The Secret's in the Word Order: Text-to-Text Generation for Linguistic Steganography

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
Linguistic steganography is a form of covert communication using natural language to conceal the existence of the hidden message, which is usually achieved by systematically making changes to a cover text. This paper proposes a linguistic steganography method using word ordering as the linguistic transformation. We show that the word ordering technique can be used in conjunction with existing translation-based embedding algorithms. Since unnatural word orderings would arouse the suspicion of third parties and diminish the security of the hidden message, we develop a method using a maximum entropy classifier to determine the naturalness of sentence permutations. The classifier is evaluated by human judgements and compared with a baseline method using the Google n-gram corpus. The results show that our proposed system can achieve a satisfactory security level and embedding capacity for the linguistic steganography application.

KEYWORDS: linguistic steganography, word ordering, surface realisation.

Proceedings of COLING 2012: Technical Papers, pages 511–528,
COLING 2012, Mumbai, December 2012.

1 Introduction

Linguistic steganography is a form of covert communication using natural language to conceal the existence of the hidden message so that the very act of communication is undetectable to an outside observer (human or computer) (Fridrich, 2009). In other words, the covert communication would fail if an outside observer is suspicious of the existence of the hidden message. Note that the outside observer here is not the recipient; the recipient needs to expect to receive a hidden message in order to extract it. (Consider the scenario of covert communication between political activists, where the observer is a government "listening" agency, for example.)

In order to embed messages, a cover text must provide information carriers that can be modified to represent the secret. For example, a lexical substitution-based stegosystem substitutes selected words (the information carriers) with their synonyms so that the concatenation of the bitstrings represented by the synonyms is identical to the secret. Note that an unmodifiable text cannot carry information. Ideally the changes made to the text must be imperceptible, resulting in a high security level. In addition, an ideal linguistic steganography scheme should allow sufficient embedding capacity to achieve efficient information transmission, resulting in a large payload capacity. There is a fundamental trade-off between security and payload since any attempt to embed additional information is likely to increase the chance of introducing anomalies into the text, thus degrading the security level.

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), semantic transformations (Atallah et al., 2002; Vybornova and Macq, 2007) and text translation (Grothoff et al., 2005; Stutsman et al., 2006; Meng et al., 2011). 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.

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 (MT) evaluation metrics BLEU and NIST, automatically measuring how close a stego sentence is to the original. Topkara et al. (2006b) admitted that MT 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. In our work, we also use human judgements to evaluate the proposed stegosystem as this is close to the linguistic steganography scenario. Note that it is hard to compare the security level of different systems since there are no standard methods to measure the security (imperceptibility) of a linguistic steganography system.

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.

For the lexical substitution transformation, Topkara et al. (2005) and Topkara et al. (2006c) achieved an average embedding payload of 0.67 bits per sentence. The payload attained by syntactic transformations was around 0.5 to 1.0 bits per sentence (Atallah et al., 2001b; Topkara et al., 2006b; Meral et al., 2009). Since the ontological semantic transformation is currently impractical, the empirical payload is not available. Another semantic method (Vybornova and Macq, 2007) achieves a payload of 1 bit per sentence. Stutsman et al. (2006) showed that their translation-based stegosystem has a payload of 0.33 bits per sentence.

In this paper, we exploit word ordering as the linguistic transformation. A cover sentence is first used to provide a bag-of-words as input to the Zhang et al. (2012) word ordering realisation system. The generated permutations provide alternatives to the original and thus the cover sentence can play the role of an information carrier. However, not all the permutations are grammatical and semantically meaningful. To solve this problem we train a Maximum Entropy classifier (Berger et al., 1996) to distinguish natural word orders from awkward wordings, and evaluate the classifier using human judgements. Note that the proposed maximum entropy classifier is trained to classify sentence-leval naturalness and thus, it is possible that even individual natural sentences might lead to an unnatural document. Modeling the document-level coherence of modified text would be useful but is outside the scope of our study. In addition, we review some translation-based stegosystems and show that the word ordering transformation can be used with the existing translation-based embedding algorithms. For readers unfamiliar with linguistic steganography, Chang and Clark (2010a) present the general linguistic steganography framework and describe several existing stegosystems using different linguistic transformations.

A contribution of this paper is to create a novel link between word ordering realisation and linguistic steganography. The various permutations provide a possible covert channel for secret communication. To our knowledge, this is the first linguistic steganography system using word ordering as the transformation. In addition, we propose a maximum entropy classifier to determine the naturalness of a permutation. The collected human judgements of sentence naturalness and the resulting classifier may be of use for other NLP applications.

2 Word Ordering-based Steganography

The general word ordering problem is to construct grammatical, fluent sentences from a set of un-ordered input words. There have been some word ordering realisation systems that take a

Rank (binary)	Permutation	Secret Bitstring $s=1$	$s=2$	$s=3$
1 (001)	In our products now there is no asbestos.	1	01	001
2 (010)	No asbestos there is now in our products.	0	10	010
3 (011)	Now in our products there is no asbestos.	1	11	011
4 (100)	There is no asbestos in our products now.	0	00	100
5 (101)	There no asbestos in our products is now.	1	01	101
6 (110)	There now is no asbestos in our products.	0	10	110

Figure 1: Ranked sentence permutations and their secret bits

bag-of-words as input and automatically generate permutations (Wan et al., 2009; Zhang and Clark, 2011; Zhang et al., 2012). Any of these word ordering realisation systems can be treated as a module integrated into the proposed secret embedding method.

For linguistic steganography, there exists a convenient modularity between the linguistic transformation and the embedding method. In other words, a secret embedding method can treat the linguistic transformation as a black box for outputting alternatives for a given cover text. How an alternative encodes secret bits is decided by the embedding algorithm. In the next section, we propose an embedding method designed to use permutations from a word ordering system as alternatives for a given cover sentence. Later, in Section 6 we show that the word ordering technique can also be combined with existing translation-based embedding algorithms.

2.1 The Embedding Method

Before any message exchange can take place, the sender and receiver must share a word ordering system and a method to sort a set of sentence permutations, such as alphabetical ordering, or ordering by realisation quality scores determined by the generation system or a language model. The sorting method must be independent of the cover sentence since the receiver does not receive it, and only those permutations having the same length as the cover are considered during sorting. In addition, the number of secret bits carried by a permutation must be fixed and known by the sender and the receiver.

The sender first turns a cover sentence into a bag-of-words and then uses the pre-agreed word ordering system to generate permutations. After eliminating permutations shorter than the cover sentence (if any), the rest are sorted using the pre-agreed method. The rank of a permutation in the sorted list is converted into a binary string, and the lowest s bits are the secret bits carried by that permutation, where s is the pre-fixed payload. Figure 1 shows six alphabetically ranked permutations and the secret bit(s) carried by them when s is equal to 1, 2 and 3. Note that, in order to embed s bits, there must be at least 2^s permutations in the ordered list; otherwise, there will be at least one desired secret bitstring not carried by any permutation. For example, in Figure 1 secret bitstring 000 and 111 cannot be found when s is equal to 3. Finally, the sender can choose a permutation that represents the secret bitstring as the stego sentence.

To recover the secret bitstring, the receiver first transforms the received stego sentence into a bag-of-words. Since we only consider permutations having the same length as the cover during embedding, the bag-of-words obtained from the stego sentence will be identical to that originally obtained from the cover sentence. Next, the receiver reproduces the ordered

permutations. According to the rank of the stego sentence and the pre-agreed payload of s bits, the receiver can extract the secret bitstring. Note that, in the proposed stegosystem, the receiver can extract the secret without knowing the cover text.

The payload of the system is controlled by the variable s, and the security level depends on the quality of the selected permutations. For example, although "In our products now there is no asbestos" and "There no asbestos in our products is now" both carry secret bits "01" in Figure 1, using the latter would significantly decrease the security level as it is an unnatural sentence. Therefore, in Section 4.2 we develop a Maximum Entropy classifier to determine the naturalness of a permutation which can be integrated into the stegosystem during data embedding. Note that, in this paper, we do not investigate the document-level coherence of stego text generated using the word ordering transformation since this requires sophisticated knowledge of natural language semantics and pragmatics. Instead, we tackle the problem of distinguishing the naturalness of a sentence permutation in isolation from the rest of a document.

3 Resources

3.1 Word Ordering Realisation System

The word ordering realisation system plays a crucial role in the proposed steganography method as it largely determines the security level and payload capacity of the stegosystem. The sender in the proposed scheme uses the n-best list output from a word ordering system as the set of possible alternatives for a cover sentence; so the security level largely depends on the quality of the n-best list. In addition, the more acceptable permutations an n-best list has, the larger the payload the stegosystem can use.

Some recent work used syntax models to certify the grammaticality of generated sentences. The combination of permutation and syntactic modelling results in a large search space, which was tackled using heuristic search (Wan et al., 2009; Zhang and Clark, 2011; Zhang et al., 2012). Wan et al. (2009) use a dependency grammar to model word ordering and apply greedy search to find the best permutation. Zhang and Clark (2011) use a syntax-based discriminative model together with best-first search to find the highest scoring permutation plus CCG derivation. Zhang et al. (2012) is an extension of Zhang and Clark (2011) using online large-margin training and incorporating a large-scale language model. The three approaches are evaluated using the generation task of word order recovery, which is to recover the original word order from an input bag-of-words. The BLEU scores reported by Wan et al. (2009), Zhang and Clark (2011) and Zhang et al. (2012) are 33.7, 40.1 and 43.8, respectively, on WSJ newspaper sentences. In this paper, we use the Zhang et al. (2012) system to generate n-best permutations for a cover sentence, but, in practice, any word ordering realisation system can be integrated into the proposed word ordering-based stegosystem.

3.2 Human Judgement Corpus

Since not all the sentence permutations generated by the Zhang et al. (2012) system are grammatical and semantically meaningful, we develop a maximum entropy classifier to determine the naturalness of permutations. In order to have a labelled corpus for training and testing the classifier, we first randomly selected 765 sentences having length between 8 and 25 tokens from sections 02-21 of the Penn Treebank (Marcus et al., 1993) as the cover sentences. The restriction on the sentence length is because a short sentence may not have enough good permu-

Score	Explanation
1	Completely or largely non-fluent, and/or completely or largely lacking in meaning.
2	Very awkward wording, major punctuation errors, and/or logical errors, but still possible to understand.
3	Slightly awkward but still relatively fluent, clear and logical; may contain slightly awkward wording and/or minor punctuation errors.
4	Perfectly natural – both grammatical and semantically meaningful.

Figure 2: Rating scale and guidelines for human evaluation

Score	Permutation
3	As a price-conscious shopper, so far, Manville hasn't bought much.
1	So, as a price-conscious shopper, far Manville hasn't bought much.
1	Far so, as a price-conscious shopper, Manville hasn't bought much.
4	So far Manville, as a price-conscious shopper, hasn't bought much.
1	Far Manville, as a price-conscious shopper, so hasn't bought much.

Figure 3: A set of five permutations rated by a subject

tations for the steganography application, and a long cover sentence increases the complexity of finding acceptable word orders from the bag-of-words and therefore is unlikely to result in good permutations.

For each cover sentence, we created a bag-of-words as input and generated 100 permutations using the Zhang et al. (2012) system. For 88% of the sentences, the original cover sentence is in the 100-best list. This not only serves as a sanity check for the realisation system, but also means the original sentence can be used to carry secret bits without any modification.

To cut down a 100-best list for the human evaluation, we only keep five permutations that retain the most grammatical relationships of the original cover sentence since these permutations are more likely to convey the same meaning as the original. We parsed the cover sentences and their permutations using a CCG parser (Clark and Curran, 2007), and calculated a dependency F-score for each permutation by comparing the CCG predicate-argument dependencies of the permutation and its original. For each cover sentence, the top five F-score permutations which are different from the cover were chosen for human annotation, which results in 3,825 sentences (765 sets of 5 permutations).

A total of 34 native English speakers were asked to judge the naturalness of the sentence permutations on a 4-point scale. The guidelines are shown in Figure 2. The annotations were carried out via a web interface; on each page, a subject was presented with 5 permutations consisting of the same words, and a total of 125 permutations (25 sets) were annotated by each subject. Figure 3 shows a set of five permutations along with the scores rated by a subject. In order to calculate the inter-annotator agreement, 425 permutations were selected to be judged by two annotators.

For the steganography application, those permutations rated as perfectly natural (score 4) can achieve a high security level and are treated as positive data when training a Maximum Entropy classifier; those permutations with scores lower than 4 are treated as negative data. After converting the scores into a positive/negative representation, we measured the inter-annotator

agreement on the binary labelled data using Fleiss' kappa (Fleiss et al., 2003). The resulting kappa score of 0.54 for the data represents "moderate" agreement according to Landis and Koch (1977). There were 47 out of the 425 agreement-measuring sentences that received different labels after applying the positive/negative representation, for which the second author of this paper made the definitive judgement. In the end, the collected human judgement corpus contained 478 positive (perfectly natural) permutations and 3,347 negative examples.

According to the human judgements, 321 out of the 765 cover sentences have at least one natural sounding permutation in the top five F-score permutations. Therefore, the upper bound of the number of possible information carriers is roughly 42% of the cover sentences. Next, since there are studies using automatic evaluation metrics to evaluate the security level of a stegosystem as described in Section 1, we observe how well the BLEU score of a permutation correlates with the human judgement. For those multi-judged sentences, an average score is assigned. Both Pearson's correlation coefficient and Spearman's rank correlation coefficient between the human judged scores and the BLEU scores are calculated, which are 0.10 and 0.09, respectively, and are significant at $p < 0.001$. This result indicates there is little association between the human judgement of sentence naturalness and the BLEU score, indicating the need for a manual evaluation to determine the likely security level.

4 Sentence Naturalness Classification

According to the human judgement corpus, most of the machine-generated permutations are not natural. In addition, as described in Section 2, the proposed secret embedding algorithm sometimes involves choosing a permutation from a group of candidates that all encode the same secret bit(s). Therefore, it is crucial to develop a method that can distinguish acceptable permutations from those having awkward wordings in a word ordering-based stegosystem. It is important to note that the checking method can take the original sentence into consideration because the permutation selection happens at the secret embedding stage and is not needed during the decoding. Having the cover sentence available at the checking stage is a feature we will exploit.

A research area that relates to the proposed permutation checking method is realisation ranking (Cahill and Forst, 2010; White and Rajkumar, 2012) where a system is given a set of text realisations and is asked to rate each text in the set. However, in the realisation ranking task there is no "cover text". Since our methods require the knowledge of the original text, they cannot be applied to the realisation ranking task.

In this section we first explain a baseline method using the Google n-gram corpus (Brants and Franz, 2006) to check whether a particular word ordering has been used frequently on the Web. Then we propose another approach using some syntactic features to train a Maximum Entropy classifier (Berger et al., 1996). Both methods require a score/probability threshold to decide the acceptability of a permutation.

4.1 Google N-gram Method

The Google n-gram corpus (Brants and Franz, 2006) contains frequency counts for n-grams from unigrams through five-grams obtained from over 1 trillion word tokens of English Web text collected in January 2006. Only n-grams appearing more than 40 times were kept in the corpus. The Google n-gram corpus has been applied to many NLP tasks such as spelling correction (Carlson et al., 2008; Islam and Inkpen, 2009), multi-word expression classification

(Kummerfeld and Curran, 2008) and lexical disambiguation (Bergsma et al., 2009). Recently, in Chang and Clark (2010b) and Chang and Clark (2010a) we have used the corpus to check the text paraphrasing grammaticality and the synonym acceptability in context in our earlier steganography systems. Therefore, we propose a baseline method similar to Chang and Clark (2010b) and Chang and Clark (2010a) using the Google n-gram corpus to calculate a score based on the n-gram counts before and after word ordering. The task is as follows: given a cover sentence and corresponding permutation, decide if the permutation is acceptable. The baseline method will do so by comparing Google n-gram counts from the two sentences.

In the Google n-gram corpus, sentence boundaries are marked by <S> and </S>, where <S> represents the beginning of a sentence and </S> represents the end of a sentence. Both tags are treated like word tokens during the n-gram collection. Hence, after tokenising the cover sentence and its corresponding permutation, we add <S> and </S> tags to the beginning and end of the sentences as shown in Figure 4. Then we extract every bi- to five-gram from the cover sentence and the permutation. Since we are only interested in newly generated wordings in the permutation, n-grams that appear in both the cover and the permutation are eliminated, and the remaining n-grams and their Google n-gram frequencies are used to calculate the score. For example, after comparing the two sentences, there are 21 n-grams from the cover and 21 n-grams from the permutation left in Figure 4. We sum up all the logarithmic counts[1] for the cover and permutation n-grams and derive Sum_{Cover} and $Sum_{Permutation}$. The *Score* of a permutation is defined as the ratio of $Sum_{Permutation}$ and Sum_{Cover}, which measures how much the *Sum* varies after performing the word ordering. In the given example, the *Score* of the permutation is 0.81. Similar to Chang and Clark (2010b) and Chang and Clark (2010a), a score threshold is needed to determine the acceptability of a permutation. Even though this baseline method is only an n-gram count comparison, Bergsma et al. (2009) show that the approach works well for lexical disambiguation tasks and produces comparable performance to other more complex methods.

4.2 Maximum Entropy Classifier

In addition to the baseline method, we propose a machine learning approach to classify natural and unnatural permutations. We choose the method of maximum entropy modelling (MaxEnt for short) because of its proven performance for NLP tasks, such as part-of-speech tagging (Ratnaparkhi et al., 1996; Curran and Clark, 2003), parsing (Ratnaparkhi, 1999; Johnson et al., 1999) and language modelling (Rosenfeld, 1996), and the ease with which features can be included in the model (Ratnaparkhi, 1999). In addition, some work has shown that MaxEnt is viable for ranking the fluency of machine generated sentences (Nakanishi et al., 2005; Velldal and Oepen, 2006; Velldal, 2008). The concept of MaxEnt is to use observed features about a certain event (y) occurring in the context (x) to estimate a probability model $p(y|x)$. Its canonical form is:

$$p(y|x) = \frac{1}{Z(x)} exp \sum_{i=1}^{n} \lambda_i f_i(x, y)$$

where $Z(x)$ is a normalisation constant over all events in context x and λ_i is the weight of the feature $f_i(x, y)$. The standard way to train a maximum entropy model is to use conditional maximum likelihood (with a Gaussian prior for smoothing), which is equivalent to picking the most uniform model subject to constraints on the feature expectations (Berger et al., 1996).

[1] log(0) and division by zero are taken to be zero.

Cover: <S> There is no asbestos in our products now . </S>		Permu: <S> In our products now there is no asbestos . </S>	
log *freq*	n-gram	log *freq*	n-gram
19.1	<S> There	20.3	<S> In
11.6	asbestos in	14.3	now there
17.6	now .	12.0	asbestos .
17.8	<S> There is	15.2	<S> In our
6.1	no asbestos in	0	products now there
6.0	asbestos in our	13.0	now there is
9.3	products now .	6.5	no asbestos .
17.6	now . </S>	12.0	asbestos . </S>
16.4	<S> There is no	6.7	<S> In our products
5.1	is no asbestos in	0	our products now there
0	no asbestos in our	0	products now there is
0	asbestos in our products	11.1	now there is no
6.8	our products now .	3.7	is no asbestos .
0	products now . </S>	0	no asbestos . </S>
4.0	<S> There is no asbestos	0	<S> In our products now
4.8	There is no asbestos in	0	In our products now there
0	is no asbestos in our	0	our products now there is
0	no asbestos in our products	0	products now there is no
0	asbestos in our products now	0	now there is no asbestos
0	in our products now .	0	there is no asbestos .
0	our products now . </S>	0	is no asbestos . </S>
$Sum_{Cover} = 142.1$		$Sum_{Permutation} = 114.9$	

Figure 4: An example of the Google n-gram method

After training a maximum entropy classifier, we can calculate the probabilities of a permutation being a natural sentence according to the feature weights. Our proposed method says that a permutation is natural if the ratio of its naturalness probability to its unnaturalness probability is greater than a threshold α. The threshold α controls the trade-off between precision and recall of the maximum entropy classifier and can be decided by steganography users. The MaxEnt implementation we used was from the Curran and Clark (2003) tagger, adapted for classification rather than sequence tagging.

4.2.1 Features

In the formulation of MaxEnt we use, a feature f_i is an "indicator function" on events which simply indicates the presence of a feature. The first feature we included is the Levenshtein distance (Levenshtein, 1966) which measures the minimum number of edits needed to transform one cover sentence into its permutation, with the allowable edit operations being insertion, deletion, or substitution. After deriving the edit distance d, an indicator "EDIST_D" becomes the feature for that permutation, where D=$\lfloor log_2 d \rfloor$. For example, a permutation with an edit distance 4 and another permutation with an edit distance 5 both have the same indicator function "EDIST_2". In addition, if the difference between a permutation and its original sentence is only a single word movement, we add the *POS* tag of the moved word to the indicator function, so the feature becomes "EDIST_1-*POS*".

The second type of feature is derived by comparing the Stanford typed dependencies (De Marneffe and Manning, 2008) of a permutation and its original sentence. The Stanford typed dependencies provide descriptions of the grammatical relationships as well as semantically contentful information in a sentence, which can be obtained from the Stanford parser (De Marneffe et al., 2006). The Stanford typed dependencies are triples denoting a relation between a governor and a dependent. For example, *amod(wine, red)* denotes that *red* is an adjectival modifier of *wine*, and *agent(killed, spy)* denotes that an agent *spy* is the complement of a verb *killed*.

Cover: There is no asbestos in our products now.	Permutation: Our products there is no asbestos in now.
Stanford typed dependencies:	Stanford typed dependencies:
expl(is-VBZ-2, there-EX-1)	poss(products-NNS-2, our-PRP$-1)
root(ROOT-ROOT-0, is-VBZ-2)	nsubj(asbestos-NN-6, products-NNS-2)
det(asbestos-NN-4, no-DT-3)	advmod(asbestos-NN-6, there-RB-3)
nsubj(is-VBZ-2, asbestos-NN-4)	cop(asbestos-NN-6, is-VBZ-4)
prep(asbestos-NN-4, in-IN-5)	det(asbestos-NN-6, no-DT-5)
poss(products-NNS-7, our-PRP$-6)	root(ROOT-ROOT-0, asbestos-NN-6)
pobj(in-IN-5, products-NNS-7)	prep(asbestos-NN-6, in-IN-7)
advmod(is-VBZ-2, now-RB-8)	pobj(in-IN-7, now-RB-8)
dependency indicator functions of the permutation: P_dep_nsubj, P_deppos_nsubj_NN_NNS, P_dep_advmod, P_deppos_advmod_NN_RB, P_dep_cop, P_deppos_cop_NN_VBZ, P_dep_root, P_deppos_root_ROOT_NN, P_dep_pobj, P_deppos_pobj_IN_RB, R_dep_poss, R_deppos_poss_NNS_PRP$_0, R_dep_det, R_deppos_det_NN_DT_0, R_dep_prep, R_deppos_prep_NN_IN_0	

Figure 5: An example of the dependency indicator functions

We first parse a permutation and its original sentence using the Stanford parser and compare their Stanford typed dependencies. If a dependency *TYPE(WORD1, WORD2)* in the permutation cannot be found in the original, two indicator functions "P_dep_*TYPE*" and "P_deppos_*TYPE_POS1_POS2*" are added to the permutation's feature set, where *POS1* and *POS2* are the POS tags of *WORD1* and *WORD2*, respectively. If a dependency *TYPE(WORD1, WORD2)* in the permutation is the same as that in the original, two indicator functions "R_dep_*TYPE*" and "R_deppos_*TYPE_POS1_POS2_DISTANCE*" are added to the permutation's feature set, where *POS1* and *POS2* are the POS tags of *WORD1* and *WORD2*, and *DISTANCE* is the difference of the distance between the two words compared to the original. Figure 5 shows the dependency indicator functions of the permutation "our products there is no asbestos in now". In this example, *nsubj(asbestos, products)* is a newly generated relation after word ordering so two indicator functions *P_dep_nsubj* and *P_deppos_nsubj_NN_NNS* are added to the permutation's feature set; *poss(products, our)* is a recovered relation from the original and the distance between *product* and *our* remains the same as that in the origial so two indicator functions *R_dep_poss* and *P_deppos_poss_NNS_PRP$_0* are added to the permutation's feature set.

5 Experiments and Results

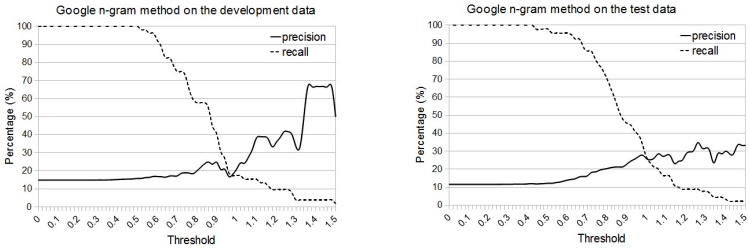
We evaluate the Google n-gram method and the maximum entropy classifier using the collected human judgements. The performance of the systems is measured in precision and recall over the natural permutations (i.e. the positive examples in the test set). Note that the trade-off between precision and recall implies the trade-off between security and payload capacity from a steganography perspective. A higher precision value means that the system-passed permutations are of high quality so using those permutations as stego sentences is unlikely to arouse suspicion; whereas a larger recall value can be interpreted as the system making the most of natural permutations and therefore embedding as much data as possible.

5.1 Data Sets

We divided the collected human judgement corpus described in Section 3.2 into a 2700-instance training set, a 350-instance development set and a 775-instance test set. The development set was mainly used for preliminary experimentation. We will present results on the development and test sets. Note that the 425 multi-judged sentences are all included in the test set. We treat

	Training set	Development set	Test set
number of positives	467	52	90
number of negatives	2,364	298	685

Table 1: Statistics of the experimental data sets



(a) Results on the development set

(b) Results on the test set

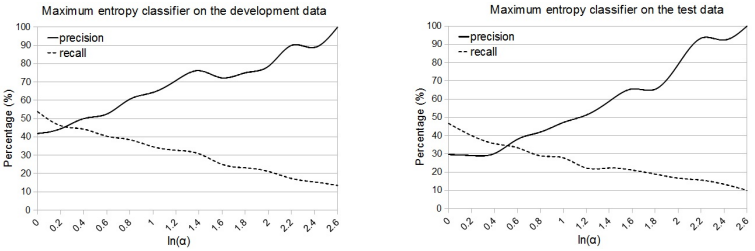
Figure 6: Performance of the Google n-gram method

natural permutations as positives and unnatural sentences as negatives. Since the number of negatives is 7 times more than the number of positives in the training set, we added another 131 positives annotated by the authors to the training set (but not the test set), in an attempt to address the imbalance. Table 1 presents the statistics of the data sets.

5.2 Experiments Using Google N-gram Method

We evaluated the Google n-gram method on the development set. Figure 6(a) shows the precision and recall curves with respect to different threshold values. The best precision achieved by the system is 66.7% with a very low recall of 3.9% when the threshold is equal to 1.36. Then we use the threshold 1.36 to classify positive and negative data in the test set. The derived precision and recall values are 28.6% and 4.4%, respectively. Figure 6(b) gives the precision and recall curves obtained by using the Google n-gram method on the test data. From the diagram we can see that, even when a threshold 1.26 is chosen, the best precision on the test set is only 34.8%, which is not appropriate for the steganography application since the low precision value would result in an unnatural stego text and hence fail the secret communication requirement.

A possible explanation of the poor performance of the n-gram baseline is that the n-gram method might be useful for checking local word changes (e.g. synonym substitution), but not the whole sentence rearrangement. In addition, longer n-grams are not frequently found in the Google n-gram corpus according to our data so in these cases the n-gram method only relies on checking the changes in lower-order n-grams, such as bi-grams or tri-grams.



(a) Results on the development set

(b) Results on the test set

Figure 7: Performance of the maximum entropy classifier

5.3 Experiments Using Maximum Entropy Classifier

Next we train a maximum entropy classifier using sentences in the training set. Each permutation in the training set is first represented by its indicator functions and is labelled as either a positive or a negative instance according to the human judgements. A total of 4,490 indicator functions are extracted from the training set. Since in the training set, the ratio of positives to negatives is about 1:5, we duplicate the positives 5 times to balance the amount of positives and negatives in the training set. The trained 5,815 weights are used to calculate the probabilities of a test instance being positive and negative. As mentioned in Section 4.2, the system determines an instance as positive if:

$$\frac{exp\sum_{i=1}^{n}\lambda_i f_i(x,positive)}{exp\sum_{i=1}^{n}\lambda_i f_i(x,negative)} > \alpha \implies \sum_{i=1}^{n}\lambda_i f_i(x,positive) - \sum_{i=1}^{n}\lambda_i f_i(x,negative) > ln(\alpha)$$

We observe the precision and recall values of the classifier with different α values. Figure 7(a) shows the performance of the classifier on the development set. The classifier achieves a 90% precision with 17.3% recall when the threshold $ln(\alpha)$ is equal to 2.2. A precision of 100% can be obtained by raising the threshold to 2.6 with the corresponding recall being 13.5%. Since the inter-annotator agreement on the collected human judgements is not 100%, as shown in Section 3.2, it is not clear whether the 90% precision achieved by the classifier really means that the remaining 10% sentences (false positives) would be viewed as suspicious in a real steganography setting. Therefore, we consider 90% to be a high level of precision/security.

The same classifier is then used to determine natural permutations in the test set and the $ln(\alpha)$ is set to 2.2 since this setting gives a satisfactory imperceptibility and payload capacity for the development set. The classifier achieves a precision of 93.3% and a recall of 15.6% with the 2.2 threshold, which again provides a confident security level and reasonable embedding capacity for the steganography application. This result is much better than the precision of 34.8% achieved by the baseline Google n-gram method. Since the training data used in our classifier is imbalanced (the number of negatives is five times more than that of positives), the features observed from positive data may not be enough to gain a higher recall. Therefore, we expect the recall value can be improved using more balanced training data.

In order to show the trade-off between precision and recall, which corresponds to the trade-off between imperceptibility and payload capacity for the linguistic steganography application, the precision and recall curves of the classifier on the test set are given in Figure 7(b). Note that we are not optimising on the test set; Figure 7(b) is just a demonstration of where on the precision-recall tradeoff a practical stegosystem might lie. In practice, the threshold value would depend on how steganography users want to trade off security for payload.

6 Word Ordering-based Stegosystem

In this section we first review the existing translation-based stegosystems. Then we explain how those translation-based embedding algorithms can not only work with a machine translation system, but can also be combined with a word ordering realisation system.

6.1 Translation-based Steganography

The first translation-based stegosystem was proposed by Grothoff et al. (2005). In their method, the sender and the receiver first agree on a set of machine translation systems that generate multiple translations for a given cover sentence. According to the translation probabilities, the sender then encodes each translation using Huffman coding (Huffman, 1952). Using Huffman coding, translations with higher probability (higher quality) have shorter codes and thus are more likely to match the secret bitstrings. After choosing translations that represent the secret bits, both the cover text and the stegotext are sent to the receiver. To extract the message, the receiver uses the same set of translation systems to generate translations for each cover sentence and runs the Huffman coding algorithm. Using the stegotext, the receiver can reconstruct the secret.

Grothoff et al. (2005) also proposed another scheme which does not require sending the original text to the receiver. Instead of using a set of machine translation systems as the secret key, the sender and the receiver share a hash function that transforms a sentence into a bitstring. The sender first generates multiple translations for a given cover sentence and then hashes each translation into a bitstring. A translation having its least significant bit as identical to the secret bit is selected as the stego sentence. To recover the message, without knowing the original text, the receiver only needs to compute the hash codes of the received sentences and concatenate the lowest bits of every stego sentence. Unlike the previous method that may be able to embed more than one bit in a translation, this embedding scheme has an upper bound of 1 bit per sentence.

Later Stutsman et al. (2006) improved the payload capacity of the hash function embedding scheme by introducing a header (h bits) to indicate that b bits are embedded in a translation, where h is shared between the sender and the receiver, and b is the integer represented by the header bits. The lowest h bits of a translation hash bitstring are the header bits and the lowest $[h{+}1, h{+}b]$ bits carry the secret. For example, assume $h = 2$; a hash bitstring "...10111" has header bits "11" to indicate 3 bits are embedded in this translation, and the three secret bits are "101". The problem of using a hash function is that the generation of a desired bitstring cannot be guaranteed. Therefore, error correction codes must be used in this protocol when there is no feasible hash code available, which increase the size of the transmission data.

Since Grothoff et al. (2005) and Stutsman et al. (2006) use multiple machine translation systems to generate alternative translations, the selected stego sentences may not have uniform style and therefore it is easy to detect the existence of the secret message (Meng et al., 2010;

Chen et al., 2011). Instead of obtaining alternative translations from multiple translation systems, Meng et al. (2011) use a statistical machine translation system to generate the n-best translations for a given cover sentence. Since translations are from one system, each of them is more similar to the rest than that derived from another translation system. Meng et al. (2011) show that the existing steganalysis methods cannot distinguish ordinary translation text and the stegotext generated by their stegosystem.

6.2 Using Word Ordering in Translation-based Embedding

The translation-based embedding algorithms described in the previous section take a cover sentence as the input and choose one of the translations as the stego sentence. We can easily replace the machine translation system(s) in these algorithms with a word ordering realisation system: a cover sentence is used to provide a bag-of-words as input, and the generated permutations can be seen as the "translations" of the cover sentence. One difference between the proposed embedding method and the translation-based embedding methods is that the former restricts the length of a permutation to be the same as the cover sentence so that the receiver is able to recover the list of permutations; while the latter allows a permutation to only include a subset of the input words and therefore provides more choices for a given cover sentence. However, dropping words introduces the risk of deleting information in the cover sentence and may lead to incoherence in the resulting text.

Conclusion

In this paper we have explored the applicability of using word ordering for linguistic steganography. We proposed a steganography method using word ordering as the linguistic transformation and also showed that the word ordering technique can be applied to existing translation-based embedding algorithms. As well as the embedding method, we also proposed a method for determining the naturalness of sentence permutations by training a maximum entropy classifier. The classifier was evaluated by human judgements and compared with a baseline method using the Google n-gram corpus. The results show that the classifier achieves 93.3% precision with 15.6% recall on the test set, which is much better than the best precision of 34.8% achieved by the Google n-gram method. In addition, the ultimate precision of 100% is reported from the experiments when using a higher probability threshold. This means the proposed maximum entropy classifier can provide a high security level for the linguistic steganography application.

For future work, it is worth tackling the problem of low embedding capacity in the existing linguistic stegosystems compared with other steganography systems using images or audios as the cover medium. For example, although the proposed classifier can achieve 100% precision on the task of determining natural sentence permutations, the corresponding recall of 10% means that many available permutations are ignored, which is a waste of information carriers.

Acknowledgments

We would like to thank Dr. Yue Zhang for providing the word ordering realisation system. In addition, we would like to thank Dr. Laura Rimell, Dr. Yue Zhang 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.

Berger, A., Pietra, V., and Pietra, S. (1996). A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Bergsma, S., Lin, D., and Goebel, R. (2009). Web-scale n-gram models for lexical disambiguation. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, pages 1507–1512, Pasadena, CA.

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.

Brants, T. and Franz, A. (2006). Web 1T 5-gram corpus version 1.1. Technical report, Google Research.

Cahill, A. and Forst, M. (2010). Human evaluation of a german surface realisation ranker. In *Empirical Methods in Natural Language Generation*, volume 5790, pages 201–221. Springer.

Carlson, A., Mitchell, T. M., and Fette, I. (2008). Data analysis project: Leveraging massive textual corpora using n-gram statistics. Technical report, School of Computer Science, Carnegie Mellon University.

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*, 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.

Chen, Z., Huang, L., Meng, P., Yang, W., and Miao, H. (2011). Blind linguistic steganalysis against translation based steganography. *Digital Watermarking*, pages 251–265.

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

Curran, J. and Clark, S. (2003). Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 91–98.

De Marneffe, M., MacCartney, B., and Manning, C. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

De Marneffe, M. and Manning, C. (2008). The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

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.

Grothoff, C., Grothoff, K., Alkhutova, L., Stutsman, R., and Atallah, M. (2005). Translation-based steganography. In *Information Hiding*, pages 219–233. Springer.

Huffman, D. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

Islam, A. and Inkpen, D. (2009). Real-word spelling correction using Google Web IT 3-grams. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1241–1249, Singapore.

Johnson, M., Geman, S., Canon, S., Chi, Z., and Riezler, S. (1999). Estimators for stochastic unification-based grammars. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 535–541.

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

Kim, M. (2009). Natural language watermarking by morpheme segmentation. In *Intelligent Information and Database Systems*, pages 144–149.

Kummerfeld, J. K. and Curran, J. R. (2008). Classification of verb particle constructions with the Google Web 1T Corpus. In *Proceedings of the Australasian Language Technology Association Workshop 2008*, pages 55–63, Hobart, Australia.

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

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710.

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.

Marcus, M., Marcinkiewicz, M., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Meng, P., Hang, L., Chen, Z., Hu, Y., and Yang, W. (2010). STBS: A statistical algorithm for steganalysis of translation-based steganography. In *Information Hiding*, pages 208–220. Springer.

Meng, P., Shi, Y., Huang, L., Chen, Z., Yang, W., and Desoky, A. (2011). LinL: Lost in n-best list. In *Information Hiding*, pages 329–341. Springer.

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.

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 *Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, San Jose, CA.

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.

Nakanishi, H., Miyao, Y., and Tsujii, J. (2005). Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 93–102.

Ratnaparkhi, A. (1999). Learning to parse natural language with maximum entropy models. *Machine learning*, 34(1):151–175.

Ratnaparkhi, A. et al. (1996). A maximum entropy model for Part-Of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, pages 133–142.

Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modelling. *Computer speech and language*, 10(3):187–228.

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.

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 SPIE*, volume 6072, pages 106–117.

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.

Velldal, E. (2008). *Empirical realization ranking*. Ph.D. thesis, University of Oslo, Department of Informatics.

Velldal, E. and Oepen, S. (2006). Statistical ranking in tactical generation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 517–525.

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.

Wan, S., Dras, M., Dale, R., and Paris, C. (2009). Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 852–860, Athens, Greece.

White, M. and Rajkumar, R. (2012). Minimal dependency length in realization ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 244–255, Jeju Island, Korea.

Zhang, Y., Blackwood, G., and Clark, S. (2012). Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 736–746, Avignon, France.

Zhang, Y. and Clark, S. (2011). Syntax-based grammaticality improvement using CCG and guided search. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1147–1157, Edinburgh, Scotland, UK.