can be selected by users, the payload can be improved by choosing a text containing more adjectives such as fictions or fairy tales, which might increase the density of deletable adjectives in text.

7 Linguistic Steganography Application

For linguistic steganography, there exists a convenient modularity between the linguistic transformation and the embedding method. In other words, the utility of a specific embedding method does not imply a particular linguistic transformation, although it will put some constraints on what transformation can be used. For example, synonym substitution, paraphrasing and translation can be applied to an embedding method which reconstructs the secret message as concatenating codewords that are directly associated with a choice. We will demonstrate that the adjective deletion technique can be integrated into our earlier Chang and Clark (2010a) secret embedding scheme.

In Chang and Clark (2010a) we proposed a secret embedding method based on text paraphrasing as shown in Figure 5. In the secret embedding phase, a cover text is first divided into embedding units of which each has an equal number of sentences and contains at least one paraphrasable sentence. In this example, the paraphrasable sentences are t_1, t_3, t_4, t_7 and t_8; the text can be divided into three embedding units u_1, u_2 and u_3 with the size equal to three sentences. One secret bit is then embedded in one embedding unit. If the secret bit is 0, all the paraphrasable sentences in the embedding unit are transformed into non-paraphrasable sentences; if the secret bit is 1, the embedding unit remains unchanged. The secret bitstring in this example is 101 so the paraphrasable sentence t_4 in u_2 is transformed into a non-paraphrasable sentence, and u_1 and u_3 are unmodified. The secret extracting can be easily performed by dividing the stego text into embedding units and using the existence of paraphrasable sentences to decide whether the embedding unit represents secret bit 0 or 1. In this embedding scheme, the embedding unit size is treated as the secret key that is only shared between the sender and the receiver.

We can replace the text paraphrasing in the above method with the adjective deletion technique as the linguistic transformation, so that each embedding unit contains at least one deletable

[6] Note that we are not optimising for one single score on the test set, e.g. F-score, but showing the full range of the precision-recall tradeoff that corresponds to a security-payload tradeoff in the steganography setting.

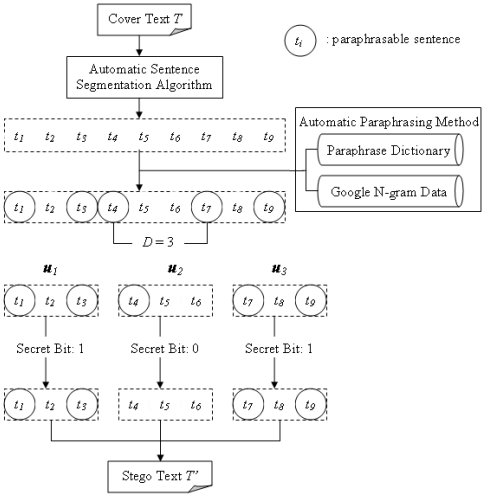


Figure 5: The Chang and Clark (2010a) stegosystem

adjective. If we want to embed 0, all the deletable adjectives will be removed from the embedding unit. Since the deletable adjectives are checked by the proposed model, the removal of the adjectives should achieve a certain level of naturalness in the sentences. If we want to embed 1, the embedding unit will not be modified. It is important to note that the recovery of the secret bitstring does not require the original text. The receiver only needs the secret key to define the size of an embedding unit and the adjective checking model to see whether there is a deletable adjective in an embedding unit.

8 Secret Sharing Scheme

We propose a novel secret sharing scheme based on the adjective deletion technique and text alignment. The secret sharing scheme converts a secret bitstring into two shares, $Share_0$ and $Share_1$, that are camouflaged in the form of natural language text. $Share_0$ holds secret bits as 0s and $Share_1$ holds secret bits as 1s. The order of the 0s and 1s can only be reconstructed by aligning the two texts.

Figure 6 illustrates an example of the secret sharing scheme. The secret bitstring is 101. We first give $Share_0$ and $Share_1$ the same text and use the proposed adjective checking method to determine deletable adjectives in the text. In this example, the n-gram count method with threshold equal to 1 is applied. The adjectives passing the check are *mysterious*, *terrible* and *single*, and one deletable adjective will embed a secret bit. The embedding rule is: to embed a secret bit 0/1, the target adjective is kept in the share that holds 0s/1s, and is removed from the other share. For example, the first secret bit is 1 so *mysterious* is kept in $Share_1$ and is deleted from $Share_0$. Next, we embed the second secret bit 0 by keeping *terrible* in $Share_0$ and removing it from $Share_1$. The third secret bit is 1 so we keep *single* in $Share_1$ and remove it from $Share_0$. Now the secret bitstring 101 is converted into two meaningful texts. The reconstruction of the

Secret bits: 101	Text: "Have you heard of the **mysterious** death of your late boarder Mr. Enoch J. Drebber, of Cleveland?" A **terrible** change came over the woman's face as I asked the question. It was some seconds before she could get out the **single** word "Yes" – and when it did come it was in a husky, unnatural tone.
Embed 1st bit: 1 Target adj: *mysterious*	$Share_0$: "Have you heard of the death of your late boarder Mr. Enoch J. Drebber, of Cleveland?" A **terrible** change came over the woman's face as I asked the question. It was some seconds before she could get out the **single** word "Yes" – and when it did come it was in a husky, unnatural tone. $Share_1$: "Have you heard of the **mysterious** death of your late boarder Mr. Enoch J. Drebber, of Cleveland?" A **terrible** change came over the woman's face as I asked the question. It was some seconds before she could get out the **single** word "Yes" – and when it did come it was in a husky, unnatural tone.
Embed 2nd bit: 0 Target adj: *terrible*	$Share_0$: "Have you heard of the death of your late boarder Mr. Enoch J. Drebber, of Cleveland?" A **terrible** change came over the woman's face as I asked the question. It was some seconds before she could get out the **single** word "Yes" – and when it did come it was in a husky, unnatural tone. $Share_1$: "Have you heard of the **mysterious** death of your late boarder Mr. Enoch J. Drebber, of Cleveland?" A change came over the woman's face as I asked the question. It was some seconds before she could get out the **single** word "Yes" – and when it did come it was in a husky, unnatural tone.
Embed 3rd bit: 1 Target adj: *single*	$Share_0$: "Have you heard of the death of your late boarder Mr. Enoch J. Drebber, of Cleveland?" A **terrible** change came over the woman's face as I asked the question. It was some seconds before she could get out the word "Yes" – and when it did come it was in a husky, unnatural tone. $Share_1$: "Have you heard of the **mysterious** death of your late boarder Mr. Enoch J. Drebber, of Cleveland?" A change came over the woman's face as I asked the question. It was some seconds before she could get out the **single** word "Yes" – and when it did come it was in a husky, unnatural tone.

Figure 6: An example of the secret sharing scheme

secret bitstring can be done by aligning the two texts. The alignment will reveal the positions of the deletable adjectives, which gives the order of the 0s and 1s, and therefore the secret can be extracted. Note that this scheme does not require either the original text or the adjective checking model to recover the secret bitstring.

Conclusion

One of the contributions of this paper is to explore the identification of redundant adjectives in noun phrases. We proposed two methods for checking the sentence naturalness after removing an adjective, which were evaluated by human judgements. The results suggest that the adjective deletion technique is applicable to cryptosystems since the transformation is able to achieve satisfactory imperceptibility leading to a high security level. According to our observations from section A of the BNC, on average there are two deletable adjectives per five sentences. In other words, the payload upper bound of using the adjective deletion technique is around 0.4 bits per sentence if a deletion encodes a secret bit. Apart from the cryptosystem applications, the proposed adjective checking model can also benefit other studies such as sentence compression, text simplification and text summarisation.

Another contribution of this paper is the integration of the adjective deletion technique into an existing stegosystem and the proposal of a novel secret sharing scheme based on adjective deletion. An advantage of our proposed system is that it is somewhat language and domain independent. For future work, we would like to explore more lexical redundancies in other POS, such as adverbs and punctuation, so the payload capacities of our cryptosystems can be further improved.

Acknowledgments

We would like to thank Dr. Laura Rimell and the anonymous reviewers for useful comments and the annotators for their time.

References

Atallah, M. J., McDonough, C. J., Raskin, V., and Nirenburg, S. (2001a). Natural language processing for information assurance and security: an overview and implementations. In *Proceedings of the 2000 workshop on New security paradigms*, pages 51–65, Ballycotton, County Cork, Ireland.

Atallah, M. J., Raskin, V., Crogan, M. C., Hempelmann, C., Kerschbaum, F., Mohamed, D., and Naik, S. (2001b). Natural language watermarking: design, analysis, and a proof-of-concept implementation. In *Proceedings of the 4th International Information Hiding Workshop*, volume 2137, pages 185–199, Pittsburgh, Pennsylvania.

Atallah, M. J., Raskin, V., Hempelmann, C. F., Karahan, M., Topkara, U., Triezenberg, K. E., and Sion, R. (2002). Natural language watermarking and tamperproofing. In *Proceedings of the 5th International Information Hiding Workshop*, pages 196–212, Noordwijkerhout, The Netherlands.

Bergsma, S., Lin, D., and Goebel, R. (2009). Web-scale n-gram models for lexical disambiguation. In *Proc. IJCAI*, pages 1507–1512, Pasadena, California.

Blakley, G. (1979). Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, volume 48, pages 313–317. AFIPS Press.

Bolshakov, I. A. (2004). A method of linguistic steganography based on collocationally-verified synonym. In *Information Hiding: 6th International Workshop*, volume 3200, pages 180–191, Toronto, Canada.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Chang, C.-Y. and Clark, S. (2010a). Linguistic steganography using automatically generated paraphrases. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 591–599, Los Angeles, California.

Chang, C.-Y. and Clark, S. (2010b). Practical linguistic steganography using contextual synonym substitution and vertex colour coding. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1194–1203, Cambridge, MA.

Chapman, M. and Davida, G. I. (1997). Hiding the hidden: A software system for concealing ciphertext as innocuous text. In *Proceedings of the First International Conference on Information and Communication Security*, volume 1334, pages 335–345, Beijing.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16:22–29.

Clark, S. and Curran, J. R. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Comp. Ling.*, 33(4):493–552.

Cohn, T. and Lapata, M. (2008). Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 137–144, Manchester, UK.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19:61–74.

Filippova, K. and Strube, M. (2008). Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32.

Fleiss, J. L., Levin, B., and Paik, M. C. (2003). *Statistical Methods for Rates & Proportions*. Wiley-Interscience, 3rd edition.

Fridrich, J. (2009). *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, first edition.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2010). A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University.

Kim, M.-Y. (2008). Natural language watermarking for korean using adverbial displacement. In *Multimedia and Ubiquitous Engineering*, pages 576–581, Busan, Korea.

Kim, M.-Y. (2009). Natural language watermarking by morpheme segmentation. In *First Asian Conference on Intelligent Information and Database Systems*, pages 144–149, Dong hoi, Quang binh, Vietnam.

Knight, K. and Marcu, D. (2002). Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Lee, L. (1999). Measures of distributional similarity. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 25–32, College Park, Maryland.

Liu, Y., Sun, X., and Wu, Y. (2005). A natural language watermarking based on Chinese syntax. In *Advances in Natural Computation*, volume 3612, pages 958–961, Changsha, China.

Meral, H. M., Sankur, B., Sumru Özsoy, A., Güngör, T., and Sevinç, E. (2009). Natural language watermarking via morphosyntactic alterations. *Computer Speech and Language*, 23(1):107–125.

Meral, H. M., Sevinc, E., Unkar, E., Sankur, B., Ozsoy, A. S., and Gungor, T. (2007). Syntactic tools for text watermarking. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, CA.

Minnen, G., Carroll, J., and Pearce, D. (2001). Applied morphological processing of English. *Nat. Lang. Eng.*, 7:207–223.

Murphy, B. (2001). Syntactic information hiding in plain text. Master's thesis, Trinity College Dublin.

Murphy, B. and Vogel, C. (2007a). Statistically-constrained shallow text marking: techniques, evaluation paradigm and results. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, California.

Murphy, B. and Vogel, C. (2007b). The syntax of concealment: reliable methods for plain text information hiding. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, CA.

Shamir, A. (1979). How to share a secret. *Commun. ACM*, 22:612–613.

Stutsman, R., Grothoff, C., Atallah, M., and Grothoff, K. (2006). Lost in just the translation. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 338–345, Dijon, France.

Taskiran, C. M., Topkara, M., and Delp, E. J. (2006). Attacks on linguistic steganography systems using text analysis. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6072, pages 97–105, San Jose, CA.

Topkara, M., Riccardi, G., Hakkani-Tür, D., and Atallah, M. (2006a). Natural language watermarking: Challenges in building a practical system. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6072, pages 106–117, San Jose, California.

Topkara, M., Taskiran, C. M., and Delp, E. J. (2005). Natural language watermarking. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 5681, pages 441–452, San Jose, CA.

Topkara, M., Topkara, U., and Atallah, M. J. (2006b). Words are not enough: sentence level natural language watermarking. In *Proceedings of the ACM Workshop on Content Protection and Security*, pages 37–46, Santa Barbara, CA.

Topkara, U., Topkara, M., and Atallah, M. J. (2006c). The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th Workshop on Multimedia and Security*, pages 164–174, Geneva, Switzerland.

Venugopal, A., Uszkoreit, J., Talbot, D., Och, F., and Ganitkevitch, J. (2011). Watermarking the outputs of structured prediction with an application in statistical machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1363–1372, Edinburgh, Scotland, UK.

Vybornova, M. O. and Macq, B. (2007). A method of text watermarking using presuppositions. In *Proceedings of the SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, CA.

Zhu, Z., Bernhard, D., and Gurevych, I. (2010). A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1353–1361, Beijing, China.