Transformations for linguistic steganography

Ching-Yun Chang

University of Cambridge
Computer Laboratory
St. Edmund's College

July 2013

This dissertation is submitted for
the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

This dissertation does not exceed the regulation length of $60,000$ words, including tables and footnotes.

Acknowledgments

I am most grateful to my supervisor, Dr. Stephen Clark, for having the confidence to take me on as a student at a time when I knew very little about research; little did he know how much hard work his choice would produce for him! His supervision, impeccable knowledge and enthusiasm have enabled me to develop an understanding of the field and become a better scientist. The completion of this thesis would not have been possible without his guidance, encouragement and patience.

I would like to express my sincere gratitude to everyone in the NLIP group for the scientific discussions, ideas and technical advice which were invaluable for the development of this project. Thanks must go to: Laura for helping me out with numerous annotation tasks and the continual support, especially giving me a lot of constructive advice when I encountered challenges; to Yue for the lunch time NLP tutorial and for giving me the chance to return to my mother tongue on occasions. I also wish to thank my examiners, Professor Ted Briscoe and Professor Carl Vogel, for your feedback which has made my research work more complete. I am also extremely grateful to the many contributing volunteer annotators for their effort and time in completing my annotations.

I would equally like to acknowledge all of the sources of financial support: from both my parents, the Taiwanese Government Studying Abroad Scholarship, Google Research Fund, Google Anita Borg Memorial Scholarship and Cambridge University St. Edmund's College Tutorial Award.

Finally, yet no less importantly, I must thank my family and friends outside the lab, especially my sister, Crystal, for providing a forum for stress-relieving sessions, and my boyfriend, Geoff, for making every effort to ensure my time in the UK has been enjoyable. Without you all, the going would have been much tougher.

I would like to dedicate this thesis to the memory of my grandmother whose love and blessings helped me throughout this journey.

Transformations for Linguistic Steganography

Ching-Yun Chang

Abstract

Linguistic steganography is a form of covert communication using natural language to conceal the existence of the hidden message. It is usually achieved by systematically making changes to a cover text, such that the manipulations, namely the very act of communication, are undetectable to an outside observer (human or computer). In this thesis, we explore three possible linguistic transformations — lexical substitution, adjective deletion and word ordering — which are able to generate alternatives for a cover text. For each transformation, we propose different transformation checkers in order to certify the naturalness of a modified sentence.

Our lexical substitution checkers are based on contextual n-gram counts and the α-skew divergence of those counts derived from the Google n-gram corpus. For adjective deletion, we propose an n-gram count method similar to the substitution n-gram checker and a support vector machine classifier using n-gram counts and other measures to classify deletable and undeletable adjectives in context. As for word ordering, we train a maximum entropy classifier using some syntactic features to determine the naturalness of a sentence permutation.

The proposed transformation checkers were evaluated by human judged data, and the evaluation results are presented using precision and recall curves. The precision and recall of a transformation checker can be interpreted as the security level and the embedding capacity of the stegosystem, respectively. The results show that the proposed transformation checkers can provide a confident security level and reasonable embedding capacity for the steganography application.

In addition to the transformation checkers, we demonstrate possible data encoding methods for each of the linguistic transformations. For lexical substitution, we propose a novel encoding method based on vertex colouring. For adjective deletion, we not only illustrate its usage in the steganography application, but also show that the adjective deletion technique can be applied to a secret sharing scheme, where the secret message is encoded in two different versions of the carrier text, with different adjectives deleted in each version. For word ordering, we propose a ranking-based encoding method and also show how the technique can be integrated into existing translation-based embedding methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the development of science and technology, we now rely heavily upon multimedia and automated systems to distribute, transmit, and gather information. One of the major concerns in such environments is data privacy since most of the data are transmitted over an open channel where information is communicated in an insecure fashion. One consequence of open communication is that potential monitoring bodies can seek to detect and control unwanted communication. Although secure communication can be implemented by using encryption techniques, sending encrypted messages frequently would draw the attention of third parties, such as crackers and hackers, and may cause attempts to break and reveal the secret messages.

In order to transmit information through an open channel without detection by anyone other than the receiver, a covert channel can be carefully designed and used. In information theory, a covert channel is a parasitic communications channel that is hidden within the medium of a legitimate communication channel (Lampson, 1973). For example, steganography is a form of covert channel in which certain properties of the medium are manipulated in an unexpected, unconventional, or unforeseen way so that, with steganographic transmission, the encrypted messages can be well camouflaged in a seemingly natural medium and sent to the receiver with less chance of being suspected and attacked. Since the changes to the medium are so subtle, anyone not specifically looking for a hidden message is unlikely to notice the changes (Fridrich, 2009).

1.1 Steganography

The word steganography has Greek origins and means "concealed writing". The original practice can be traced back to around 440 BC when the ancient Greeks hid messages within wax tables by writing messages on the wood before applying a wax surface (Herodotus, 1987). Another early recorded use of steganography is that in ancient Greece messengers tattooed messages on their shaved heads and concealed the messages with the hair that grew over them afterwards, a technique also used by German spies in the early 20th century (Newman, 1940). With the advent of tiny images, during the Franco-Prussian War (1870-1871) messages were put on microfilm and sent out by pigeon post (Tissandier, 1874); in the Russo-Japanese War (1905) microscopic images were hidden in ears, nostrils or under finger nails (Stevens, 1957); during both World Wars messages were reduced to microdots and stuck on top of printed periods or commas in innocent cover material such as magazines, or inserted into slits of the edges of post cards (Newman, 1940; Hoover, 1946). In both World Wars invisible inks were also used extensively to write messages under visible text (Kahn, 1967). The application of special inks is still used today in the field of currency security to write a hidden message on bank notes or other secure documents.

Since the 1980s, with the advent of computer technologies, digital equivalents of these camouflage techniques were invented to hide messages in digital cover media, such as images, video and audio signals (Fridrich, 2009). For example, in 2010, the United States Department of Justice retrieved more than 100 encrypted messages embedded in images that were posted to the Web. According to the Steganography Analysis and Research Centre,[1] there have been over 1,100 digital steganography applications identified. Most of the digital steganography systems exploit the redundancy of the cover media and rely on the limitations of the human auditory or visual systems. For example, a standard image steganography system uses the least-significant-bit (LSB) substitution technique. Since the difference between 11111111 and 11111110 in the value for red/green/blue intensity is likely to be undetectable by the human eye, the LSB can be used to hide information other than colour, without being perceptable by a human observer.[2]

[1] http://www.sarc-wv.com/ (last verified in June 2013)

[2] The observer may also be a computer program, designed to detect statistical anomalies in the image representation which may indicate the presence of hidden information.

Simmons (1984) formulated steganography as the "Prisoners' Problem". The problem describes a scenario where two prisoners named Alice and Bob are locked up in separate cells far apart from each other and wish to hatch an escape plan. All their communications have to pass through the warden, Willie. If Willie detects any sign of conspiracy, he will thwart their plan by throwing them into high-security cells from which nobody has ever escaped; as long as Willie does not suspect anything, the communication can be put through. So Alice and Bob must find some way for embedding hidden information into their seemingly natural messages. Alice and Bob can succeed if they are able to exchange information allowing them to coordinate their escape without arousing Willie's suspicion. According to information hiding terminology (Pfitzmann, 1996), a legitimate communication among the prisoners is called a *cover datatype*, and a message with embedded hidden information is called a *stego datatype*, where *datatype* stands for "text", "image", "audio", or whatever media is being used. The algorithms that Alice used for creating the *stego datatype* and Bob used for decoding the message are collectively called a *stegosystem*.

A stegosystem has to fulfil two fundamental requirements. The first and foremost requirement is *security*. This means that the stegomedia in which the secret message is hidden must not be suspicious to a human or a computer. The second requirement is *payload capacity*. The payload is the size of the secret message that the sender wishes to conceal and transport relative to the size of the cover media. Since steganography aims at covert information transmission, it requires sufficient embedding capacity. An ideal stegosystem would have a high level of security and large payload capacity. However, there is a fundamental trade-off between security and payload since any attempt to embed additional information in the cover media is likely to increase the chance of introducing anomalies into the media, thus degrading the security level.

A related area to steganography is digital watermarking, in which changes are made to a cover medium in order to verify its authenticity or to show the identity of its owners, for example for copyright purposes (Cox et al., 2008; Shih, 2008). An interesting watermarking application is "traitor tracing", in which documents are changed in order to embed individual watermarks. These marks can then be used to later identify particular documents, for example if a set of documents — identical except for the changes used to embed the watermarks — has been sent to a group of individuals, and one of the documents has been leaked to a newspaper. Both steganography and

watermarking employ steganographic techniques to embed information in cover media. However, steganography aims for the imperceptibility of a secret message to an observer, whereas watermarking tries to mark cover media with information that is robust against modifications. For steganography a user can have the freedom to choose the cover medium to carry messages, whereas for watermarking the cover medium is already decided.

1.2 Linguistic Steganography

A key question for any stegosystem is the choice of cover medium. Given the ubiquitous nature of natural languages and omnipresence of text, text is an obvious medium to consider. For example, a Nazi spy in World War II sent the following message (Kahn, 1967):

Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.

By taking the second letter from each word the following message emerges:

Pershing sails from NY June I

The advantage of this method is that the secret message appears as some normal communication which may not arouse suspicion. However, given the current state-of-the-art of Natural Language Processing (NLP) technology, NLP techniques are not capable of creating meaningful and natural text from scratch and of hiding messages in it. Therefore, most of the existing linguistic stegosystems take already existing text as the cover text, and linguistic properties of the text are used to modify it and hide information.

Figure 1.1 shows the general Linguistic Steganography framework. First, some secret message, represented as a sequence of bits, is hidden in a *cover text* using the embedding algorithm, resulting in the *stego text*.[3] Next, the stego text passes the observer (human or computer), who is happy for innocuous messages to pass between the sender

[3]The message may have been encrypted initially also, as in the figure, but this is not important in this thesis; the key point is that the hidden message is a sequence of bits.

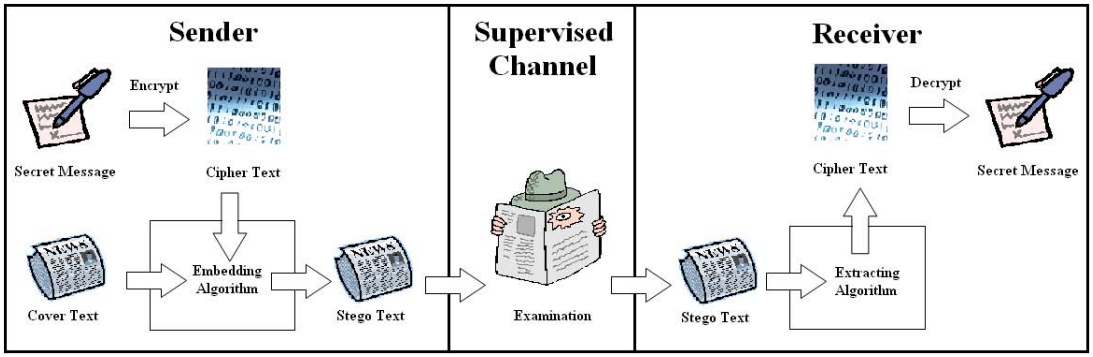


Figure 1.1: The Linguistic Steganography framework

and receiver, but will examine the text for any suspicious looking content. Once the stego text reaches the receiver, the hidden message is recovered using the extracting algorithm.

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. So far, the literature on Linguistic Steganography is small compared with other media (Bergmair, 2007). One of the likely reasons is that it is easier to make changes to images and other non-linguistic media which are undetectable by an observer. Language has the property that even small local changes to a text, e.g. replacing a word by a word with similar meaning, may result in text which is anomalous at the document level, or anomalous with respect to the state of the world. Hence, finding linguistic transformations which can be applied reliably and often is a challenging problem for Linguistic Steganography.

An addition challenge for linguistic steganography is that evaluation of linguistic stegosystems is much more difficult than that of image, audio or video stegosystems because such evaluation requires us to consider many controversial linguistic issues, such as meaning, grammaticality, fluency and style. Unlike some of the NLP tasks, such as machine translation, text summarisation, paraphrasing and simplification which aim at transforming the original text in a grammatical and meaning preserving way, linguistic steganography does not require a stego text to convey the same meaning as its cover text. Instead, linguistic steganography only considers whether the modified text is nat-

ural; hence, a stego text should be evaluated independently of the cover text. Although some computational steganalysis systems have been developed to identify stego text from innocent text using statistical methods, the systems can only be applied to text that undergoes certain linguistic transformations such as translation and lexical substitution, and they are not accurate enough for practical evaluation. Therefore, most of the current linguistic stegosystems were evaluated by human judges (Murphy and Vogel, 2007a,b; Meral et al., 2007; Kim, 2008; Meral et al., 2009; Kim, 2009; Chang and Clark, 2010a), where a human assessor was provided with stego text and was asked to rate or improve the naturalness of the stego text.

1.3 The scope of this thesis

In this thesis I focus on linguistic steganography rather than watermarking, since I am interested in the requirement that any changes to a text must be imperceptible to an observer, as this makes for a strong test of the NLP technology used to modify the cover text. There are various practical security issues in the steganography application that I have chosen to ignore or simplify in order to focus on the underlying NLP techniques. For example, I assume the adversary in the espionage scenario is a human acting passively rather than actively. A passive warden examines all messages exchanged between Alice and Bob but crucially does not modify any message. In other words, I have ignored the possibility of computational steganalysis and steganographic attacks (Fridrich, 2009), via an active warden who deliberately modifies messages in order to thwart any hidden communication. In addition, I do not test the security level of the proposed stegosystems by applying Kerckhoffs' principle (Kerckhoffs, 1883), which states that a method of secretly coding and transmitting information should be secure even if everything about the system, except the key and any private randomizer, is public knowledge. In the proposed stegosystems there are some parameters that can be treated as secret keys only shared between a sender and a receiver. However, I do not measure the secrecy of the hidden information using the keys. Instead, I evaluate the security level of the proposed systems by the naturalness of generated stego text. Another evaluation aspect that I do not emphasize particularly in this thesis is the computational complexity of the proposed stegosystems since steganography traditionally is not an instantaneous communication as the sender and the receiver encode

and decode messages off-line.

In this research I focus on hiding information in English text. However, the proposed methods can also be applied to other languages as long as the same resources and tools are available for the other language, such as synonym dictionaries, n-gram corpora, parsers and word ordering systems. In order to embed messages in a cover document, I exploit three different linguistic transformations to modify the original text. The first transformation I use is lexical substitution which replaces a selected word with a same part-of-speech synonym; the second linguistic transformation deletes unnecessary adjectives in noun phrases; and the third transformation rearranges words in a sentence. For each transformation I develop a checker to certify the naturalness of modified sentences. Note that, in this thesis, I do not investigate the document-level coherence of stego text since this requires sophisticated knowledge of natural language semantics and pragmatics which I consider to be outside the scope of this work. Instead, I tackle the problem of distinguishing the naturalness of a modified sentence in isolation from the rest of a document. Finally, I propose possible secret embedding methods that can work with the linguistic transformation checkers and generate natural stego text.

1.4 The approaches in this thesis

The main objective of this thesis is to explore possible linguistic transformations for the application of steganography. As mentioned previously, the transformation must be applied reliably and often to a cover text in order to fulfil the requirements of security and payload. The following gives a brief overview of the three transformations exploited in the proposed stegosystems and the linguistic transformation checkers that I develop to certify the naturalness of modified sentences (in order to satisfy the security requirement). More details of how each transformation and checker work in the proposed stegosystem are given in Chapter 3, Chapter 4 and Chapter 5.

The first transformation I use is lexical substitution which is a relatively straightforward modification of text. It replaces selected words with the same part of speech (POS) synonyms, and does not involve operating on the sentence structure so the modification is likely to be grammatical.

There are two practical difficulties associated with hiding bits using synonym substi-

tution. The first is that words can have more than one sense. In terms of WordNet (Fellbaum, 1998), which is the electronic dictionary I use, words can appear in more than one synonym set (synset).[4] This is a problem because a word may be assigned different secret bitstrings in the different synsets, and the receiver does not know which of the senses to use, and hence does not know which hidden bitstring to recover. My solution to this problem is a novel vertex colouring method which ensures that words are always assigned the same bitstring, even when they appear in different synsets.

The second problem is that many synonyms are only applicable in certain contexts. For example, the words in the WordNet synset {*bridge*, *span*} share the meaning of "a structure that allows people or vehicles to cross an obstacle such as a river or canal or railway etc." However, *bridge* and *span* cannot be substituted for each other in the sentence "suspension bridges are typically ranked by the length of their main span", and doing so would likely raise the suspicion of an observer due to the resulting anomaly in the text. My solution to this problem is to perform a contextual check which utilises the Google n-gram corpus (Brants and Franz, 2006). I evaluate the substitution checker using the data from the English Lexical Substitution task for SemEval-2007[5] and a human judgement corpus created specifically for the work in this thesis.

The second linguistic transformation I exploit is adjective deletion. If an adjective can be removed from a sentence without significant change in the sentence's naturalness, the adjective can be used as an information carrier in the proposed stegosystem. In order to certify the deletion grammaticality, I only accept a deletion that does not change the CCG categories of the rest of the words. The CCG parser used in this grammaticality check was developed by Clark and Curran (2007). Next, I propose two methods to determine whether the removal of the adjective in a noun phrase is natural to the context. The first method uses the Google n-gram corpus, whereas the second method, which performs better, trains an SVM model that combines n-gram statistics, lexical association measures, entropy-based measures and an n-gram divergence statistic. The methods are evaluated using human judgements of sentence naturalness after removing selected adjectives.

The third transformation is word re-ordering. The sender in the proposed scheme uses

[4] A synset contains a group of synonymous words or collocations that convey similar meaning; different senses of a word are in different synsets.

[5] http://www.dianamccarthy.co.uk/task10index.html (last verified in June 2013)

the n-best list output from a word ordering system as the set of possible alternatives for a cover sentence. Since not all the sentence permutations generated by a word ordering system are grammatical and semantically meaningful, I develop a maximum entropy classifier to distinguish natural word orders from awkward wordings. The proposed classifier is again evaluated by human judgements and also compared to a baseline method using the Google n-gram corpus.

For the proposed linguistic transformation checkers, the more natural the passed sentences are, the less suspicious the stego text may be. In addition, the more sentences that pass the check, the more information carriers the stegosystem can use. Therefore, the evaluation of the proposed linguistic transformation checkers can be seen as an indirect evaluation of the proposed stegosystems. For this reason, the performance of the proposed linguistic transformation checkers is evaluated in terms of precision and recall. Precision is the percentage of sentences judged acceptable by the checker which are determined to be natural by the human judges; recall is the percentage of sentences determined to be natural by the human judges which are also passed by the checker. The interpretation of the measures for a stegosystem is that a higher precision value implies a better security level; whereas a larger recall value means a greater payload capacity.

1.5 Contributions

A significant contribution of the thesis is to advertise the Linguistic Steganography problem to the NLP community. The requirement that any linguistic transformations generate natural sounding and meaningful stego text makes the problem a strong test for existing NLP technology. In this research I explore a variety of linguistic transformations, each with different properties, including lexical substitution, adjective deletion and word ordering, and create novel links between these transformations and linguistic steganography. In addition, the proposed transformation checkers for certifying sentence naturalness potentially benefit not only the steganography application but also other NLP applications, such as sentence compression, text summarization, surface realisation and machine translation, because these tasks all require the generated sentences to be as natural as possible.

Another contribution of the thesis is the evaluation of the proposed stegosystems. The

results suggest that it is possible to develop practical linguistic steganography systems with current NLP techniques. In addition, the collected human judgement corpora for evaluating my systems could be used as a gold standard for other NLP tasks, such as automated sentence scoring systems or sentence compression systems.[6]

The rest of this thesis is organized as follows. Chapter 2 reviews the current state-of-the-art in linguistic steganography and the various linguistic transformations that have been employed in existing stegosystems. In Chapter 3, I describe my first stegosystem using lexical substitution, along with the method for checking substitution quality together with an empirical evaluation. In Chapter 3, I also propose a novel vertex colour coding algorithm to solve the decoding ambiguity problem. Chapter 4 gives the details of my second stegosystem that hides messages in redundant adjectives. Apart from the steganography application, I also show that the adjective deletion technique can be used in a secret sharing scheme, in which the secret message is encoded in two different versions of the carrier text, with different adjectives deleted in each version. In Chapter 5, I present details of my word ordering-based stegosystem and the maximum entropy classifier for determining the naturalness of sentence permutations. Moreover, I explain how the word ordering transformation can be used with existing translation-based secret embedding algorithms. Finally, Chapter 6 gives some conclusions and future research directions.

[6]The human judgement corpora are available at www.cl.cam.ac.uk/∼cyc30 (last verified in June 2013).

Chapter 2

Background

This chapter reviews existing linguistic stegosystems. Under my interpretation of the term Linguistic Steganography, I am only concerned with stegosystems which make changes that are linguistic in nature, rather than operating on superficial properties of the text, e.g. the amount of white space between words (Por et al., 2008), font colors (Khairullah, 2009), or relying on specific file formats, such as ASCII or HTML (Bennett, 2004; Shahreza, 2006).

Most of the existing stegosystems consist of three independent modules — linguistic transformation, data encoding and text selection — as shown in Figure 2.1. As explained in Chapter 1, in order to embed messages, a cover text must provide information carriers that can be modified to represent the secret, and the modification must be natural to an observer. This step is called linguistic transformation. According to the linguistic transformation used in a stegosystem, we can classify existing work into three major categories: lexical or phrase substitutions, syntactic transformations, and semantic transformations. After generating different versions of the cover text, an encoding method is used to assign bitstrings to the alternatives, which is called data encoding. The final phase is text selection which chooses the alternative representing the secret bitstring as the stego text. If there is no alternative associated with the secret bitstring, the secret embedding fails. Therefore, it is important to generate sufficient alternatives as well as to efficiently encode each option. The convenient modularity between the linguistic transformation and data encoding allows a transformation to be combined with different encoding algorithms, although it may put some constraints on what method can be used. I will refer back to Figure 2.1 when describing the proposed

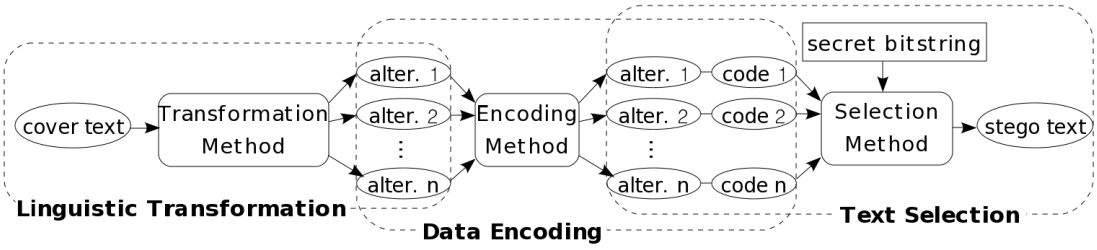


Figure 2.1: Three modules in a linguistic stegosystem

stegosystems in Chapter 3, Chapter 4 and Chapter 5. In the following sections, I first explain the three linguistic transformation categories mentioned above and then introduce several encoding methods. In addition, I summarise the evaluation methods and performance of existing linguistic stegosystems.

2.1 Linguistic transformations

In this section, I explain the three linguistic transformation categories — lexical or phrase substitutions, syntactic transformations, and semantic transformations — that have been used in existing stegosystems to modify cover text. For each transformation, some examples are provided to demonstrate the text manipulation.

2.1.1 Lexical and phrase substitutions

Lexical substitution is a relatively straightforward modification of text. It replaces selected words with synonyms, without operating on the sentence structure, so the modification is likely to be grammatical. There are a few electronic dictionaries available that are designed to capture various lexical relationships between words and serve as lexical reference systems (Fellbaum, 1998; Schuler, 2005). One of the most well-known electronic dictionaries is WordNet (Fellbaum, 1998) in which English nouns, verbs, adjectives and adverbs are categorized into *synonym sets* (synsets). Words in the same synset have the same or similar meaning and in principle can be substituted with each other. For example, a search result of the word *marry* in WordNet 3.1 is summarized in Table 2.1. According to this table, we can change the sentence "The minister will marry us on Sunday" to "The minister will wed us on Sunday" without

marry (verb)
gloss: take in marriage synset: marry, get married, wed, conjoin, hook up with, get hitched with, espouse
gloss: perform a marriage ceremony synet: marry, wed, tie, splice

Table 2.1: Synsets of the word *marry* in WordNet 3.1

introducing much semantic difference since *marry* and *wed* express a similar lexical concept in this context.

There are three main challenges when using lexical substitution as the linguistic transformation. The first challenge is word-category disambiguation, which marks up a word with a particular part-of-speech (POS) based on both its definition as well as the context. For example, *fast* is an adverb in the phrase "hold fast to the rope", an adjective in the phrase "a fast car", and a verb in the phrase "Catholics fast during Lent". Existing POS taggers have achieved 97% accuracy on the Penn Treebank (Toutanova et al., 2003; Shen et al., 2007; Spoustová et al., 2009; Søgaard, 2010) and are widely used in lexical substitution-based stegosystems (Chapman et al., 2001; Bolshakov, 2004; Taskiran et al., 2006; Topkara et al., 2006c,a; Chang and Clark, 2010b).

The second challenge is word-sense disambiguation, which identifies the sense of a word in context (if the word has more than one meaning) so the correct synset can be used. For example, according to the context, *bottom* means "a cargo ship" instead of "the lower side of anything" in the sentence "we did our overseas trade in foreign bottoms", and therefore it can be replaced with *freighter* but not *undersurface*. The first lexical substitution stegosystem was proposed by Winstein (1999). In order to handle the fact that a word may appear in more than one synset in WordNet, Winstein defines "interchangeable words" as words that belong to the same synsets, and only uses these words for substitution. For example, *marry* and *wed* in Table 2.1 are interchangeable words since they are always synonyms even under different meanings. Any words that are not interchangeable are discarded and not available for carrying information. Winstein (1999) calculates that only 30% of WordNet can be used in such a system.

The main purpose of linguistic transformations is to generate unsuspicious alternatives for a cover sentence. Although replacing a word with its synonym that conveys the same concept may preserve the meaning of the sentence, much of the time there are still

semantic and pragmatic differences among synonyms. For example, the synset {*chase*, *trail*, *tail*, *tag*, *dog*, *track*} means "go after with the intent to catch". However, an awkward sentence would be generated if we replace *chase* with *dog* in the sentence "the dogs chase the rabbit". Hence, it is important to check the acceptability of a synonym in context. Bolshakov (2004) used a collocation-based test to determine whether a substitution is applicable in context. Taskiran et al. (2006) attempted to use context by prioritizing the alternatives using an n-gram language model; that is, rather than randomly choose an option from the synset, the system relies on the language model to select the synonym. In Chapter 3, I describe how the proposed lexical substitution-based stegosystem uses the Google n-gram corpus to certify the naturalness of the proposed substitution.

Similar to synonym substitution, text paraphrasing restates a phrase using different words while preserving the essential meaning of the source material being paraphrased. In other words, text paraphrasing is multi-word substitution. For example, we can paraphrase "a high percentage of" by "a large number of" in the sentence "a form of asbestos has caused a high percentage of cancer deaths". However, text paraphrasing may have more effect on the grammaticality of a sentence than lexical substitution. In my Masters research (Chang and Clark, 2010a; Chang, 2010) I developed a stegosystem exploiting a paraphrase dictionary (Callison-Burch, 2008) to find potential information carriers, and used the Google n-gram corpus and a CCG parser (Clark and Curran, 2007) to certify the paraphrasing grammaticality. Similar techniques will be used in the proposed stegosystems to check the applicability of a linguistic transformation.

2.1.2 Syntactic transformations

Syntactic transformation methods are based on the fact that a sentence can be transformed into more than one semantically equivalent syntactic structure, using transformations such as passivization, topicalization and clefting. Table 2.2 lists some of the common syntactic transformations in English.[1]

The first syntactic transformation method was presented by Atallah et al. (2000). Later, Atallah et al. (2001) generated alternative sentences by adjusting the structural properties of intermediate representations of a cover sentence. In other words, instead of

[1]The categories of transformations are adopted from Topkara et al. (2005).

Transformation	Original sentence	Transformed sentence
Passivization	The dog kissed Peter.	Peter was kissed by the dog.
Topicalization	I like pasta.	Pasta, I like.
Clefting	He won a new bike.	It was a new bike that he won.
Extraposition	To achieve that is impossible.	It is impossible to achieve that.
Preposing	I like cheese bagels.	Cheese bagels are what I like.
There-construction	A cat is in the garden.	There is a cat in the garden.
Pronominalization	I put the cake in the fridge.	I put it there.
Fronting	"What!" Peter said.	"What!" said Peter.

Table 2.2: Some common syntactic transformations in English

performing lexical substitution directly on the text, the modifications are performed on the syntactic parse tree of a cover sentence. Murphy (2001), Liu et al. (2005), Topkara et al. (2006b), Meral et al. (2007), Murphy and Vogel (2007b) and Meral et al. (2009) all belong to this syntactic transformation category. After manipulating the syntactic parse tree, the modified deep structure form is converted into the surface structure format via language generation tools.

Aside from the above systems, Wayner (1995) and Chapman and Davida (1997) proposed mimicry text approaches associated with linguistic syntax. These two stegosystems generate stego text from scratch instead of modifying an existing text. Wayner (1995) proposed a method with his context-free mimic function (Wayner, 1992) to generate a stego text that has statistical properties close to natural language. The context-free mimic function employs a probabilistic grammar-based model to structure the stego text. Since the mimicry method only puts emphasis on the syntactic structure of a sentence, it is likely to generate nonsensical stego text which is perceptible by humans. Chapman and Davida (1997) developed a stegosystem called *NICETEXT* that generates stego sentences using style sources and context-free grammars to simulate certain aspects of writing style. Comparing with Peter Wayner's mimicry method, the stego text generated by *NICETEXT* is more natural in terms of the semantics, but still not at a level that would be suitable for practical steganography.

2.1.3 Semantic transformations

The semantic transformation is the most sophisticated approach for linguistic steganography, and perhaps impractical given the current state-of-the-art for NLP technology. It requires some sophisticated tools and knowledge to model natural language semantics and to evaluate equivalence between texts in order to perform deep semantic manipulations. For example, consider the following sentences:

Bond takes revenge for Vesper's death.
Vesper's death is avenged by Bond.
007 takes revenge for Vesper's death.

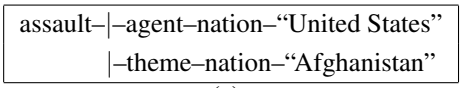
The idea is to define the semantic representation in such a way that the translation from any of the above sentences to their semantic representations would yield the same form. In this manner, the meaning of the cover sentence can be expressed in another natural language text. For this to be successful for the example, we would have to understand sentences in different voice, such as active and passive voice, and make use of some world knowledge, such as the fact that the codename of James Bond is *007*.

The work of Atallah et al. (2002) used semantic transformations and aimed to output alternatives by modifying the text-meaning representation (TMR) tree of a cover sentence. The modifications include pruning, grafting, or substituting the tree structure with information available from ontology resources. A linguistic ontology is a formal knowledge representation of the world; a conceptualization of entities, events, and their relationships in an abstract way. For example, Figure 2.2 taken from Atallah et al. (2002) shows parts of the ontological semantics for *Afghanistan* that are structured in a tree-like hierarchy.[2] An ontology provides concepts that are used to define propositions in TMR. TMR of a natural language expression can show information such as clause relationship, author attitude, topic composition, and so on. It is constructed through mapping lexical items and events that are referred in the expression to their ontology concepts. Figure 2.3(a) shows the TMR of the sentence "the United States are attacking Afghanistan". A modification of the tree can be performed by grafting additional semantic information of *Afghanistan* as shown in Figure 2.3(b), yielding the alternative sentence "the United States are attacking Afghanistan, which is ruled by Mullah Mohamme Omar". Vybornova and Macq (2007) also exploited the linguistic

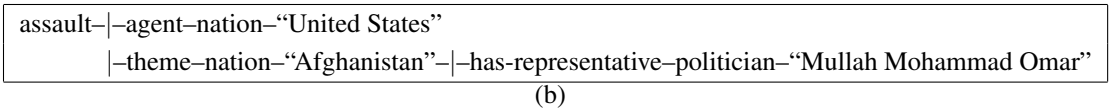
[2] Afghanistan was ruled by Mullah Mohammed Omar at the time of Atallah et al. (2002).

Afghanistan (nation)	
borders-on	China, Iran, Pakistan, Tajikistan, Uzbekistan
has-currency	afghani
has-member	Pashtun, Tajik, Hazara, Uzbek
has-representative	Mullah Mohammad Omar

Figure 2.2: Parts of the ontological semantics for *Afghanistan*



(a)



(b)

Figure 2.3: An example of the TMR tree modification taken from Atallah et al. (2002)

phenomenon of presupposition, with the idea that some presuppositional information can be removed without changing the meaning of a sentence.

Another group of studies aims to use machine translated sentences as the alternatives. The main advantage of using machine translated text is that translations are seldom perfect, and therefore it is hard to determine whether the anomalies are introduced by a translation system or due to the camouflage of secret information.

The first translation-based stegosystem was proposed by Grothoff et al. (2005). In their method, the sender uses a set of machine translation systems to generate multiple translations for a given cover sentence. Stutsman et al. (2006) also utilised multiple translation systems to output alternatives for a cover sentence. Since Grothoff et al. (2005) and Stutsman et al. (2006) used multiple machine translation systems to generate alternative translations, the selected stego sentences may not have a 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) and Venugopal et al. (2011) used 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.

So far I have explained different transformation methods for generating alternatives for an input text. This procedure is seen as the Linguistic Transformation module in Figure 2.1. After deriving alternatives for a cover text, the Data Encoding module maps each alternative to a code that can be used to represent a secret bitstring. In the next section, I will explain several encoding methods that have been used in existing stegosystems.

2.2 Encoding methods

In order to demonstrate each encoding method, let me assume the cover sentence is "we finish the charitable project" and the transformation applied to the text consists of simply replacing a word with its synonym. The alternatives for the cover text arise from replacing *finish* with *complete*, and replacing *project* with *labour*, *task* or *undertaking*. In the following I introduce four encoding methods that assign bitstrings to the alternative candidates. Note that, as mentioned at the beginning of the Chapter, linguistic transformations are largely independent of the encoding methods and therefore the encoding methods explained here are not restricted to lexical substitutions. After encoding the alternatives, the secret can be embedded by selecting alternatives that directly associate with the secret bitstring.

2.2.1 Block code method

For a set with cardinality n, the block code method assigns m-bit binary codes from 0 to (2^m-1) to the elements in the set, where $2^m \leq n$. For example, the synonym set $\{$*complete*, *finish*$\}$ has cardinality $n = 2$ so 1-bit binary codes *0* and *1* are assigned to *complete* and *finish*, respectively. Since the synonym set $\{$*labour*, *project*, *task*, *undertaking*$\}$ has cardinality $n = 4$, the block code method can use either 1-bit or 2-bit codes to represent the words as shown in Figure 2.4. When 1-bit codes are used, both *labour* and *task* represent code *0*, and both *project* and *undertaking* represent code *1*; when 2-bit codes are used, the 4 words are assigned different codes *00*, *01*, *10* and *11*. The advantage of using 1-bit codes is that the cover word *project* needs to be replaced with its synonym only 50% of the time, whereas the 2-bit scheme has a 75% chance of modifying the cover word. However, 1-bit codes embed less information.

1-bit	Word
0	*complete*
1	*finish*

We the charitable .

1-bit	2-bit	Word
0	00	*labour*
1	01	*project*
0	10	*task*
1	11	*undertaking*

Figure 2.4: An example of the block code method

Code	Word
0_2	*complete*
1_2	*finish*

We the charitable .

Code	Word
0_4	*labour*
1_4	*project*
2_4	*task*
3_4	*undertaking*

Figure 2.5: An example of the mixed-radix number method

Hence, there is a trade-off between security and payload capacity. It is worth noting that, in this simple scheme, each block code representation has the same probability of being chosen, even though native speakers might have a preference for the choice of synonyms, which would be security-relevant.

2.2.2 Mixed-radix number method

In a mixed-radix number system, the numerical base differs from position to position. For example, 8 hours, 41 minutes and 21 seconds can be presented relative to seconds in mixed-radix notation as: $8_{(24)}41_{(60)}21_{(60)}$, where each digit is written above its associated base. The numerical interpretation of a mixed-radix number $a_{n(b_n)}a_{n-1(b_{n-1})}...a_{0(b_0)}$ is $a_nb_{n-1}b_{n-2}...b_0 + a_{n-1}b_{n-2}b_{n-3}...b_0 + ... + a_1b_0 + a_0$, and any number can be uniquely expressed in mixed-radix form (Soderstrand et al., 1986).

Figure 2.5 shows the use of the mixed-radix number method with the lexical substitution example which is described in Bergmair (2004). Firstly, the words in the synsets {*complete*, *finish*} are encoded with *0* and *1* with base 2, and the words in the synset {*labour*, *project*, *task*, *undertaking*} are encoded with *0*, *1*, *2* and *3* with base 4. Therefore, the combinations of the substitutions yield the 2-digit mixed-radix numbers from 0_20_4 to 1_23_4, which are equal to the decimal numbers 0 to 7. Assume the secret bit-

We

Code	Word	Prob.
0	*complete*	0.77
1	*finish*	0.23

the charitable

Code	Word	Prob.
110	*labour*	0.05
0	*project*	0.69
10	*task*	0.25
111	*undertaking*	0.01

.

Figure 2.6: An example of the Huffman code method

string to be embedded is *110* which can be seen as the binary number for six. Since "we finish the charitable task" represents the mixed-radix number $1_2 2_4$, which is the decimal number 6, this sentence will be the stego sentence that embeds the secret. Like the Block code method, each mixed-radix number representation has the same probability of being chosen, which may cause some security concern. To solve this issue, one can utilise variable-length code methods described in the next section.

2.2.3 Huffman code method

Figure 2.6 demonstrates the use of variable-length codes, in the form of the Huffman code (Huffman, 1952), for representing words in a synset. Assuming there are utility rates for each word, then the Huffman algorithm determines a way to produce a variable-length binary string for each word. More importantly, it does so in such a way that an optimal encoding is created; that is, words with higher utility rates have shorter codes while words with lower utility rates get longer codes. Thus, words frequently used by native speakers are more likely to be chosen by the stegosystem (assuming utility corresponds to frequency). The process shown in Figure 2.7 begins with leaf nodes each containing a word along with its associated probability (Figure 2.7a). The two nodes with the smallest probabilities are then chosen to become the children of a new node whose probability is the sum of the probabilities of its children (Figure 2.7b). The newly created left and right branches are assigned bit 0 and 1, respectively. Now only the newly created node is taken into consideration instead of its children. The procedure is repeated until only one node remains, thereby constructing the Huffman tree (Figure 2.7d). To determine the binary code assigned to a particular word, we start from the root node and gather the bits on the path to the leaf node connected to that word. In this example we can see that "project" has the highest probability among

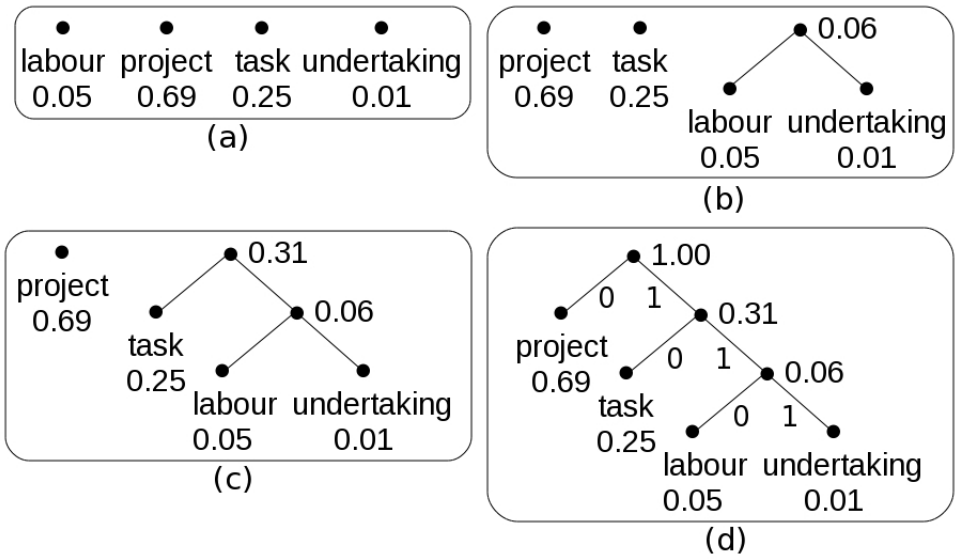


Figure 2.7: The process of constructing a Huffman tree

words in the same synset and is encoded with the shortest code. Thus, it is more likely to match the secret bitstring.

As described in Section 2.1.2, Wayner (1995) generates stego text by exploiting a probabilistic context-free grammar. His method creates a Huffman tree for each set of productions that expand the same non-terminal symbol. In this way, each production has its own Huffman code representation as shown in Figure 2.8(a). Then, we begin with a designated start-symbol *S*, and expand a non-terminal symbol by choosing the production whose Huffman code representation is identical to the portion of the secret bitstring. The procedure is repeated until a grammatical message is generated. In the embedding example given in Figure 2.8(b), the secret bitstring is *1101110* and a symbol "•" is used to indicate the current bit in reading the string. At the beginning, the prefix string of the secret message •1101110 is "1" which is associated with the second production, so the start-symbol *S* is expanded to *AC*. Now, the prefix string of the message 1•101110 becomes "10". The fourth production is applied, and a string "You *C*" is generated. Next, we see the prefix string "1" in the message 101•110, and therefore, the output string turns into "You won the *D*". Finally, the end of the secret message 1101110• is reached, and a stego sentence "You won the championship" is generated. Theoretically, the block code representation or the mixed-radix technique explained in the previous sections can be utilized in Wayner's stegosystem.

Rule No.	Rule	Code	Probability
1.	$S \rightarrow AB$	0	0.3
2.	$S \rightarrow AC$	1	0.7
3.	$A \rightarrow$ I	0	0.4
4.	$A \rightarrow$ You	10	0.3
5.	$A \rightarrow$ He	110	0.15
6.	$A \rightarrow$ She	111	0.15
7.	$B \rightarrow$ lost	0	0.4
8.	$B \rightarrow$ won	1	0.6
9.	$C \rightarrow$ lost the D	0	0.4
10.	$C \rightarrow$ won the D	1	0.6
11.	$D \rightarrow$ game	0	0.4
12.	$D \rightarrow$ match	10	0.3
13.	$D \rightarrow$ championship	110	0.2
14.	$D \rightarrow$ competition	111	0.1

(a)

Position	Prefix	Rule	Output
•1101110	1	2.	AC
1•101110	10	4.	You C
110•1110	1	10.	You won the D
1101•110	110	13.	You won the championship

(b)

Figure 2.8: An example of the Wayner (1995) mimicry method

2.2.4 Hash function method

For the block code method, the mixed-radix number approach and the Huffman code representation, the encoding process is dependent on knowing all the alternatives (e.g. the synset). Hence, in order to extract the code assigned to the stego text during the secret recovery process, all the alternatives must be known to the receiver as well. Note that the receiver does not need to know the original cover text. However, not all the linguistic transformations can meet this requirement. For example, if the sender encodes the four best machine translations of the cover sentence using block coding and sends the translation that encodes the secret bits to the receiver, it is unlikely that the receiver can retrieve the four best machine translations without knowing the original cover sen-

tence. Thus, the secret recovery fails. For this reason, Grothoff et al. (2005), Stutsman et al. (2006), Meng et al. (2011) and Venugopal et al. (2011) used a hash function to map a translation to a code which is independent of the rest of the alternatives.

Venugopal et al. (2011) defined a random hashing operation that maps a translation to a bit sequence of fixed length. Venugopal et al. stated that a good hash function should produce a bitstring whose 0s and 1s are generated with equal probability. Grothoff et al. (2005) used the least significant bit of a translation hash bitstring as the code represented by the translation, and therefore the payload capacity is 1 bit per sentence. Later Stutsman et al. (2006) improved the payload capacity of the hash function encoding scheme by introducing a footer (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 footer bits. The lowest h bits of a translation hash bitstring are the footer bits and the lowest [h+1, h+b] bits are the code. For example, assume $h = 2$; a hash bitstring "...10111" has footer bits *11* to indicate a 3-bit code is carried by this translation, and the three bits are *101*.

The problem of using a hash function is that the generation of a desired bitstring cannot be guaranteed. For example, in the case where both the two translations happen to have the least significant bit 1, choosing either of the two translations as the stego sentence cannot embed a secret bit 0. 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.

So far I have introduced different linguistic transformations used to produce alternatives for a cover text as well as some encoding methods that are used to assign a bitstring to a candidate. The final procedure is Text Selection in which an alternative that encodes the secret bits is chosen as the stego text. We can see that the quality of a stego text mainly relies on the quality of the applied linguistic transformation, typically requiring sophisticated NLP tools and resources to produce a realistic stego text. 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.

2.3 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.

For the work presented in this thesis, I use human judgements to evaluate the proposed stegosystem, as this is close to the linguistic steganography scenario defined in Section 1.3 where I assume the adversary is a human acting passively. I asked subjects to judge the *naturalness* of a given sentence rather than linguistic acceptability and meaning preservation of the cover text. As explained in Section 1.2, only the stego text is observed by the adversary so it does not matter whether the stego text preserves the meaning of the cover text as long as the stego text is reasonable and meaningful. For example, changing a cover sentence “I like cats” to a stego sentence “I like dogs” does not preserve the original meaning, but since “I like dogs” is just as natural as “I like cats”, it should not draw the adversary’s attention. In addition, I do not evaluate the linguistic acceptability of a stego sentence because it is possible that a sentence often used by native English speakers is ungrammatical, for example “long time no see”.

Therefore, throughout the thesis, I use "naturalness" as the main criterion along with some examples as the guideline for human evaluations.

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 to 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, despite the large number of synonyms in English. The payload attained by syntactic transformations was around 0.5 to 1.0 bits per sentence. For example, both Atallah et al. (2001) and Topkara et al. (2006b) achieved an embedding payload of 0.5 bits per sentence, and Meral et al. (2009) reported the data embedding rate of their system as 0.81 bits per sentence. Since the ontological semantic transformation is currently impractical, the empirical payload is not available. Another semantic method (Vybornova and Macq, 2007) that aims at modifying presuppositional information in text achieved a payload of 1 bit per sentence through the use of a secret key to indicate sentences with or without presupposition information. Stutsman et al. (2006) showed that their translation-based stegosystem has a payload of 0.33 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 terms in a medical paper which cannot be changed or even cannot be found in a standard dictionary. To the best of my knowledge, there is no study on the

practical issue of using different types of cover text for the steganography application.

In this thesis, I indirectly evaluated the security levels of my three stegosystems by measuring the precision and recall of the proposed transformation checkers. The precision of a linguistic checker implies the security level of a stegosystem since the more natural the passed alternatives are, the less suspicious the stego text is likely to be. In contrast, the recall of a checker implies the system's payload capacity since the more sentences that pass the check, the more information carriers the stegosystem can use, and therefore, the more data can be embedded in the text. For each stegosystem, I demonstrate the trade-off between the security and payload using precision and recall diagrams so readers can observe the embedding rates at different security levels.

Chapter 3

Lexical substitution

In this chapter I introduce my first linguistic stegosystem based on lexical substitution.[1] In the original work on linguistic steganography in the late 1990s, Winstein (1999) proposed an information hiding algorithm using a block coding method to encode synonyms, so that the selection of a word from a synset directly associates with part of the secret bitstring. An example of Winstein's system can be found in Figure 2.4 in the previous chapter. In his system, a sender and a receiver share the same coded synonym dictionary as the secret key. To recover the hidden message, the receiver first seeks words in the stego text which can be found in the shared dictionary. Those words are information carriers, and therefore the codes assigned to them are secret bitstrings. Note that the receiver does not need the original cover text to recover the secret message.

One of the problems faced by a synonym-based stegosystem is that many words are polysemous, having more than one sense, and this may cause ambiguities during the secret recovery stage. In WordNet a synset contains words expressing a similar concept, and a word may appear in more than one synset. For example, both *marry* and *wed* appear in the two synsets in Table 2.1. Figure 3.1 shows what happens when the block coding method is applied to the two overlapping synsets, assuming the stego sentence received by the receiver is "the minister will wed us on Sunday". Note that I only take single word substitution into consideration in order to avoid the confusion of finding information carriers during the secret recovering phase. For example, if the

[1] Most of the content in this chapter was published in Chang and Clark (2010b).

Synset 1		Synset 2	
Word	Code	Word	Code
conjoin	00	marry	00
espouse	01	splice	01
marry	10	tie	10
wed	11	wed	11

Figure 3.1: An example of decoding ambiguity using lexical substitution

cover word *espouse* is replaced by *hook up with*, the receiver would not know whether the secret message is embedded in the word *hook* or the phrase *hook up with*. After deleting multi-word synonyms, words in the two synsets are sorted alphabetically and assigned 2-bit codes. As can be seen in Figure 3.1, *marry* is represented by two different codewords and thus the secret bitstring cannot be reliably recovered, since the receiver does not know the original cover word or the sense of the word.

In order to solve the problem of words appearing in more than one synonym set, Winstein defines *interchangeable words* as words that are always synonyms to each other even under different meanings (i.e. always appear together in the same synsets). For example, *marry* and *wed* are interchangeable words under Winstein's definition. The advantage in this approach is that interchangeable words always receive the same codeword. The disadvantage is that many synonyms need to be discarded in order to achieve this property. Winstein reported that only 30% of words in WordNet are interchangeable words. In addition, as explained in Section 2.1.1, many synonyms are only applicable in certain contexts. However, in Winstein's steganography scheme there is no method to filter out unacceptable substitutions, so the generated stego text may be unnatural and arouse suspicion in others.

Another synonym substitution-based stegosystem was proposed by Bolshakov (2004), who applies *transitive closure* to overlapping synsets to avoid the decoding ambiguity. Applying transitive closure leads to a merger of all the overlapping synsets into one set which is then seen as the synset of a target word. Consider the overlapping synsets in Figure 3.1 as an example. After applying transitive closure, the resulting set is {*conjoin*, *espouse*, *marry*, *splice*, *tie*, *wed*}. The disadvantage of Bolshakov's system is that all words in a synonym transitive closure chain need to be considered, which can lead to very large sets of synonyms, many of which are not synonymous

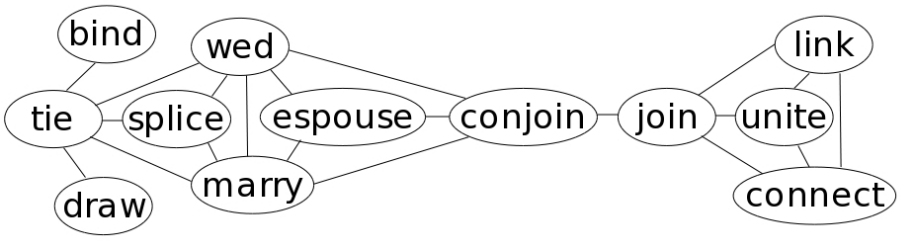
with the original target word. For this reason, Bolshakov used a collocation-based test to remove unsuitable words after merging the synsets. Finally, the collocationally verified synonyms are encoded using the block coding method. Note that in Bolshakov's system it is possible to replace an original word with a non-synonymous word if the non-synonymous word passes the collocation-based test.

Similar to Bolshakov's method, my approach takes words in a synonym transitive closure chain into consideration and assigns a score to each word using the proposed substitution checker. A score threshold is applied to eliminate low-score words; that is, the remaining words are those in the synonym transitive closure chain that are acceptable to the context. More details of the proposed substitution checker will be described later in the chapter. I then construct a synonym graph which has a vertex for each remaining word and an undirected edge for every pair of words that share the same meaning. After constructing the synonym graph, I use a novel coding method based on vertex colouring to assign codes to every word in the graph.

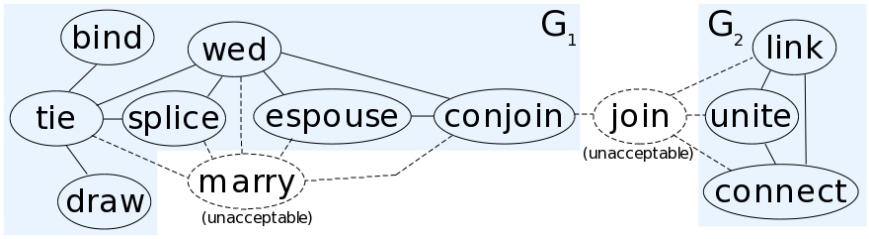
A crucial difference from Bolshakov's method is that in my approach the sender only considers words that are synonymous with the cover word as alternatives, even though the other words in the synonym graph can also fit into the context. The reason for also including non-synonymous words during the encoding is because the receiver does not know the cover word and, therefore, I need a method to ensure that the receiver is encoding the same list of words, namely the same synonym graph, as the sender during the secret recovery. In other words, the sender and the receiver must derive the same synonym graph given that the sender knows the cover word and the receiver knows the stego word.

Figure 3.2(a) shows a synonym graph constructed from a synonym transitive closure chain which contains six synsets: {*bind*, *tie*}, {*tie*, *draw*}, {*tie*, *wed*, *splice*, *marry*}, {*marry*, *wed*, *espouse*, *conjoin*}, {*conjoin*, *join*}, {*join*, *link*, *unite*, *connect*}. Assume the cover word is *conjoin*. In Bolshakov's system, there is a chance of replacing *conjoin* with *draw* which is three steps away from the original word in the graph; while in my method I only consider a cover word's synonyms as alternatives, that is, *conjoin* is only allowed to be replaced by *wed*, *espouse*, *marry* or *join*. Note that I have not applied the substitution check in this example.

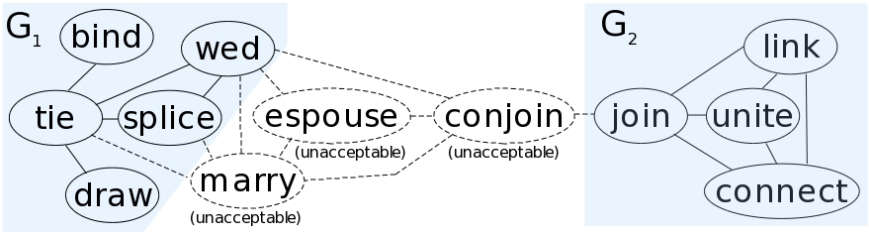
Now let me apply the substitution check to words in the synonym transitive closure chain, and suppose *join* and *marry* do not pass the check. Figure 3.2(b) shows the



(a) An unchecked synonym graph



(b) Synonym graphs derived after substitution checking



(c) Another example of checked synonym graphs

Figure 3.2: Synonym graphs with and without the substitution check

two disconnected synonym graphs G_1 and G_2 derived from the checked pool. The two synonym graphs are then encoded independently. In other words, the encoding of G_1 does not affect the codes assigned to the words in G_2. Since *conjoin* is the cover word, the system may replace *conjoin* with either *wed* or *espouse*, or keep the original word depending on the encoding of G_1 and the secret bits. Assume *wed* is chosen as the stego word. In order to work out the embedded message, the receiver needs to construct and encode the same graphs as that generated by the sender. The decoding process starts from extracting the synonym transitive closure chain of *wed*, and then applying the substitution checker to the pool to filter out unacceptable words. Since the remaining words are the same as those used by the sender, the receiver can successfully extract the secret bits after constructing and encoding the synonym graphs.

Since the proposed substitution checker measures the acceptability of a word according to the context, the synonym graph for a target word varies depending on its context. Let us consider another case where the cover word is still *conjoin*, but this time the substitution checker determines that *conjoin*, *espouse* and *marry* are not acceptable to the context. Figure 3.2(c) shows the corresponding synonym graphs of the remaining words. In this case, the applicable alternatives are either *wed* or *join* since they are synonyms of *conjoin*. As mentioned previously, disconnected graphs are encoded independently. Therefore, it is possible that both *wed* and *join* are assigned the same codeword which does not match the secret bits. If neither of the synonyms can be used as the stego word, the sender will keep the original word and send *conjoin* to the receiver. During the decoding process, the receiver should be able to know that *conjoin* fails the check and thus does not carry any message. In contrast, if *wed* and *join* are encoded by different codewords, say 0 and 1, respectively, the system can choose the one that represents the secret bit as the stego word.

The following sections are organised so that the basic substitution checker using the Google n-gram corpus (Brants and Franz, 2006) is described first. Then I introduce the α-skew divergence measure (Lee, 1999) that can be combined with the basic n-gram method. The proposed checkers are evaluated using data from the SemEval lexical substitution task (McCarthy and Navigli, 2007), which is independent of the steganography application. I also perform a more direct evaluation of the imperceptibility for the steganography application by asking human judges to evaluate the naturalness of sentences. After explaining the linguistic transformation module in my stegosystem, I proceed with the data encoding module and present the vertex colouring coding method. Finally I use an example to demonstrate the complete stegosystem.

3.1 Substitution Checkers

The aim of the proposed checkers is to filter out inapplicable substitutes given the original word in context. The substitution checkers can work not only with the proposed linguistic stegosystem, but can also be integrated into other synonym substitution-based applications to certify the transformation quality.

3.1.1 N-gram count method

The basic checking method, referred to as NGM, utilises the Google n-gram corpus to calculate a substitution score for a candidate word in context. The Google n-gram corpus was collected by Google Research for statistical language modelling, and has been used for many 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). It contains frequency counts for n-grams from uni-grams through to five-grams obtained from over 1 trillion word tokens of English Web text. Only n-grams appearing more than 40 times were kept in the corpus.

The checking method first extracts contextual bi- to five-grams around the word to be tested and uses tools of Minnen et al. (2001) for correcting the form of an indefinite and a verb's tense. For example, if the word to be tested is *maverick* and it is going to replace *unorthodox* in the phrase "the help of an *unorthodox* speech therapist named Lionel", the indefinite *an* will be corrected as *a* when extracting contextual n-grams. As another example, assume the word to be replaced is *bleach* in the original phrase "he might be *bleaching* his skin", then a verb substitute *decolour* will be corrected as *decolouring* since the original word is in the progressive tense.

After extracting contextual bi- to five-grams, the checking method queries the n-gram frequency counts from the Google n-gram corpus. For each n, the total count f_n is calculated by summing up individual n-gram frequencies, for every contextual n-gram containing the candidate word. I define a *count function* $Count(w) = \sum_{n=2}^{5} log(f_n)$ where *log(0)* is defined as zero. If $Count(w)=0$, I assume the word w is unrelated to the context and therefore is eliminated from the synonym transitive closure chain. After calculating *Count(w)* for each word in the pool, the word that has the highest count is called *the most likely word*, and its count is referred as max_{count}. The main purpose of having max_{count} is to score each word relative to the most likely substitute in the chain, so even in less frequent contexts which lead to smaller frequency counts, the score of each word can still indicate the degree of feasibility. I also need to use the most likely word, rather than the original cover word, since the receiver does not have access to the cover text when applying the check. The most likely word in the context may be the original word or another word in the synonym transitive closure chain. The substitution score is defined as $Score_{NGM}(w) = \frac{Count(w)}{max_{count}}$. The hypothesis is that a word

n-gram	frequency	f_n
was *clever*	40,726	f_2 = 302,492
clever and	261,766	
He was *clever*	1,798	f_3 = 8,072
was *clever* and	6,188	
clever and independent	86	
He was *clever* and	343	f_4 = 343
was *clever* and independent	0	
clever and independent and	0	
He was *clever* and independent	0	f_5 = 0
was *clever* and independent and	0	
clever and independent and proud	0	
$Count(clever) = \log(f_2) + \log(f_3) + \log(f_4) + \log(f_5) = 28$		
$Score_{NGM}(clever) = \frac{Count(clever)}{max_{count}} = 0.93 > \text{threshold}(0.9)$		

Figure 3.3: An example of the proposed NGM method

with a high score is more suitable for the context, and I apply a threshold so that words having a score lower than the threshold are discarded.

Figure 3.3 demonstrates an example of calculating the substitution score for the candidate word *clever* which is going to replace *bright* in the cover sentence "he was *bright* and independent and proud." First of all, various contextual n-grams are extracted from the sentence and the Google n-gram corpus is consulted to obtain their frequency counts. $Count(clever)$ is then calculated using the n-gram frequencies. Suppose the threshold is 0.9, and the max_{count} is 30 from the synonym transitive closure chain. The substitution score $Score_{NGM}(clever)$ is 0.93, and so the word *clever* is determined as acceptable for this context and is kept in the pool.

One disadvantage of using n-gram statistics is that high-frequency n-grams may dominate the substitution score, especially lower-order n-grams. For example, *even* is not a good substitute for *eve* in the sentence "on the eve of the wedding, Miranda tells Mr. Big that marriage ruins everything", but it still has a reasonably high score of 0.74 since the bi-grams "the even" and "even of" have high frequency counts compared with those of the four-grams and five-grams. As a way of overcoming this problem, I take the n-gram distributional similarity between a most likely word and a candidate

substitute in context into consideration using alpha-skew divergence as explained in the next section. I assume that an acceptable substitute should have a similar n-gram distribution to the most likely word across the various n-gram counts.

3.1.2 Contextual α-skew divergence

The α-skew divergence is a non-symmetric measure of the difference between two probability distributions *P* and *Q*. Typically *P* represents the observations, in my case the n-gram count distribution of the most likely word, and *Q* represents a model, in my case the candidate's distribution. 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 (Kullback, 1959):

$$D(P \| Q) = \sum_v P(v) \, log \, \frac{P(v)}{Q(v)}$$

The α parameter is for avoiding the problem of zero probabilities, and in my method I use α=0.99. The value of the α-skew divergence measure is zero if the two probability distributions are identical and increases positively as the distributions become less similar.

I will use the example in Figure 3.4 to demonstrate how to calculate the contextual divergence between the most likely word *bright* and a substitute *clever*. First I calculate the n-gram frequency distributions of both words. I divide each n-gram frequency by the total frequency to get C_{ni} as shown in Figure 3.4, where *i* means the *i*th *n*-gram; e.g. C_{32} is the second tri-gram. For a word, C_{ni} should sum up to 1 (over all *n*, *i*). Then I can derive the α-skew divergence of these two distributions, which is 0.014 in my example. Similar to the NGM method, I define a score function $Score_{DVG}(w) = 1 - \frac{S_\alpha(\overrightarrow{w}, \overrightarrow{the\_most\_likely\_word})}{max_{divergence}}$, where $\overrightarrow{w}$ and $\overrightarrow{the\_most\_likely\_word}$ are the probability distributions of n-gram counts of the target substitute and the most likely word, respectively, and $max_{divergence}$ is the maximum divergence between the most likely word and another word in the synonym transitive closure chain. In this example, $max_{divergence}$ is 0.15 and the derived $Score_{DVG}(clever)$ is 0.91. The reason to calculate $\frac{S_\alpha(\overrightarrow{w}, \overrightarrow{the\_most\_likely\_word})}{max_{divergence}}$ is to spread the divergence score between 0 and 1.

	C_{21}	C_{22}	C_{31}	C_{32}	C_{33}	C_{41}	C_{42}	C_{43}	C_{51}	C_{52}	C_{53}
bright	0.081	0.892	0.002	0.024	0.0002	0	0	0	0	0	0
clever	0.130	0.843	0.006	0.020	0.0002	0.001	0	0	0	0	0
S_α*(clever,bright)* = $\sum_n \sum_i C_{ni}^{bright} \cdot log(\frac{C_{ni}^{bright}}{\alpha C_{ni}^{clever} + (1-\alpha) C_{ni}^{bright}})$ = 0.014											
Score$_{DVG}$*(clever)* = $1 - \frac{0.014}{0.15}$ = 0.91											

Figure 3.4: An example of the proposed DVG method

Note that the higher the divergence $S_\alpha(\overrightarrow{w}, \overrightarrow{\textit{the\_most\_likely\_word}})$ is, the lower the score *Score*$_{DVG}$*(w)*. Finally I combine the distributional similarity with the NGM method, referred as NGM_DVG method, by modifying the score function as follows:

$$\textit{Score}_{NGM\_DVG}(w) = \lambda \cdot \textit{Score}_{NGM}(w) + (1 - \lambda) \cdot \textit{Score}_{DVG}(w),$$

where $0 \leq \lambda \leq 1$. The value of λ decides the weight of *Score*$_{NGM}(w)$ and *Score*$_{DVG}(w)$, and is an empirical parameter.

Both NGM and NGM_DVG assign a score to a word according to the context and the most likely word in the group. In order to evaluate the performance of the proposed scoring methods, I apply my approaches to a ranking task that requires a system to rank a list of candidate words given an original word and its context. The task can test whether the proposed methods are capable of assigning higher scores to appropriate substitutes than to unacceptable ones, and thus is useful for the steganography application. The gold standard data is derived from the English Lexical Substitution task for SemEval-2007 (McCarthy and Navigli, 2007) and the the evaluation measure used is Generalised Average Precision (Kishida, 2005).

3.2 Ranking task evaluation

The ranking task gives a system a list of substitute words and the original word to be replaced in context. A system then ranks the candidate list so that ideally appropriate substitutes rank higher than words that are not acceptable to the context. In this section I first describe the gold standard data used in this evaluation and then provide the experiment results. I compare my results with two other models developed by Erk and

Sentence	Substitutes
He was *bright* and independent and proud.	intelligent(3), clever(3)
The roses have grown out of control, wild and carefree, their *bright* blooming faces turned to bathe in the early autumn sun.	colourful(2), brilliant(1), gleam(1), luminous(1)

Table 3.1: Two sentences in the SemEval-2007 lexical substitution gold standard

Padó (2010) and Dinu and Lapata (2010), both of which are designed for measuring word meaning similarity in context.

3.2.1 Data

For this evaluation, I use the SemEval-2007 lexical substitution dataset as the gold standard. The original purpose of the dataset is to develop systems that can automatically find feasible substitutes given a target word in context. The human annotation data comprises 2,010 sentences selected from English Internet Corpus (Sharoff, 2006), and consists of 201 target words: nouns, verbs, adjectives and adverbs each with ten sentences containing that word. The five annotators were asked to provide up to three substitutes for a target word in the context of a sentence, and were permitted to consult a dictionary or thesaurus of their choosing. After filtering out annotation sentences where the target word is part of a proper name and for which annotators could not think of a good substitute, the data was separated into 298 trial sentences and 1,696 test sentences. Table 3.1 illustrates two examples from the gold standard, both featuring the target word *bright*. The right column lists appropriate substitutes of *bright* in each context, and the numbers in parentheses indicate the number of annotators who provided that substitute.

To allow comparison with previous results reported on the substitution ranking task, following Erk and Padó (2010) and Dinu and Lapata (2010), I pool together the positive substitutes for each target word, considering all contexts, and rank the substitutes using my scoring methods. For instance, assume in the gold standard there are only two sentences containing the target word *bright* as shown in Table 3.1. I merge all the substitutes of *bright* given by the annotators and derive a large candidate pool $\{$*intelligent*, *clever*, *colourful*, *brilliant*, *gleam*, *luminous*$\}$. I expect *intelligent* and *clever* to be

ranked at the top of the list for the first sentence, with *colourful*, *brilliant*, *gleam* and *luminous* ranked at the top for the second sentence.

3.2.2 Experiments and results

In the SemEval-2007 lexical substitution task participants were asked to discover possible replacements of a target word so the evaluation metrics provided are designed to give credit for each correct guess and do not take the ordering of the guesses into account. In contrast, in the ranking task a system is already given a fixed pool of substitutes and is asked to recover the order of the list. Therefore, I use the Generalised Average Precision (GAP) to evaluate the ranked lists rather than the metrics provided in the SemEval-2007 lexical substitution task. GAP rewards correctly ranked items with respect to their gold standard weights while the traditional average precision is only sensitive to the relative positions of correctly and incorrectly ranked items. Let $G = \langle g_1, g_2, ..., g_m \rangle$ be the list of gold substitutions with weights $\langle y_1, y_2, ..., y_m \rangle$ for a target word in context. In my task, the weight is the frequency of a substitute in the gold standard. Let $S = \langle s_1, s_2, ..., s_n \rangle$ be the system ranked substitute list and $\langle x_1, x_2, ..., x_n \rangle$ be the weights associated with them, where $m \leq n$ and $x_i = 0$ if s_i is not in the gold list and $G \subseteq S$. Then

$$GAP(S, G) = \frac{1}{\sum_{j=1}^{m} I(y_j)\bar{y_j}} \sum_{i=1}^{n} I(x_i)\bar{x_i} \quad \text{and} \quad \bar{x_i} = \frac{1}{i} \sum_{k=1}^{i} x_k$$

where $I(x_i) = 1$ if x_i is larger than zero, zero otherwise; $\bar{x_i}$ is the average gold weight of the first i system ranked items; $\bar{y_i}$ is defined analogously.

After experimenting on the trial data, I decided a λ value of 0.6 for the NGM_DVG method. I then applied the proposed NGM and NGM_DVG methods to rank pooled substitutes for each sentence in the test data. Table 3.2 summarises the performances of my approaches, where mean GAP values are reported on the whole test data as well as different POS. One can see that the NGM_DVG performs better than the NGM system on the ranking task and achieved a mean GAP of 50.8% on the whole test data. I then compare my results with that achieved by Erk and Padó (2010) and Dinu and Lapata (2010). Erk and Padó (2010) developed an exemplar-based model for capturing word meaning in context, where the meaning of a word in context is represented by a set of

System	test set (%)	noun (%)	verb (%)	adj (%)	adv (%)
NGM	49.7	48.5	44.3	53.2	64.7
NGM_DVG	50.8	50.9	44.6	53.7	66.2
Dinu and Lapata (2010)	42.9	n/a	n/a	n/a	n/a
Erk and Padó (2010)	38.6	n/a	n/a	n/a	n/a

Table 3.2: GAP values of the ranking task evaluation

exemplar sentences most similar to it. Dinu and Lapata (2010) proposed a vector-space model which models the meaning of a word as a probability distribution over a set of latent senses. The best mean GAP values reported by Erk and Padó (2010) and Dinu and Lapata (2010) are 38.6% and 42.9% on the test data, respectively. According to Table 3.2, it seems to be easier to rank adjective and adverb lists compared to that of nouns and verbs.

Although the ranking task evaluation gives some indication of how reliable the proposed scoring methods are, for the steganography application I require a system that can correctly distinguish acceptable substitutes from unacceptable ones. Thus, in the next section, I conduct a classification task evaluation and observe the performances of the NGM and NGM_DVG methods with different score threshold values.

3.3 Classification task evaluation

The classification task is more related to the steganography application. The task requires a system to determine acceptable substitutes from a group of candidates given the word to be replaced and its context. Those passed substitutes can then carry different codes and be used as stego words. Similar to the previous section, I first describe the data and then explain the experimental setup and the evaluation results.

3.3.1 Data

I use the sentences in the gold standard of the SemEval-2007 Lexical Substitution Task as the cover text in my experiments so that the substitutes provided by the annotators can be the positive data. Since I only take into consideration the single word

	noun	verb	adj	adv
number of target words	59	54	57	35
number of sentences	570	527	558	349
number of positives	2,343	2,371	2,708	1,269
number of negatives	1,914	1,715	1,868	884

Table 3.3: Statistics of experimental data

substitutions, multi-word substitutes are removed from the positive data. Moreover, I use WordNet as the source of providing candidate substitutes in my stegosystem, so if a human-provided substitute does not appear in any synsets of its target word in WordNet, there is no chance for my stegosystem to replace the target word with the substitute; therefore, the substitute can be eliminated. Table 3.3 presents the statistics of the positive data for my experiments.

Apart from positive data, I also need some negative data to test whether my methods have the ability to filter out bad substitutions. I extract the negative data for my experiments by first matching positive substitutes of a target word to all the synsets that contain the target word in WordNet. The synset that includes the most positive substitutes is used to represent the meaning of the target word. If there is more than one synset containing the highest number of positives, all of those synsets are taken into consideration. I then randomly select up to six single-word synonyms other than positive substitutes from the chosen synset(s) as negative instances of the target word. Figure 3.5 shows an example of automatically collected negative data from WordNet given a target word and its positive substitutes. The synset {*remainder*, *balance*, *residual*, *residue*, *residuum*, *rest*} is selected for negative data collection since it contains one of the positives while the other synsets do not. I assume the selected synset represents the meaning of the original word, and those synonyms in the synset which are not annotated as positives must have a certain degree of mismatch to the context. Therefore, from this example, *balance*, *residue*, *residuum* and *rest* are extracted as negatives to test whether my checking methods can pick out the substitutions not used by a small number of human annotators from a set of words sharing similar or the same meaning.

In order to examine whether the automatically collected instances are true negatives and hence form a useful test set, a sample of automatically generated negatives was se-

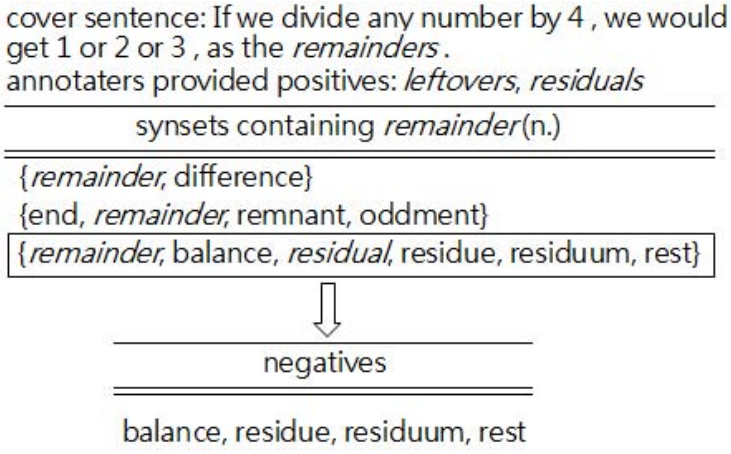


Figure 3.5: An example of automatically collecting negative data

	noun	verb	adj	adv
number of true negatives	234	201	228	98
number of false negatives	9	20	28	16

Table 3.4: Annotation results for negative data

lected for human evaluation. For each POS, one sentence of each different target word was selected, which results in roughly 13% of the collected negative data, and every negative substitute of the selected sentences was judged by my supervisor, a native English speaker. As can be seen from the annotation results shown in Table 3.4, most of the instances are true negatives, and only a few cases are incorrectly chosen as false negatives. Since the main purpose of the data set is to test whether the proposed checking methods can guard against inappropriate lexical substitutions and be integrated in the stegosystem, it is reasonable to have a few false negatives in my experimental data. Also, it is less harmful to rule out a permissible substitution than including an inappropriate replacement for a stegosystem in terms of the security. Table 3.3 gives the statistics of the automatically collected negative data for my experiments.

	System classification = T	System classification = F
Gold standard label = T	T_p	F_n
Gold standard label = F	F_p	T_n

Table 3.5: Definition of T_p,T_n,F_p and F_n

3.3.2 Experiments and results

I evaluate the classification performances of the NGM system and the NGM_DVG system in terms of accuracy, precision and recall. Accuracy is the percentage of correct classification decisions over all acceptable and unacceptable substitutes; precision is the percentage of system accepted substitutes being human-provided; recall is the percentage of human-provided substitutes being accepted by the system. Accuracy is less important for the steganography application, and the reasons for using precision and recall were explained in Section 1.4: a higher precision value implies a better security level, and a larger recall value means a greater payload capacity. Accuracy, precision and recall are defined as follows:

$$\textit{Accuracy} = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \quad \textit{Precision} = \frac{T_p}{T_p + F_p} \quad \textit{Recall} = \frac{T_p}{T_p + F_n},$$

where the definitions of true positive (T_p), true negative (T_n), false positive (F_p) and false negative (F_n) are illustrated in Table 3.5. It is worth noting that although there will be a decrease in recall if more false negatives are obtained from a system, there will not be a negative effect on the value of precision. That is, from a security perspective, it would be harmless if a system rejects an acceptable substitute since this will not drop the security level, but it will lower the payload capacity.

Both the NGM system and the NGM_DVG system require a threshold to decide whether a word is acceptable in context. In order to derive sensible threshold values for each POS, 5-fold cross validation was used for the experiments. For each fold, 80% of the data is used to find the threshold value which maximises the accuracy, and that threshold is then applied to the remaining 20% to get the final result.

I first test whether the proposed methods would benefit from using only longer n-grams. I compare the performance of different combinations of n-gram counts, which are frequency counts of bi- to five-grams, tri- to five-grams, four- to five-grams and

	NGM				NGM_DVG			
POS	Acc (%)	Pre (%)	Rec (%)	Threshold	Acc (%)	Pre (%)	Rec (%)	Threshold
noun	70.2	70.0	80.2	0.58	68.1	66.5	67.3	0.70
verb	68.1	69.7	79.5	0.56	64.8	65.7	66.7	0.70
adj	72.5	72.7	85.7	0.48	70.2	68.8	77.7	0.63
adv	73.7	76.4	80.1	0.54	68.0	66.4	75.9	0.63

Table 3.6: Performance of the NGM and NGM_DVG systems on the classification task

five-grams only. The results show that for both methods the accuracy, precision and recall values drop when using fewer n-grams. In other words, among the four combinations, the one including bi-gram to five-gram frequency counts performs the best across different POS and, therefore, is adopted in the NGM system and the NGM_DVG system. Table 3.6 gives the results for the two checking methods and the average threshold values over the five folds. In contrast to the results of the ranking task evaluation, this time the NGM system slightly outperforms the NGM_DVG system. Since imperceptibility is an important issue for steganography, I would prefer a system with a higher precision value. Thus I adopt the NGM method as the linguistic transformation checker in my lexical substitution-based stegosystem.

In order to have a rough idea why the NGM_DVG system is not effective on the classification task, I examine some of the false positives of the NGM_DVG system which have been correctly classified as unacceptable by the NGM system. The qualitative evaluation suggests that the major reason for the NGM_DVG method getting such false positives is because the most unlikely word of a substitution group has a very different n-gram count distribution to the most likely word, which leads to a larger $max_{divergence}$ for that substitution group, and therefore, an unacceptable substitute w in that group may have a higher $Score_{DVG}(w)$. This situation usually happens when most of the n-grams of the most unlikely word cannot be found in the Google n-gram corpus. For example, in the experiment data {*luminous, clear, light, brilliant, burnished, shining, vivid, hopeful, lustrous*} is the substitution group for the target word *bright* in the sentence "The actual field is not much different than that of a 40mm, only it is smaller and quite a bit noticeably brighter, which is probably the main benefit" where *luminous*, *clear* and *light* are acceptable to the context. For this substitution group the $max_{divergence}$ is derived from *bright* and *lustrous* since *lustrous* only has a non-zero

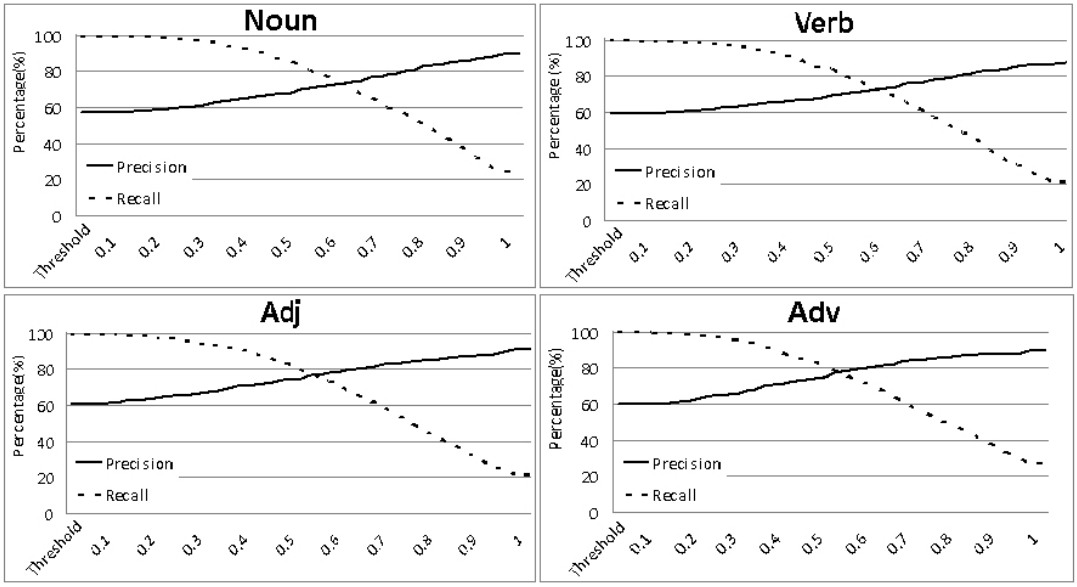


Figure 3.6: The performance of the NGM method under various thresholds

n-gram count for the bi-gram "lustrous ," while most of the contextual n-grams of *bright* can be found in the Google n-gram corpus. This $max_{divergence}$ leads to three false positives *shining*, *vivid* and *hopeful* determined by the NGM_DVG method.

In addition, I am interested in the effect of the threshold value on the performance of the NGM method. Figure 3.6 shows the precision and recall values with respect to different thresholds for each POS. From the charts one can clearly see the trade-off between precision and recall. Although a higher precision can be achieved by using a higher threshold value — for example noun substitutions reach almost 90% precision with threshold equal to 0.9 — the large drop in recall means many applicable substitutes are being eliminated. In other words, the trade-off between precision and recall implies the trade-off between imperceptibility and payload capacity for linguistic steganography. Therefore, the practical threshold setting would depend on how steganography users want to trade off imperceptibility for payload.

So far I have presented the performance of my checking methods using two different automatic evaluations, the ranking task and the classification task. From the ranking task evaluation one can see that the n-gram distributional similarity does have the ability to further eliminate some bad substitutes after applying the basic n-gram method.

However, when facing the classification task, which is more related to the steganography application, I find that the checking method simply based on counts from the Google n-gram corpus is hard to beat. In addition, from the results of different order n-gram combinations I can conclude that the more information I include in the checking method, namely using all counts from bi- to five-grams, the better the performance. This is similar to the conclusion given by Bergsma et al. (2009): it is not only important to use the largest possible corpus, but to get maximum information from the corpus. Apart from the automatic evaluations, in the next section I will describe a more direct evaluation of the imperceptibility for the steganography application by asking human judges to evaluate the naturalness of sentences.

3.4 Human evaluation

I want to test how reliable the proposed NGM method is if it is used in a lexical substitution-based stegosystem to guard against inappropriate substitutions. Thus, I conducted a human evaluation of sentence naturalness. In the following sections, I explain the evaluation data first and then describe the evaluation setup and results.

3.4.1 Data

I collected a total of 60 sentences from Robert Peston's BBC blog.[2] For each noun, verb, adjective and adverb in a sentence, I first group the target word's synset(s) in WordNet and apply the NGM method with a score threshold equal to 0.95 to eliminate bad substitutes. If more than one substitute passes the check, the one with the lowest score is used to replace the original word. The reason for choosing the word with the lowest score is because this makes the test more challenging. This process is applied to a sentence where possible and results in around two changes being made per sentence.

I also generated another version of a sentence changed by random choice of a target word and random choice of a substitute from a target word's synset(s) (in order to provide a baseline comparison). The number of changes made to a sentence using this

[2] http://www.bbc.co.uk/news/correspondents/robertpeston (last verified in June 2013).

Version	Sentence
COVER	Apart from anything else, **big** companies have the size and muscle to derive gains by **forcing** their suppliers to cut prices (as **shown** by the furore highlighted in yesterday's Telegraph over Serco's **demand** - now withdrawn - for a 2.5% rebate from its suppliers); smaller businesses lower down the food chain simply don't have that opportunity.
SYSTEM	Apart from anything else, **large** companies have the size and muscle to derive gains by **pushing** their suppliers to cut prices (as **evidenced** by the furore highlighted in yesterday's Telegraph over Serco's **need** - now withdrawn - for a 2.5% rebate from its suppliers); smaller businesses lower down the food chain simply don't have that opportunity.
RANDOM	Apart from anything else, **self-aggrandising** companies have the size and muscle to derive gains by forcing their suppliers to **foreshorten** prices (as shown by the furore highlighted in yesterday's Telegraph over Serco's demand - now withdrawn - for a 2.5% rebate from its suppliers); smaller businesses lower down the food chain simply don't **birth** that **chance**.

Table 3.7: Different versions of a cover sentence

random method is the same as that in the version generated by the NGM method. In this way, it is fair to compare the qualities of the two modified versions since both of them receive the same number of substitutions. Table 3.7 shows lexical substituted sentences generated by my method and by the random method. One can see that my system replaces four words (in bold) in the original sentence so the same number of words (in bold) are randomly selected when applying the random method. Note that the random method just happens to pick the word *big* in the original sentence which is also replaced by my system. I refer to an original sentence as COVER, a version generated by my method as SYSTEM and a version modified by the random method as RANDOM.

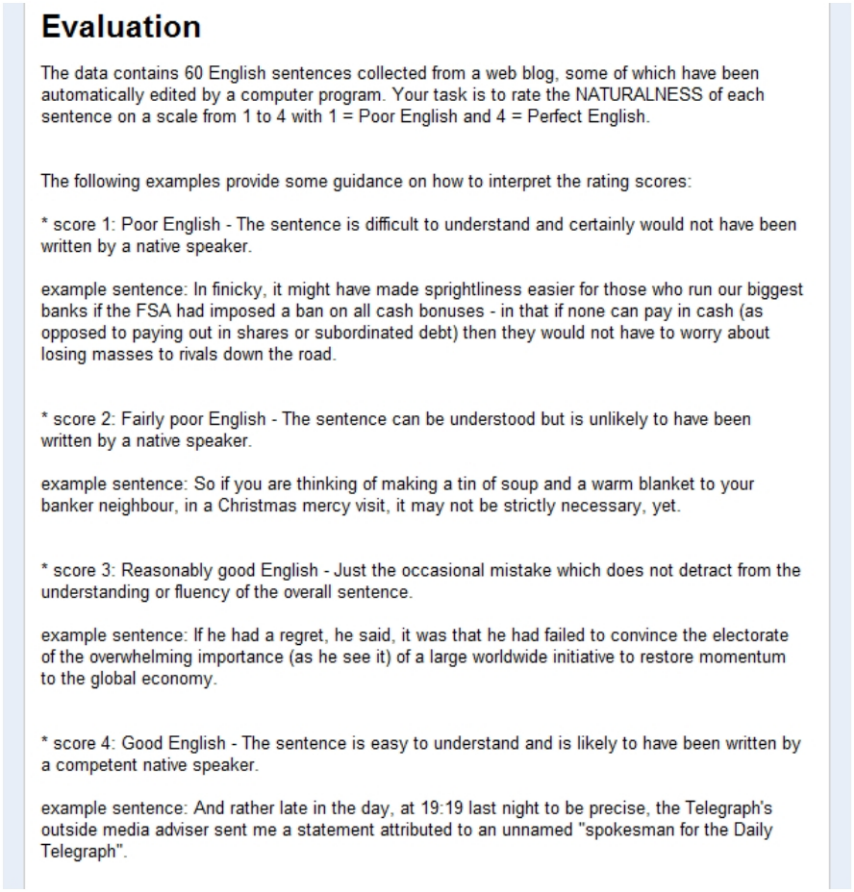
	$s_1, s_2, \ldots, s_{20}$	$s_{21}, s_{22}, \ldots, s_{40}$	$s_{41}, s_{42}, \ldots, s_{60}$
Group 1	COVER	SYSTEM	RANDOM
Group 2	RANDOM	COVER	SYSTEM
Group 3	SYSTEM	RANDOM	COVER

Table 3.8: Latin square design with three groups of judges

3.4.2 Evaluation setup and results

The experimental setup follows a Latin square design (Kirk, 2012) with three groups of 10 native English speakers as shown in Table 3.8. In this table, each row represents a set of annotation sentences for a group of judges, and one can see that each sentence is presented in three different conditions: COVER, SYSTEM and RANDOM, as shown in a column. Subjects in the same group receive the 60 sentences under the same set of conditions, and each subject sees each sentence only once in one of the three conditions. The annotation process is web-based. At the beginning of the annotation task, I describe the aim of the annotation as shown in Figure 3.7. Subjects are asked to rate the naturalness of each sentence on a scale from 1 to 4 with score 1 meaning *Poor English* and score 4 meaning *Perfect English*. Each judgement score is explained followed by an example sentence. Figure 3.8 shows a screen capture of an annotation example presented to a subject.

The annotation results show that my judges gave an average score 3.67 out of 4 for the original sentences; 3.33 for the sentences checked by the NGM system; and 2.82 for the randomly changed sentences. I measure the significance level of my annotation results using the Wilcoxon signed-rank test (Wilcoxon, 1945). The test statistic shows that the differences between the three versions (original, system changed and randomly changed) are highly significant ($p < 0.01$). The payload capacity for this level of imperceptibility is around 2 information carriers per sentence and each information carrier guarantees to encode at least 1 bit. These results show that my stegosystem achieves better payload capacity than existing lexical substitution-based stegosystems which have achieved 0.67 bits per sentence (Topkara et al., 2005, 2006c). In addition, in terms of security, the results suggest that with the proposed checking method, the quality of stego sentences is improved compared to the random substitution baseline, and a certain security level is achieved so the changes are not likely to be spotted.



Evaluation

The data contains 60 English sentences collected from a web blog, some of which have been automatically edited by a computer program. Your task is to rate the NATURALNESS of each sentence on a scale from 1 to 4 with 1 = Poor English and 4 = Perfect English.

The following examples provide some guidance on how to interpret the rating scores:

* score 1: Poor English - The sentence is difficult to understand and certainly would not have been written by a native speaker.

example sentence: In finicky, it might have made sprightliness easier for those who run our biggest banks if the FSA had imposed a ban on all cash bonuses - in that if none can pay in cash (as opposed to paying out in shares or subordinated debt) then they would not have to worry about losing masses to rivals down the road.

* score 2: Fairly poor English - The sentence can be understood but is unlikely to have been written by a native speaker.

example sentence: So if you are thinking of making a tin of soup and a warm blanket to your banker neighbour, in a Christmas mercy visit, it may not be strictly necessary, yet.

* score 3: Reasonably good English - Just the occasional mistake which does not detract from the understanding or fluency of the overall sentence.

example sentence: If he had a regret, he said, it was that he had failed to convince the electorate of the overwhelming importance (as he see it) of a large worldwide initiative to restore momentum to the global economy.

* score 4: Good English - The sentence is easy to understand and is likely to have been written by a competent native speaker.

example sentence: And rather late in the day, at 19:19 last night to be precise, the Telegraph's outside media adviser sent me a statement attributed to an unnamed "spokesman for the Daily Telegraph".

Figure 3.7: The introduction and guidelines for the lexical substitution annotation

As described at the beginning of this chapter, the proposed stegosystem extends the original synset by adding words in a synonym transitive closure chain while retaining the synonymous relationships between words using a synonym graph representation. The NGM checker effectively controls the size of a synonym graph according to the context. After constructing the graph, namely obtaining all the good alternatives for a cover word, the data encoding module needs to assign codes to every word in the graph. In the next section, I explain the encoding method used in my stegosystem which is based on vertex colouring.

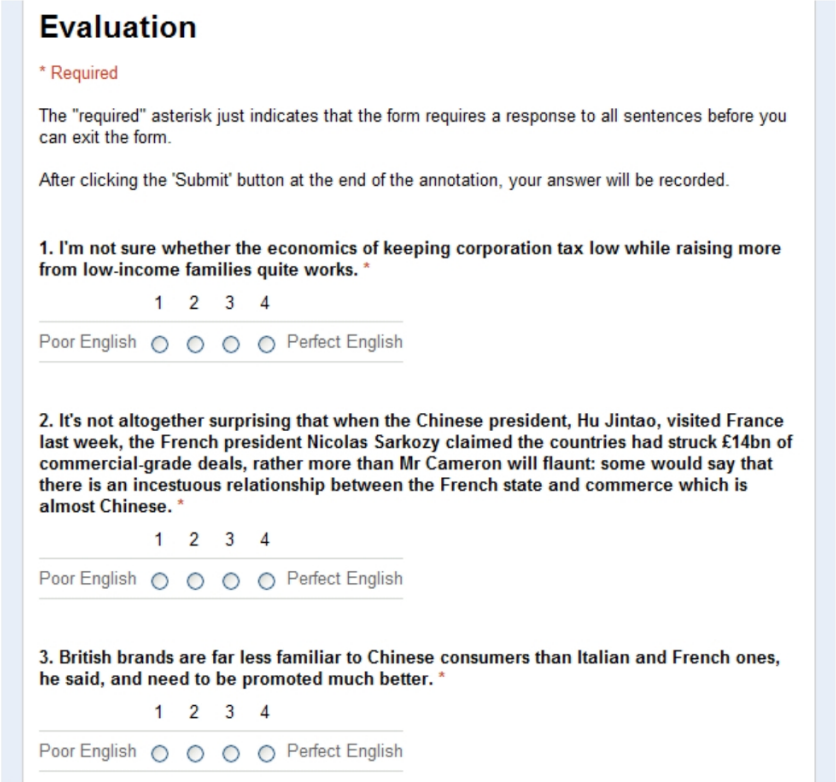


Figure 3.8: A screen capture of the lexical substitution annotation

3.5 Vertex colouring coding method

A vertex colouring (Gould, 1988) is a labelling of a graph's vertices with colours subject to the condition that no two adjacent vertices share the same colour. The smallest number of colours required to colour a graph G is called its chromatic number $\chi(G)$, and a graph G having chromatic number $\chi(G) = k$ is called a k-chromatic graph. The aim of the proposed coding method is to convert an input synonym graph into a coloured k-chromatic graph so that each node, namely word, is encoded by the code that represents the colour. Figure 3.9 shows the coloured 4-chromatic synonym graph of the two joint synsets from Figure 3.1, and the four colours are represented by four block codes *00*, *01*, *10* and *11*. Now, the problematic word *marry* receives a unique codeword no matter which synset is considered, which means the secret recovery will not encounter an ambiguity since the receiver can apply the same coding method to derive identical codewords used by the sender.

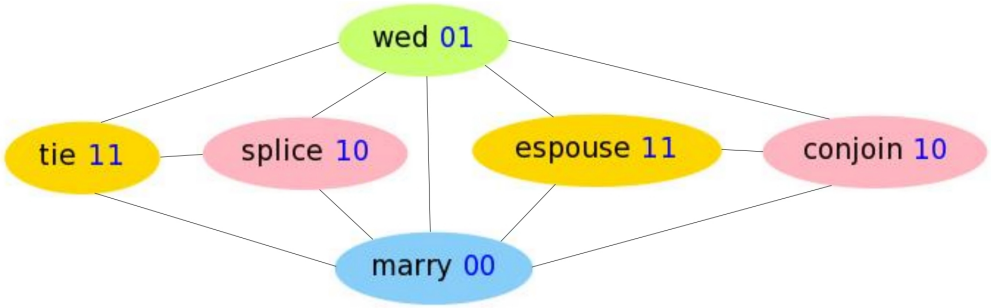
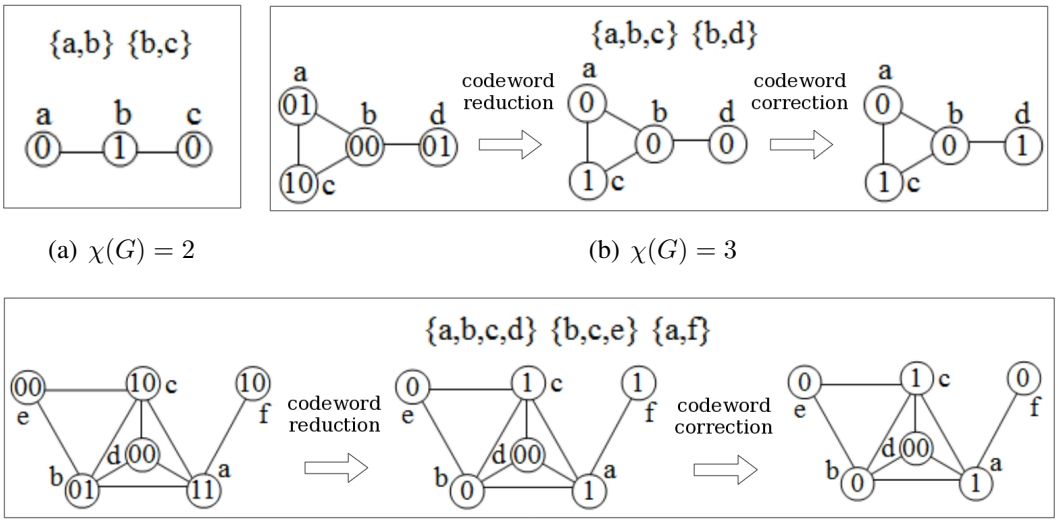


Figure 3.9: Encoding two joint synsets using the vertex colouring coding method



(a) $\chi(G) = 2$

(b) $\chi(G) = 3$

(c) $\chi(G) = 4$

Figure 3.10: Examples of coded 2,3,4-chromatic synonym graphs

99.6% of synsets in WordNet have size less than 8, which means most of the synsets cannot exhaust more than a 2-bit coding space (i.e. one can only encode at most 2 bits using a typical synset). Therefore, I restrict the chromatic number of a synonym graph G to $1 < \chi(G) \leq 4$, which implies the maximum size of a synset is 4. When $\chi(G) = 2$, each vertex is assigned a single-bit codeword, either *0* or *1*, as shown in Figure 3.10(a). When $\chi(G) = 3$, the overlapping set's size is either 2 or 3, which cannot exhaust the 2-bit coding space although codewords *00*, *01* and *10* are initially assigned to each vertex. Therefore, only the most significant bits are used to represent the synonyms, which I call *codeword reduction*. After codeword reduction, if a vertex

has the same codeword, say *0*, as all of its neighbours, the vertex's codeword must be changed to *1* so that the vertex would be able to accommodate either secret bit *0* or *1*, which I call *codeword correction*. Figure 3.10(b) shows an example of the process of codeword reduction and codeword correction for $\chi(G) = 3$. For the case of $\chi(G) = 4$, codeword reduction is applied to those vertices that have no access to all the codewords *00*, *01*, *10* and *11*. For example, vertices a, b, c, e and f in Figure 3.10(c) meet the requirement of needing codeword reduction. The codeword correction process is then further applied to vertex f to rectify its accessibility.

Figure 3.11 describes a greedy algorithm for constructing a coded synonym graph using at most 4 colours, given n substitutes w_1, w_2,..., w_n in an input synonym graph. Let us define a function $E(w_i, w_j)$ which returns an edge between w_i and w_j if w_i and w_j are in the same synset; otherwise returns false. Another function $C(w_i)$ returns the colour of the word w_i. The procedure loops through all the input substitutes. For each iteration, the procedure first finds available colours for the substitute word w_i. If there is no colour available, namely all the four colours have already been given to w_i's neighbours, w_i is randomly assigned one of the four colours; otherwise, w_i is assigned one of the available colours. After adding w_i to the graph G, the procedure checks whether adding an edge of w_i to graph G would violate the vertex colouring. After constructing the coloured graph, codeword reduction and codeword correction as previously described are applied to revise improper codewords.

3.6 Proposed lexical stegosystem

Figure 3.12 illustrates the framework of my lexical substitution-based stegosystem. Note that I have preprocessed WordNet by excluding multi-word synonyms and single-entry synsets. Table 3.9 shows the statistics of synsets used in my stegosystem. A possible information carrier is first found in the cover sentence. I define a possible information carrier as a word in the cover sentence that belongs to at least one synset in my processed WordNet. Starting from the cover word's synset, all words in the synonym transitive closure chain are examined by the NGM method. A synonym graph(s) is then built based on the remaining words. Next, I assign codes to each

INPUT: words w_1, w_2,..., w_n in a synonym graph G and an empty graph G_{coded}
OUTPUT: a coded synonym graph G_{coded} using at most four colours

FOR every word w_i in G, initialize four colours as available for w_i
    FOR every w_j in graph G_{coded}
        IF $E(w_i, w_j)$ **THEN**
            set $C(w_j)$ as unavailable
        END IF
    END FOR
    IF there is a colour available **THEN**
        A randomiser assigns one of the available colours to w_i
    ELSE
        A randomiser assigns one of the four colours to w_i
    END IF
    ADD w_i to graph G_{coded}
    FOR every w_j in graph G_{coded}
        IF $E(w_i, w_j)$ and $C(w_i)$ is not equal to $C(w_j)$ **THEN**
            ADD edge $E(w_i, w_j)$ to G_{coded}
        END IF
    END FOR
END FOR
codeword reduction
codeword correction
OUTPUT graph G_{coded}

Figure 3.11: Constructing a coloured synonym graph

word in the synonym graph(s). During the data encoding procedure, if words in the synonym graph all belong to the same synset, the block coding method is used to encode the words; otherwise the vertex colouring coding is applied to the synonym graph. Finally, according to the secret bitstring, the system selects a substitute that is synonymous with the cover word and has as its codeword the longest potential match with the secret bitstring.

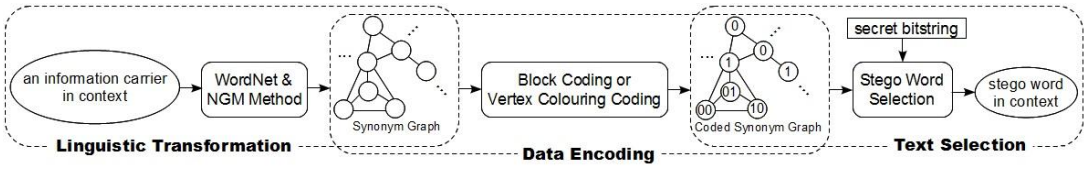


Figure 3.12: Framework of the proposed lexical substitution-based stegosystem

	noun	verb	adjective	adverb
number of synsets	16,079	4,529	6,655	964
number of words	30,933	6,495	14,151	2,025
average synset size	2.56	2.79	2.72	2.51
max synset size	25	16	21	8

Table 3.9: Statistics of synsets used in the stegosystem

Repeating a comment made earlier, I use the transitive closure chain of WordNet containing the target word as a simple method to ensure that both sender and receiver encode the same graph. It is important to note, however, that the sender only considers the synonyms of the target word as potential substitutes; the transitive closure chain is only used to consistently assign the codes.

For the decoding process, the receiver does not need the original text for extracting secret data. An information carrier can be found in the stegotext by referring to WordNet in which related synonyms are extracted. Those words in the related sets undergo the NGM checking method, and the words passing the check form a synonym graph(s). The synonym graph(s) are encoded by either block coding or the vertex colouring coding scheme depending on whether the remaining words are in the same synset. Finally, the secret bitstring is implicit in the codeword of the information carrier and therefore can be extracted.

I demonstrate how to embed secret bit 1 in the sentence "it is a *shame* that I could not reach the next stage." A possible information carrier *shame* is first found in the sentence. Table 3.10 lists the synsets in the synonym transitive closure chain extracted from WordNet. The score of each word calculated by the NGM method is given in parentheses. For the purpose of demonstrating the use of vertex colouring coding, I select a low threshold score of 0.27. The output of the synonym graph is shown

cover sentence: It is a *shame* that we could not reach the next stage.
{pity (0.97), shame (1.0)}
{shame (1.0), disgrace (0.84), ignominy (0.24)}
{commiseration (0.28), pity (0.97), ruth (0.13), pathos (0.31)}
{compassion (0.49), pity (0.97)}
{condolence (0.27), commiseration (0.28)}
{compassion (0.49), compassionateness (0)}
{pathos (0.31), poignancy (0.31)}
{poignance (0.12), poignancy (0.31)}

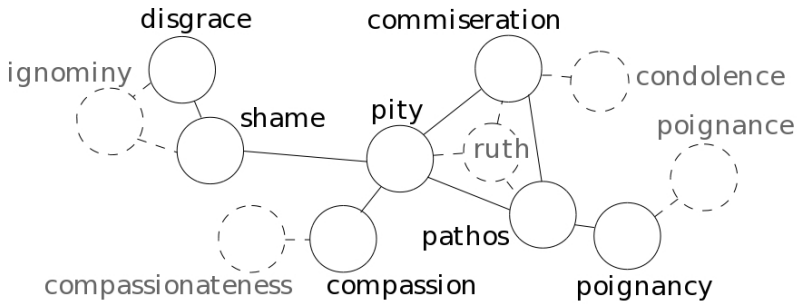
Table 3.10: Synsets of *shame* in the synonym transitive closure chain with substitution scores

in Figure 3.13(a). Since the remaining words do not belong to the same synset, the vertex colouring coding method is then used to encode the words. Figure 3.13(b) shows the coded synonym graph in which each vertex is assigned one of the four colours; Figure 3.13(c) is the graph after applying codeword reduction. Although both *disgrace* and *pity* are encoded by 1, *pity* is chosen to replace the cover word since it has a higher score. Finally, the stegotext is generated, "it is a *pity* that we could not reach the next stage." As explained previously, even if a cover word does not pass the NGM check, the proposed stegosystem can still use its synonyms to embed secret bits. For example, assume the cover sentence is "it is a *ignominy* that we could not reach the next stage". The same coded synonym graph as Figure 3.13(c) will be constructed since both the context and the synonym transitive closure chain are the same as that in the original example. This time, the replacement of *shame* represents secret bit 0, and the replacement of *disgrace* represents secret bit 1. In other words, a change must be made in order to embed a secret bit in this case.

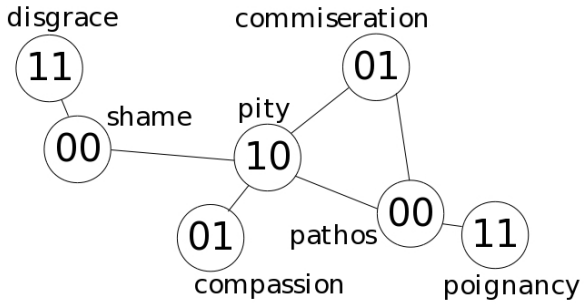
In Section 1.3 I explained Kerckhoffs's principle for designing cryptosystems (Kerckhoffs, 1883) where the security of a cryptosystem only depends on the secrecy of the key and any private randomizer shared between the sender and receiver. In the proposed steganography scheme, the secret key is the score threshold in the NGM method, and the private randomiser is the one that assigns colours in the proposed vertex colouring coding methods. The score threshold decides the size of a synonym graph, while

the randomiser controls the encoding of a synonym graph. To extract the secret message, the enemy needs to generate the same coded synonym graphs as constructed by the sender. Therefore, it is difficult to recover the secret bits without knowing the score threshold and the colour randomiser.

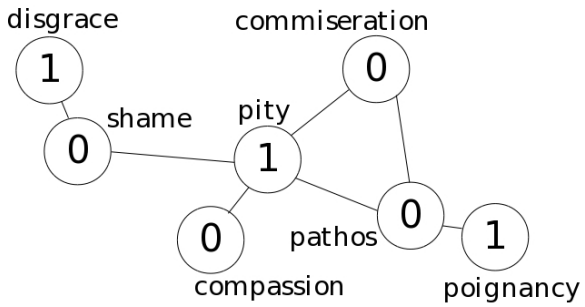
One of the contributions of this work is to develop a novel lexical substitution-based stegosystem using vertex colouring coding which improves the data embedding capacity compared to existing systems. The vertex colouring coding method represents synonym substitution as a synonym graph so the relations between words can be clearly observed. In addition, the NGM method, an automatic system for checking synonym acceptability in context, is integrated in my stegosystem to ensure information security. The proposed stegosystem was automatically evaluated by the gold standard from the SemEval2007 lexical substitution task as well as a human evaluation. From the evaluation results I may conclude that my substitution-based stegosystem has achieved a reasonable level of security while reaching the payload capacity of around 2 bits per sentence. In the next chapter, I will explain my second stegosystem which exploits unnecessary adjectives in text.



(a) The synonym graph of the synsets in Table 3.10



(b) Coded synonym graph using four colours



(c) Coded synonym graph after codeword reduction

Figure 3.13: Synonym graphs generated by the proposed stegosystem

Chapter 4

Adjective deletion

In this chapter I introduce my second linguistic stegosystem based on adjective deletion.[1] As explained previously, in order to make steganographic encoding difficult to detect, the changes to a cover medium due to the embedding of secret information must be imperceptible. From an information theoretical point of view, this means that the changes must be made to the redundant parts of a cover medium which are not required for the cover signal. In linguistic steganography, the cover medium is text, and the cover signal is the meaning conveyed by the text. Therefore, my intuition of this work is to develop a system that can automatically determine words in text that are unnecessary for forming a natural meaningful sentence.

I have identified adjectives as a potentially large source of redundancy in text, in the sense that adjectives can often be removed without significantly affecting the naturalness of the resulting text. For example, "he spent only his *own* money" and "he spent only his money" are both natural and meaningful sentences. 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*. Therefore, I explore the identification of redundant adjectives in context for the applications of linguistic steganography. In addition, I apply the adjective deletion technique to another cryptographic method called secret sharing (Blakley, 1979; Shamir, 1979) which aims at distributing a secret among a group of people so that the secret can only be recovered when a sufficient number of secret pieces are combined together. I propose a novel linguistic secret shar-

[1] The content in this chapter was published as Chang and Clark (2012a).

Sentence	Grammatical relations
He is affluent rich.	(ncmod _ $rich_{JJ}$ $affluent_{JJ}$)
	(xcomp _ is_{VBZ} $rich_{JJ}$)
	(ncsubj is_{VBZ} He_{PRP} _)
This is a small beer festival.	(ncmod _ $festival_{NN}$ $beer_{NN}$)
	(ncmod _ $festival_{NN}$ $small_{JJ}$)
	(det $festival_{NN}$ a_{DT})
	(xcomp _ is_{VBZ} $festival_{NN}$)
	(ncsubj is_{VBZ} $This_{DT}$ _)
We own a nice house.	(ncmod _ $house_{NN}$ $nice_{JJ}$)
	(det $house_{NN}$ a_{DT})
	(dobj own_{VBP} $house_{NN}$)
	(ncsubj own_{VBP} We_{PRP} _)

det: the relation between a noun and its determiner.
dobj: the relation between a predicate and its direct object.
ncmod: the relation between a non-clausal modifier and its head.
ncsubj: the relation between a non-clausal subject and its verbal heads.
xcomp: the relation between a head and an unsaturated VP complement.

Table 4.1: Examples of different grammatical relations

ing method where a secret is distributed among and camouflaged in two comparable texts using the adjective deletion technique. 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.

Before explaining the proposed adjective deletion checkers, let me define the adjectives I focus on. First, I only consider an adjective if it modifies a noun according to the grammatical relation (GR) (Williams, 1984) derived from a parser, and the adjective is immediately in front of the noun in the sentence. Grammatical relations are shallow semantic representations of relationships between words in a sentence. For example, Table 4.1 shows the GRs for some example sentences derived by the Clark and Curran (2007) parser along with brief explanations of those relations. A full list of GRs and their usage can be found in Briscoe (2006). I am interested in the *ncmod* relation since it indicates the relationship between a modifier and an argument. One can see that in

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 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 4.2: Comparing supertags before and after adjective deletion

the first example sentence, the adjective *affluent* modifies *rich*; however, *rich* is also an adjective so this is not the case I am looking for. In the second example sentence, although the adjective *small* modifies the noun *festival*, they are not next to each other in the sentence and therefore do not qualify either. The third example sentence has a *ncmod* relation indicating that the adjective *nice* modifies the noun *house* and the words are next to each other so this is a case that I consider.

The second restriction is that a target adjective must pass a grammaticality check. I use the syntactic filter proposed in my earlier paper (Chang and Clark, 2010a) to prevent an ungrammatical adjective deletion. In my original work of Chang and Clark (2010a), the syntactic filter exploits the Clark and Curran (2007) CCG parser to analyse sentences before and after text paraphrasing in order to eliminate ungrammatical transformations. Combinatory Categorial Grammar (CCG) is a lexicalised grammar formalism, in which CCG lexical categories, also called supertags — typically expressing subcategorisation information — are assigned to each word in a sentence. The grammatical check works by checking if the words in the sentence outside of the phrase and paraphrase receive the same supertags before and after paraphrasing. If there is any change in supertag assignment to these words then the paraphrase is judged ungrammatical.

For my application, I observe whether there is an inconsistency in the supertags assigned to words other than the target adjective in a sentence before and after removing the target adjective. If an adjective deletion does not change the supertags of the other words, the deletion is seen as grammatical. Table 4.2 shows two adjective deletion examples where both instances of *last* meet my adjective-noun pair requirement.[2] However, only the first deletion case passes the grammaticality check since all the supertags

[2]There is a parse error in the first sentence, but it does not affect the supertag comparison.

remain the same after deleting *last*; while in the second example, both *UK* and *time*'s supertags are changed after the deletion. Therefore, the instance of *last* in the sentence "those awaiting execution spent their last days alone" passes the grammaticality check and is the target adjective I am interested in.

The syntactic filter is at the word, rather than derivation, level; however, CCG lexical categories contain a large amount of syntactic information which this method is able to exploit. This grammaticality check is only a preliminary check and does not guarantee sentence fluency. Therefore, in the following sections, I propose two methods for checking the acceptability of adjective deletions 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; therefore, the results give some indication of the security level of an adjective deletion-based stegosystem that uses my deletion checkers to certify the transformation quality. I then demonstrate that the adjective deletion technique can be combined with a coding method proposed in Chang and Clark (2010a). In addition, I explain the proposed secret sharing scheme based on adjective deletion.

4.1 Adjective deletion checkers

In this section, I propose two methods for determining deletable adjectives. A deletable adjective is defined as one where the removal of the adjective does not affect the naturalness of the resulting sentence; that is, the generated sentence must be grammatical and semantically meaningful. Note that the generated sentence does not necessarily convey the same or similar meaning as the original, since in this work I only consider the sentence-level naturalness rather than the coherence of the whole document. As for the lexical substitution, I use tools of Minnen et al. (2001) for correcting the form of an indefinite after removing a target adjective. For example, the indefinite *a* is changed to *an* after removing *little* in the sentence "UVA Radiation has a little effect on skin damage".

4.1.1 N-gram count method

My first adjective deletion checker is similar to that proposed in the previous chapter using the Google n-gram corpus to calculate a deletion score for an adjective in context. The score is based on the n-gram counts before and after a potential deletion, as demonstrated in Figure 4.1, and a score threshold is used to prevent a low-score deletion, namely an unacceptable case. For the example in Figure 4.1 I first extract contextual bi- to five-grams containing the target adjective *alternative* as well as that across the target position with *alternative* removed. In this example there are 14 before-deletion n-grams and 10 after-deletion n-grams. The Google n-gram corpus is then consulted to obtain their frequency counts. I sum up all the logarithmic counts[3] 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 different numbers of n-grams are extracted before and after the adjective removal, I divide the sum by the number of extracted n-grams and call the derived average value the $Count_{deletion}$. Finally, I use a $Score_{deletion}$ function which is equal to $\frac{Count^{After}_{deletion}}{Count^{Before}_{deletion}}$ to measure how much the $Count^{Before}_{deletion}$ changes after deleting the target adjective *alternative*. In this example the $Score_{deletion}$ 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.

Some n-grams may be more informative than others when deciding whether an adjective can be deleted. Therefore, in the next section I propose a machine learning method that combines the n-gram counts and other numerical measures to train a Support Vector Machine (SVM) classifier (Cortes and Vapnik, 1995; Hearst et al., 1998) which has been successfully applied to different classification problems with numeric data.

[3] 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)	
$Count^{Before}_{deletion} = 4.8$	$Count^{After}_{deletion} = 6.8$
$Score_{deletion} = \frac{Count^{After}_{deletion}}{Count^{Before}_{deletion}} = 1.4$	

Figure 4.1: An example of the Google n-gram count method for checking adjective deletion in context

4.1.2 SVM method

Support vector machines (SVMs) are supervised learning models capable of solving linear and non-linear classification and regression problems and have been applied to many NLP problems, such as text categorization (Joachims, 1998; Leopold and Kindermann, 2002) and syntactic/semantic dependency structure analysis (Kudo and Matsumoto, 2000; Bohnet, 2009). The basic SVM takes n training data points of the form (x_i, y_i), where x_i is a d dimensional vector (a list of d numerical features), $y_i \in -1, 1$ indicating one of the two classes to which the point x_i belongs, and $i = 1 \ldots n$. The SVM then constructs a (d-1)-dimensional hyperplane that has the largest distance to the nearest training data point of any class — the maximum-margin hyperplane — which divides the points having $y_i = 1$ on the one side and having $y_i = -1$ on the

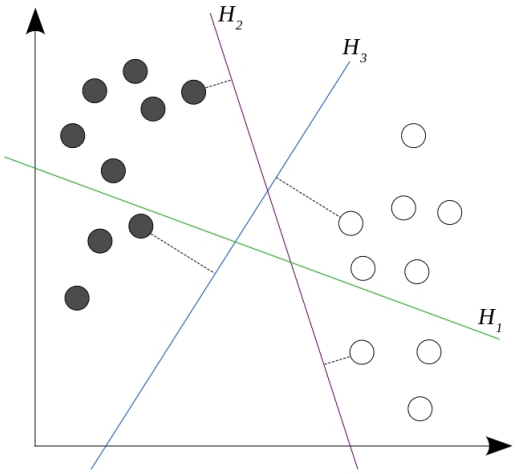


Figure 4.2: Three different hyperplanes in a 2-dimensional space

other. In other words, the goal of a basic SVM is to find a hyperplane such that the two classes are divided linearly by a margin that is as wide as possible. Figure 4.2 shows three different hyperplanes in a 2-dimensional space. In this Figure, H_1 does not separate the two classes; both H_2 and H_3 successfully divide the two classes, but H_3 has the maximum margin and therefore will be used in a trained SVM model to predict the class of a new data point.

There are cases where an SVM cannot find a suitable hyperplane to separate the training data points. In this situation, an SVM can map the data points to a higher-dimensional space using a non-linear kernel function. In this way, an appropriate linear separating hyperplane may be found in the higher-dimensional space; that is, the hyperplane is non-linear in the original data space. The commonly used kernel functions include polynomial function, Gaussian radial basis function and sigmoid function.

I use the LIBSVM (Chang and Lin, 2011) implementation of SVMs with the default Gaussian radial basis function (RBF) kernel for classifying deletable adjectives. In this SVM, there are two parameters, the misclassification penalty parameter C and the kernel parameter γ. In order to identify what C and γ values are suitable for my problem, I use the model selection (parameter search) tool grid.py provided from LIBSVM to determine the best (C, γ). The grid-search tool tries various pairs of (C, γ), the values of which are exponentially growing sequences (e.g. $C = 2^{-5}, 2^{-3}, ..., 2^{15}$; $\gamma = 2^{-15}, 2^{-13}, ..., 2^{3}$), and then selects the (C, γ) pair that achieves the best

cross-validation accuracy on the training data. Using the cross-validation procedure can prevent the problem of training data over fitting.

In addition, I exploit the probability estimate implementation in LIBSVM which is originally proposed by Platt (1999), and train the SVM model to output a posterior class probability $P(y = 1|x)$ instead of a definite prediction. Platt (1999) proposes approximating the posterior by fitting a sigmoid function to all estimated $f(x_i)$ and derives probabilities of the form:

$$P(y = 1|x) \approx P_{A,B}(f) \equiv \frac{1}{1 + exp(Af + B)}$$

where $f = f(x)$ and the best parameter setting (A, B) is determined by minimizing the negative log-likelihood of the training data:

$$\min_{A,B} \quad -\sum_{i=1}^{n}(\frac{y_i + 1}{2}log(p_i) + (1 - \frac{y_i + 1}{2})log(1 - p_i)), \text{where } p_i = P_{A,B}(f_i).$$

In this way, I can control the confidence of the deletable adjective prediction using a probability threshold. In other words, the probability threshold will affect the adjective deletion quality and is closely related to the security level of the proposed stegosystem.

As explained previously, an SVM uses a hyperplane to classify data points. In my case, a data point is an adjective deletion case. Thus, I need to represent an adjective deletion as a vector of numerical features. In the next section, I describe the different features used in my SVM method.

4.2 Features for the SVM

For representing an adjective deletion case, I exploit features including contextual n-gram counts found in the Google n-gram corpus, lexical association measures, adjective-noun modification entropies and contextual α-skew divergence. Each of these features is explained individually in the following sections.

4.2.1 N-gram counts

The first set of features consists of logarithmic contextual bi- to five-gram counts. Before the deletion, there are 14 contextual windows; after the deletion, there are 10

contextual windows as shown in Figure 4.1. If a target adjective has less than four context words on either sides, there will be contextual windows that span beyond the current sentence. In this situation, the counts of those unavailable windows are set to zero. For each contextual window I 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 $Score_{deletion}$ function described in Section 4.1.1. 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 $Score_{deletion}$ function. For instance, if I only use bi-gram counts to calculate the deletion score of the example in Figure 4.1, $Count^{n=2}_{Before}$ is 12.7 and $Count^{n=2}_{After}$ is 15.2 and thus, $Score^{n=2}_{deletion}$ is 1.2. Again, each score is provided with an additional boolean feature to indicate whether the $Count^{Before}_{deletion}$ is equal to zero. There are a total of 58 features contributed from the n-gram counts.

4.2.2 Lexical association measures

In addition to n-gram features, I 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 and is defined as:

$$I(x, y) = log\frac{p(x, y)}{p(x)p(y)}$$

In my case, x is an adjective and y is the noun modified by the adjective. In order to calculate PMI, I 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.

I collect adjective-noun pairs and their frequency counts from grammatical relations (GRs). The GRs I use are derived by parsing a Wikipedia dump (dated October 2007) with Clark and Curran (2007)'s CCG parser. I first consider GRs having the pattern (ncmod _ *noun adjective*) and extract the (*adjective*, *noun*) pair. Next I 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 the) (xcomp _ be red) and (ncsubj be car _), and since *car* and *red* match the two patterns, (*red*, *car*) is seen

as an eligible pair for my database. A total of 63,896,006 adjective-noun pairs are extracted from the parsed Wikipedia corpus which includes 832,320 noun types and 792,914 adjective types.

One problem with PMI is that it is sensitive to data sparseness. In particular, the level of association between words can be greatly over-estimated with PMI when the words are rare. Therefore, I also use the log likelihood ratio (LLR), an alternative to PMI, which is reported to handle rare events better (Dunning, 1993; Manning and Schütze, 1999). 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 research, I use PMI and LLR as numerical features in the SVM.

4.2.3 Noun and adjective entropy

Entropy (Manning and Schütze, 1999) is a measure of surprise or unpredictability. The entropy H of a discrete random variable X with possible values $\{x_1, x_2, \ldots, x_n\}$ is defined as:

$$H(X) = -\sum_{i=0}^{n} p(x_i) log\, p(x_i)$$

where $p(x_i)$ is the probability of outcome x_i. Suppose one observes 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 one can conclude that the modifier of N_1 is more unpredictable than that of N_2. Similarly, I calculate an adjective's argument entropy based on the entropy of the noun given a fixed adjective.

I also observe the modification frequency of a noun using the parsed Wikipedia corpus described in Section 4.2.2. From the corpus, I 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 $p(mod_{other})$

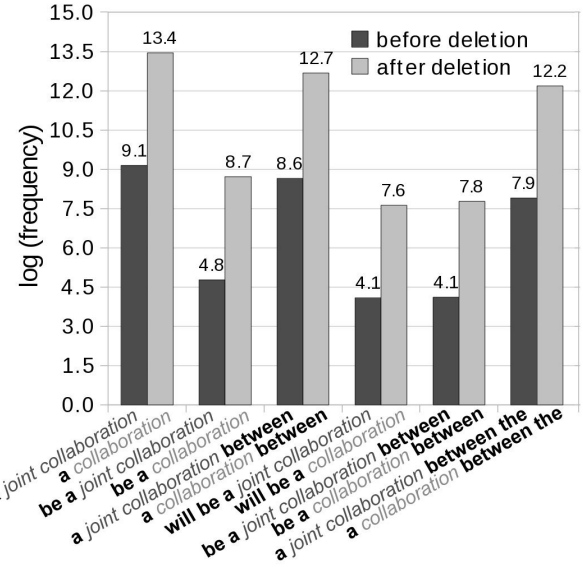


Figure 4.3: N-gram count distributions before and after deleting *joint*

is not included in the definition of $M(N)$ since I want to focus on the adjectival modification of a noun. Modifier entropy, argument entropy, modification entropy as well as the modification probabilities $p(mod_{adj})$, $p(mod_{other})$ and $p(mod_{non})$ are used as features for the SVM.

4.2.4 Contextual α-skew divergence

Another feature included in the SVM is the contextual n-gram divergence between the original text and the generated text after removing the target adjective. I 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 various n-gram counts. Figure 4.3 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. One can see that the counts have similar distributions before and after the deletion.

I use α-skew divergence (Lee, 1999) to calculate the n-gram distributional similarity between the original and the modified sentences. A brief introduction to the α-skew

divergence can be found in Section 3.1.2. Under my 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.

So far I have described the 67 features used to form a data point to represent an adjective deletion in the SVM. As suggested by Hsu et al. (2010), I scale feature values to the range [-1, +1]. However, I still need the class of each data point, namely deletable or undeletable, so the SVM can construct the maximum-margin hyperplane to separate the two classes during training. In the next section, I describe the collection of two labelled corpora for training, developing and testing the SVM classifier. In addition, the data is also used to test the proposed n-gram count method.

4.3 Adjective deletion data

I collected two labelled datasets in order to experiment with and evaluate the proposed adjective deletion checkers. Both datasets consist of independent sentences, each of which has an adjective labelled as either deletable or undeletable. Recall that the target adjective must meet my adjective-noun pair requirement and pass the syntactic filter as explained earlier. In addition, the adjectives labelled as deletable in the datasets mean that the removal of the adjective does not affect the naturalness of the resulting sentence. In contrast, if removing an adjective leads to an awkward sentence, the adjective is judged as undeletable.

4.3.1 Pilot study data

I first created a small dataset for a preliminary study. In order to experiment with redundant adjectives in sentences, I collected 90 sentences from the Internet, each of which contained an adjective-noun pleonasm.[4] 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.

[4] A collection of pleonasms can be found at http://www.pleonasms.com (last verified in June 2013).

Apart from deletable adjectives (positive data), I also need some undeletable adjectives (negative data) to test whether the n-gram count method and the SVM classifier have the ability to filter out bad deletions. My supervisor, a native English speaker, manually selected 76 undeletable adjectives in sentences from the British National Corpus (BNC)[5] 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 linguistic steganography. However, pleonasms are not general enough to be found frequently in text, which diminishes the amount of secret information which can be embedded in the text. Thus I collect more positive and negative data which are more frequent in text for training, developing and testing the proposed deletion checkers. This additional set serves as my main data source (described in the next section), with the pleonasm set serving as a useful pilot study.

4.3.2 Human annotated data

In order to have a more practical dateset than the pilot study data, I asked 30 native English speakers to judge whether a target adjective can be removed without affecting the naturalness of the rest of the sentence. The annotation was designed as web-based. At the beginning of the annotation task, I described the aim of the annotation and my definition of deletable and undeletable adjectives as follows:

Your task is to judge whether the removal of an adjective in a noun phrase significantly affects the naturalness of the resulting sentence. You will be shown 60 different sentences, each of which contains an adjective in brackets. Below each sentence, the same sentence will appear again, but this time with the adjective removed. Please indicate whether the sentence after the adjective removal looks unnatural or tampered with.

In addition, as part of the annotator instructions, I gave six example annotation sentences along with the judgements and explanations as shown in Table 4.3. After the introduction, each annotator was then shown 60 different sentences, each of which contained an adjective in brackets. Below each sentence, the same sentence appeared

[5]Distributed by Oxford University Computing Services on behalf of the BNC Consortium. URL: http://www.natcorp.ox.ac.uk (last verified in June 2013).

Example 1	He was putting on his [heavy] overcoat, asked again casually if he could have a look at the glass.
Judgement	In this case we say that *heavy* is deletable (YES), since the resulting sentence after deletion looks entirely natural.
Example 2	We are seeking to find out what [local] people want, because they must own the work themselves.
Judgement	In this case we say that *local* is deletable (YES), since the resulting sentence sounds natural even though the meaning is slightly changed.
Example 3	We are just at the beginning of the [worldwide] epidemic and the situation is still very unstable.
Judgement	In this case we say that *worldwide* is deletable (YES).
Example 4	He asserted that a modern artist should be in tune with his times, careful to avoid [hackneyed] subjects.
Judgement	In this case we say that *hackneyed* is not deletable (NO), since the meaning of the resulting sentence after deletion is odd semantically (because *hackneyed* is essential to the meaning of the original noun phrase).
Example 5	With various groups suggesting police complicity in township violence, many blacks will find [little] security in a larger police force.
Judgement	In this case we say that *little* is not deletable (NO), since although the resulting sentence sounds natural, the sentence as a whole is odd semantically (since the second clause is somewhat contradictory with respect to the first).
Example 6	There can be [little] doubt that such examples represent the tip of an iceberg.
Judgement	In this case we say that *little* is not deletable (NO), since the phrase "There can be doubt that" sounds unnatural.

Table 4.3: Judgement examples given to annotators

again, but this time with the adjective removed. Figure 4.4 shows a screen capture of an annotation example presented to a subject.

The sentences for creating the data were randomly selected from section A of the BNC other than those that were used as negatives in the pilot study data. I 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. I calculated the inter-annotator

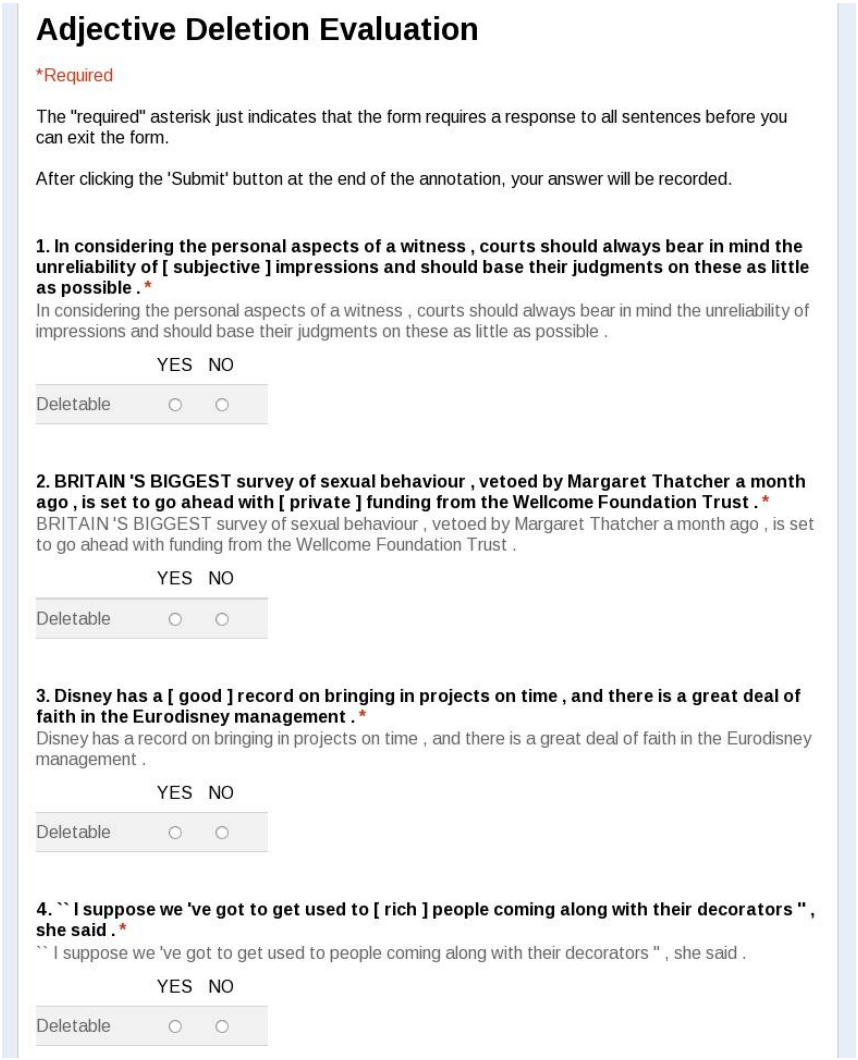


Figure 4.4: A screenshot of the deletable adjective annotation

agreement using Fleiss' kappa (Fleiss et al., 2003) scored between 0 and 1. Fleiss' kappa allows different items rated by different individuals with one restriction that every item receives the same number of assessments; that is, item 1 is judged by annotators A, B and C; while item 2 is judged by annotators D, E and F. For the 300 multi-judged instances, the Fleiss' kappa was 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 was around 2:1 for all the datasets.

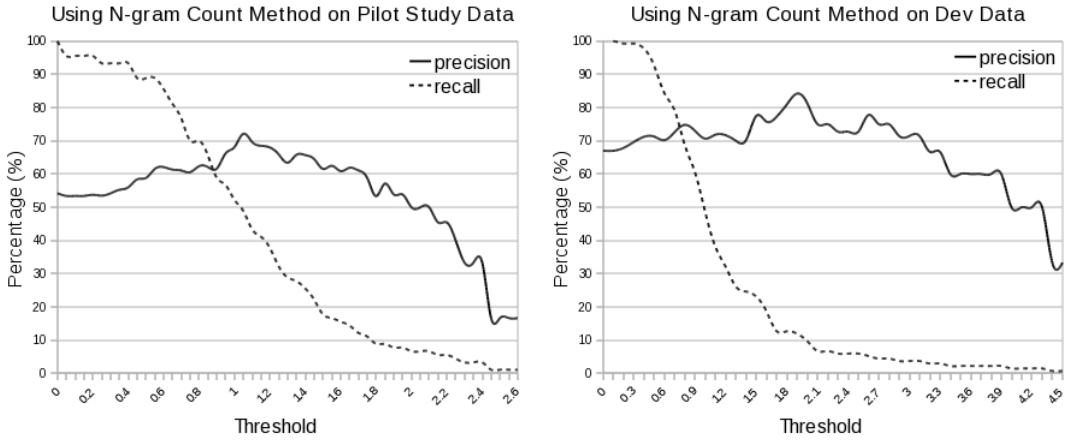
Since I now know the label of each deletion in the corpus, I can present those cases in the training set as labelled points to the SVM classifier and find the optimal hyperplane to separate deletable instances from undeletable ones. The hyperplane is then used to classify a given point, either deletable or undeletable depending on the new point's position in the space. In the next section, I first observe the performance of the n-gram count method on the pilot study data and the development set using various score thresholds. Next, I experiment with different SVM classifiers trained using different features on the development set. Finally, the best-performing SVM is tested on the test set.

4.4 Experiments and results

The performance on the adjective deletion task is measured using 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 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.

4.4.1 Experiments using the n-gram count method

I test the n-gram count method on the pilot study data and the development data. Figure 4.5(a) and Figure 4.5(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 of 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.



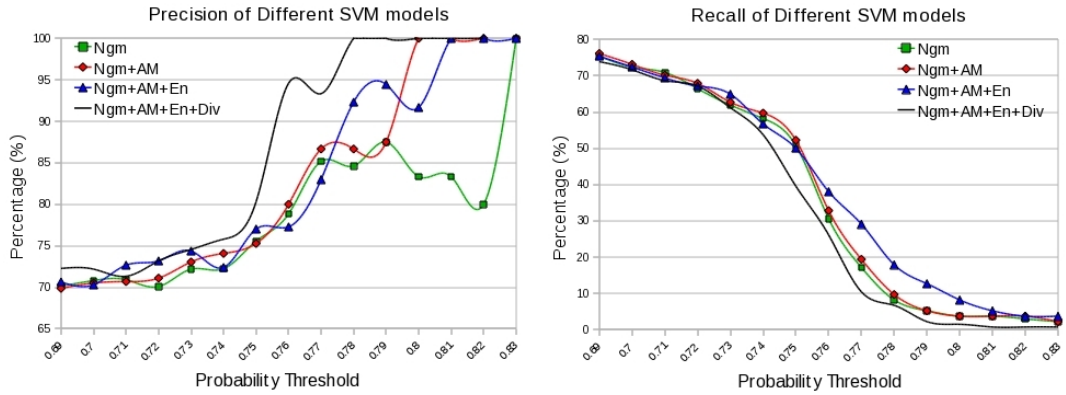
(a) Results on the pilot study data

(b) Results on the development data

Figure 4.5: Performance of the n-gram count method

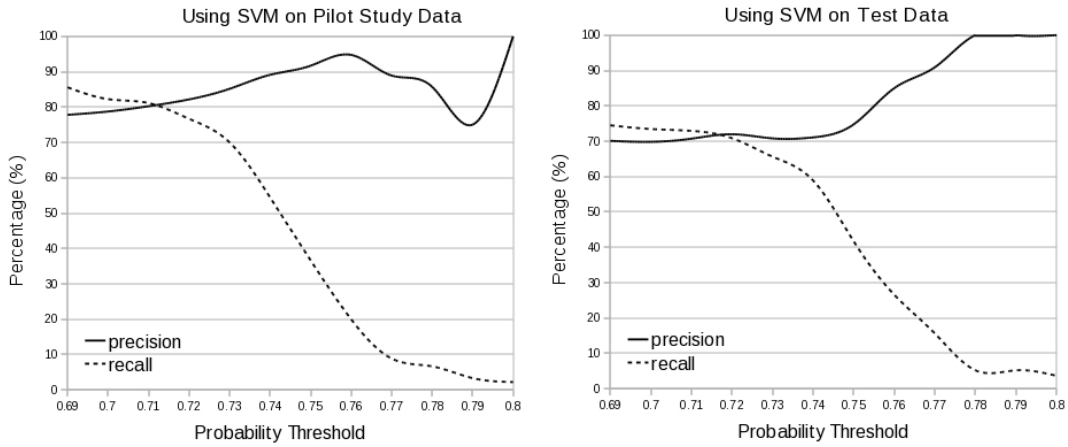
4.4.2 Experiments using the SVM classifier

For the SVM learning approach, I first train models with different features and test the models on the development data. As mentioned in Section 4.1.2 I train an SVM to output probability estimation; hence, a probability threshold is needed in order to determine an acceptable deletion. Figure 4.6(a) and Figure 4.6(b) show the precision and recall curves of the models with probability thresholds greater than 0.69 and lower than 0.83 (since these values result in a reasonable precision range). In addition, I 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 4.2.1. 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 I add the 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%. I then add features by including entropies and modification probabilities described in Section 4.2.3 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, I further evaluate this model using the pilot



(a) Precision curves of the models (b) Recall curves of the models

Figure 4.6: Performance of SVM models using different features



(a) Results on the pilot study data (b) Results on the test data

Figure 4.7: Performance of the Ngm+AM+En+Div model

study dataset and the test dataset.

Figure 4.7(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. One 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, I 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 4.7(b) shows the full range of precision-recall scores using different probability threshold values on the test data.[6] From this figure, one 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 fiction or fairy tales, which might increase the density of deletable adjectives in text.

With the proposed SVM model, I can determine which adjective is deletable in cover text. Therefore, different versions of a cover text can be generated by removing different adjectives each time, and these alternatives can represent different codewords depending on the coding method in a stegosystem. In the following sections, I first demonstrate a stegosystem that uses adjective deletion as the linguistic transformation and combines the deletion module with the coding method proposed in one of my earlier papers (Chang and Clark, 2010a). Later, I propose another cryptographic application for adjective deletion called secret sharing which distributes a secret bitstring among two comparable texts, and only when aligning the two texts can the secret be recovered.

4.5 Adjective deletion-based stegosystem

As explained in Chapter 2, for linguistic steganography there exists a convenient modularity between the linguistic transformation and the encoding method. In other words, the utility of a specific encoding method does not imply a particular linguistic transformation, although there might be some constraints on the transformation. In this

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

		Segmentation Size			
Text	Sentences	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Cover	$s_1\ s_2^d\ s_3\ s_4\ s_5^d\ s_6$	101	11	1	1
Alternative	$s_1\ s_2\ s_3\ s_4\ s_5^d\ s_6$	001	01	0	1
Alternative	$s_1\ s_2^d\ s_3\ s_4\ s_5\ s_6$	100	10	1	1
Alternative	$s_1\ s_2\ s_3\ s_4\ s_5\ s_6$	000	00	0	0

Table 4.4: An example of the Chang and Clark (2010a) segment encoding method

section, I will show that the adjective deletion technique can be integrated into a data encoding scheme proposed in my earlier paper (Chang and Clark, 2010a).

In Chang and Clark (2010a) we proposed a data encoding method based on text segmentation. The method first divides an n-sentence text into equal-sized segments, each of which contains k sentences and is called an embedding unit.[7] Each segment is then encoded by a one-bit codeword, namely either 0 or 1, depending on whether a linguistic transformation can be applied to a sentence in that segment; a segment represents bit 0 if no sentence in the segment can be changed, and bit 1 otherwise. In my case, an unchangeable segment is one that does not have any deletable adjectives, and a changeable segment is one that contains at least one deletable adjective.

Table 4.4 shows a cover text consists of six sentences $s_1, s_2, \ldots, s_6$ and three different versions of the cover text after removing different adjective(s), where a sentence with a superscript d means the sentence contains a deletable adjective. When $k = 2$, i.e. the size of an embedding unit is two sentences, each text represents a three-bit codeword as shown in the Table. However, there are no texts representing codewords 010, 011, 110 and 111. In other words, when $k = 2$, this segmentation coding does not provide sufficient codeword choices for the text selection module, which might lead to an embedding failure. In contrast, when $k = 3$, the four possible 2-bit codewords are individually represented by each of the four texts and, therefore, the text selection module can have a full range of choices while selecting the stego text. When $k = 4$ or $k = 5$, both bit 0 and bit 1 are encoded by at least one text so one secret bit can be successfully embedded. Note that when $k = 4$ or $k = 5$, the last segment does not contain enough words and is ignored.

[7] If the last segment has less than k sentences, it is ignored.

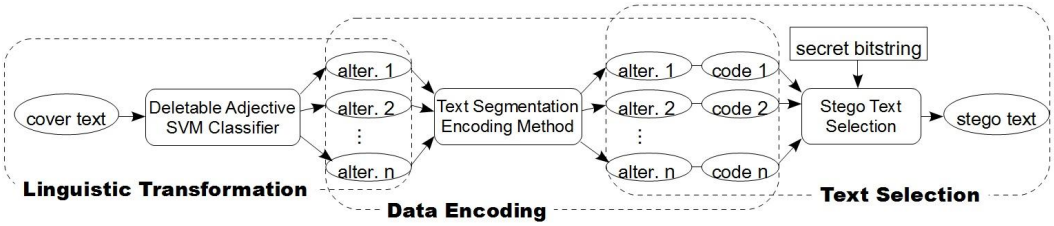


Figure 4.8: Framework of the proposed adjective deletion-based stegosystem

In order to avoid embedding failure, each embedding unit must have the ability to represent bit 0 and bit 1. In the previous example, when $k = 2$, the second embedding unit contains s_3 and s_4, both of which cannot be modified, and thus always encode bit 0. To solve this problem, in Chang and Clark (2010a) we define an applicable segment size k such that, for every k sentences in a cover text, there will be at least one changeable sentence. In this way, after encoding, all the embedding units in a cover text represent 1s, and since all the segments are changeable, a full range of $\lfloor \frac{n}{k} \rfloor$-bit codewords can be encoded by removing different adjective(s).

Figure 4.8 illustrates the framework of the adjective deletion-based stegosystem. During linguistic transformation, the system first finds deletable adjectives in a cover text using the proposed SVM classifier and generates different versions of the input text by removing deletable adjectives. Next, a valid segment size is used to define embedding units in each alternative and, according to the existence of deletable adjectives in the units, a corresponding codeword is assigned to an alternative. Finally, the system chooses one alternative that represents the secret bits as the stego text.

For the secret recovery, it is important to note that the receiver does not require the original text. The receiver only needs the segment size to define embedding units in the stego text and the adjective checking model to see whether there is a deletable adjective in an embedding unit. Therefore, in the proposed stegosystem the secret keys shared between the sender and receiver are the segment size k and the probability threshold used in the adjective deletion checker.

The embedding capacity of the segmentation method relies on the number of changeable sentences in the cover text. When every sentence in the cover text is changeable, the stegosystem can have the maximum payload equal to 1 bit per sentence. In addition, the embedding capacity of the system highly depends on the distribution of changeable sentences over the cover text. The maximum number of segments, i.e. the

maximum number of embedding bits, can be derived when the changeable sentences among the n cover sentences are uniformly distributed.

As an addition to the steganography application, in the next section I propose a novel linguistic secret sharing scheme based on adjective deletion. The scheme not only camouflages a secret bitstring in natural language, but also distributes the secret in two comparable texts.

4.6 Secret sharing scheme based on adjective deletion

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.[8] From the above two secret sharing schemes one can see that the share can be in different forms, such as numbers and points, depending on the methods used.

I propose a novel secret sharing scheme based on the adjective deletion technique and

[8] http://en.wikipedia.org/wiki/Shamir's_Secret_Sharing (last verified in June 2013).

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 4.9 illustrates an example of the secret sharing scheme. The secret bitstring is 101. I 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, I 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 I 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 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.

One of the contributions of this work is to explore the identification of deletable adjectives in noun phrases. I 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 my 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.

Another contribution of this chapter 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 my proposed system is that it is somewhat language and domain independent.

Apart from the cryptosystem applications, the proposed adjective checking model can

also benefit other studies such as sentence compression, text simplification and text summarisation which 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 word deletion operations in their systems. However, to my 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.

My first and second stegosystems exploit lexical substitution and adjective deletion, respectively, for generating alternatives for a cover text. Both of the two manipulations are word-level transformations. In the next chapter, I will introduce my third stegosystem that generates alternatives by changing word orders in a cover sentence, which belongs to a sentence-level transformation. As for the previous two stegosystems, I also propose a word-ordering checker to certify the naturalness of the transformation when applied to a particular sentence.

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 4.9: An example of the secret sharing scheme

Chapter 5

Word ordering

In this chapter I introduce my third linguistic stegosystem which exploits word ordering as the linguistic transformation.[1] The motivation of this work is that some sentences can be paraphrased by simply changing their word order without inserting, deleting or replacing any words. For example, a cover sentence "there is no asbestos in our products now" can be paraphrased to another grammatical sentence "in our products there is now no asbestos" using the same words, and the meaning conveyed by the original sentence is not significantly affected by the word reordering.

In order to discover possible word reordering in a cover sentence, I first turn a cover sentence into a set of un-ordered words: a bag-of-words. Then, a word ordering realisation system is used to construct possible sentence permutations from the bag-of-words. There have been some word ordering realisation systems that take a bag-of-words as input and automatically generate permutations (Wan et al., 2009; Zhang and Clark, 2011; Zhang et al., 2012). Existing realisation systems 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

[1] The content in this chapter was published as Chang and Clark (2012b).

Input bag-of-words: *asbestos*, *in*, *is*, *no*, *now*, *our*, *products*, *there*, .	
Permutations:	In our products now there is asbestos.
	In our products now there is no asbestos.
	No asbestos there is now in our products.
	Now in our products there is no asbestos.
	Our products there now is no asbestos in.
	There is no asbestos in our products now.
	There no asbestos in our products is now.
	There now is no asbestos in our products.

Table 5.1: Sentence permutations generated by Zhang et al. (2012) system

large-margin training and incorporating a large-scale language model. The above three realisation systems were evaluated using the generation task of word order recovery, which is to recover the original word order from an input bag-of-words. The evaluation metric of the generation task is the BLEU score which measures how similar the generated permutation and the original sentence are. Wan et al. (2009), Zhang and Clark (2011) and Zhang et al. (2012) reported BLEU scores of 33.7, 40.1 and 43.8, respectively, on Wall Street Journal newspaper sentences.

In this work, I 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. Table 5.1 shows eight permutations generated by the Zhang et al. (2012) system given a bag-of-words {*asbestos*, *in*, *is*, *no*, *now*, *our*, *products*, *there*, .}. Note that it is possible for a generated permutation to only contain a subset of the input words, which 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 significant incoherence in the resulting text. For example, the first permutation in Table 5.1 expresses the existence of asbestos in the products because it excludes the cover word *no*, which conveys the opposite meaning of the cover sentence "there is no asbestos in our products now". Therefore, in the proposed stegosystem, I only use permutations that include all the cover words. Another reason to only consider full-length permutations is because the proposed stegosystem relies on the receiver generating the same list of permutations as the sender, in order

to recover the secret bits. That is, the bag-of-*cover*-words and the bag-of-*stego*-words used by the sender and receiver, respectively, must be identical. If a shortened permutation is selected as the stego sentence, the bag-of-stego-words is different from the bag-of-cover-words, and thus, the secret recovery will fail. More details of the proposed word ordering-based stegosystem will be given in Section 5.5.

From Table 5.1 one can see that there are some unnatural sentences generated from the Zhang et al. (2012) system, e.g. "there no asbestos in our products is now." and "our products there now is no asbestos in." Using an unnatural permutation as the stego sentence significantly decreases the security level of a stegosystem. Therefore, it is crucial to develop a method that can distinguish acceptable permutations from those having awkward wordings in a word ordering-based stegosystem.

In the following sections I first explain a baseline word ordering checker using the Google n-gram corpus to check whether a particular word ordering has been used frequently on the Web. Then I propose another approach using some syntactic features as part of a Maximum Entropy classifier (Berger et al., 1996) to determine the naturalness of a permutation. As for the proposed substitution checkers and adjective deletion checkers, the word ordering checkers described in this chapter only tackle the problem of distinguishing the naturalness of a sentence permutation in isolation from the rest of a document; thus, it is possible that individual naturally sounding sentences might lead to an unnatural document. Modelling the document-level coherence of stego text would be useful but is outside the scope of my study. I evaluate both the baseline n-gram count method and the maximum entropy classifier using human judgements of sentence naturalness. The evaluation results can be seen as a reflection of the security level of the proposed word ordering-based stegosystem since the better the checker's performance is, the more natural the passed permutations are and, thus, the less suspicious the stego text may be. At the end of this chapter, I not only demonstrate the proposed word ordering-based stegosystem, but also show that the word ordering technique can be combined with the hash function coding method used in translation-based stegosystems (Grothoff et al., 2005; Stutsman et al., 2006; Meng et al., 2011).

5.1 Word ordering checkers

Since not all the permutations are grammatical and semantically meaningful, I propose two checking methods to distinguish natural word orders from awkward wordings. Both methods output a score/probability for a permutation and require a threshold to decide the acceptability of a permutation. It is important to note that my word ordering checkers take the original sentence into consideration when calculating a score/probability because the permutation filtering happens at the secret embedding stage where the sender knows what the cover sentence is; whereas the secret decoding does not require the word ordering checker. Having the cover sentence available at the checking stage is a feature I will exploit.

A research area that relates to the proposed permutation checking methods is realisation ranking (Cahill and Forst, 2010; White and Rajkumar, 2012) where a system is given a set of permutations (e.g. a set of sentence permutations or a set of paragraph permutations) and is asked to rate each text in the set. However, in the realisation ranking task there is no "cover text" for a ranking system to refer to. Since the proposed checking methods require the knowledge of the original text, they cannot be adapted to the realisation ranking task directly.

5.1.1 N-gram count method

My baseline checker is similar to that proposed in the previous two chapters 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 its corresponding permutation, decide if the permutation is acceptable in terms of naturalness. 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, I add <S> and </S> tags to the beginning and end of the sentences as shown in Figure 5.1. Then I extract every bi- to five-gram from the cover sentence and the permutation. Since I am only interested in newly generated wordings in the permutation, n-grams that appear

Cover: <S> There is no asbestos in our products now . </S>		Permutation: <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 5.1: An example of the n-gram count method for checking word reordering

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 5.1. I sum up all the logarithmic counts[2] for the cover and permutation n-grams and derive Sum_{Cover} and $Sum_{Permutation}$. The $Score_{reordering}$ of a permutation is defined as the ratio of $Sum_{Permutation}$ and Sum_{Cover}, which measures how much the Sum_{Cover} varies after performing the word ordering. In the given example, the $Score_{reordering}$ of the permutation is 0.81. 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, and it has worked well for my lexical substitution check.

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

5.1.2 Maximum entropy classifier

In addition to the baseline method, I propose a machine learning approach to classify natural and unnatural permutations. I 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. 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 class (y) occurring in the context (x) to estimate a probability model $p(y|x)$. Its canonical form is:

$$p(y|x) = \frac{1}{Z(x)} \, exp \sum_i \lambda_i f_i(x, y)$$

where $Z(x)$ is a normalisation constant to ensure a proper probability distribution and λ_i is the weight of the feature $f_i(x, y)$. In the formulation of MaxEnt I use, a feature f_i is an "indicator function", a boolean matching function with the value of either 0 or 1, which simply indicates the presence of a "contextual element" together with a particular class y. In my naturalness classification scenario, contextual elements can be any attributes in a permutation sentence x, such as n-grams, grammatical relations or supertag sequences, and y is a class label, either *natural* or *unnatural*.

The training process of a MaxEnt classifier is to choose parameters λ_i that maximize the conditional likelihood of the training data, which is equivalent to picking the most uniform model subject to constraints on the feature expectations (Berger et al., 1996). The MaxEnt implementation I used was from the Curran and Clark (2003) tagger, adapted for classification rather than sequence tagging. After training a MaxEnt classifier, I can calculate the probabilities of a permutation being a natural sentence according to the feature weights. My 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 MaxEnt classifier and can be decided by steganography users.

As explained above, to train a MaxEnt classifier, the first step is to identify a set of indicator functions $f_i(x, y)$ from the training data. In the next section, I describe the

contextual elements that I extract from a permutation. Later, I describe the human annotation for collecting the labels of my experiment sentences.

5.2 Features for the maximum entropy classifier

Recall that my word ordering checkers take the original sentence into consideration since the checking is only performed at the sender side, and the sender has access to the cover text. The first attribute I extract from a permutation 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 observation "EDIST_D" becomes the attribute 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 attribute "EDIST_2". In addition, if the difference between a permutation and its original sentence is only a single word movement, I add the POS tag of the moved word to the attribute, so the feature becomes "EDIST_1-*POS*".

The second type of contextual element 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*. I would expect that using grammatical relations described in Chapter 4 will not make much difference to the results; one of the reasons for choosing Stanford parser and the Stanford typed dependencies is to avoid using the same source for training the classifier and for selecting the experiment data where grammatical relations are used as described in Section 5.3.

I 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 contextual elements "P_dep_*TYPE*" and "P_deppos_*TYPE_POS1_POS2*" are added to the permutation's attribute set, where

Cover:.	Permutation:
There is no asbestos in our products now.	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 attributes 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 5.2: An example of the dependency indicator functions

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 contextual elements "R_dep_*TYPE*" and "R_deppos_*TYPE_POS1_POS2_DISTANCE*" are added to the permutation's attribute 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 5.2 shows the dependency attributes 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 contextual elements *P_dep_nsubj* and *P_deppos_nsubj_NN_NNS* are added to the permutation's attribute 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 original so two contextual elements *R_dep_poss* and *R_deppos_poss_NNS_PRP$_0* are added to the permutation's attribute set.

The indicator functions $f_i(x, y)$ for the MaxEnt classifier are derived from permutation-class pairs. So far I have described the attribute set observed from a permutation x. I

still need the label y of each permutation, namely natural or unnatural, in order to form indicator functions. In the next section, I describe the collection of labelled permutations for training, developing and testing the MaxEnt classifier. In addition, the data is also used to test the baseline n-gram count method.

5.3 Human annotated data

In order to have a labelled corpus for training, developing and testing the MaxEnt classifier, I 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 permutations 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, I 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, I 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. I 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).

The annotations were carried out via a web interface. A total of 34 native English speakers were asked to judge the naturalness of the sentence permutations on a 4-point scale. The instructions given to the annotators are as follows:

Your task is to judge the naturalness of sentences on a four-point scale. Each time you

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.

Table 5.2: Rating scale and guidelines for human evaluation

Score	Sentence and Judgement explanation
2	The yield on six-month Treasury bills sold at Monday's auction from 8.04%, for example, rose to 7.90%. *explanation: relatively fluent, but slightly illogical: from 8.04% to 7.90% is a drop, not a rise*
1	8.04% rose 7.90% from, for example to, the yield on six-month Treasury bills sold at Monday's auction. *explanation: completely non-fluent*
1	The yield on six-month Treasury bills sold at Monday's auction, for example, rose 8.04% from 7.90% to. *explanation: completely non-fluent*
2	The yield on six-month Treasury bills sold at Monday's auction, for example rose to 8.04% from 7.90%,. *explanation: major punctuation error ",."*
4	The yield on six-month Treasury bills sold at Monday's auction from 7.90%, for example, rose to 8.04%. *explanation: natural*

Table 5.3: Some of the judgement examples given to annotators

will be shown 5 sentences consisting of the same words, but arranged in a different order, or with different punctuation. You will be given a total of 25 sets of sentences. Please note that sentences are presented in a random order, and it is possible that all the 5 sentences in the same set are all unnatural, or all natural.

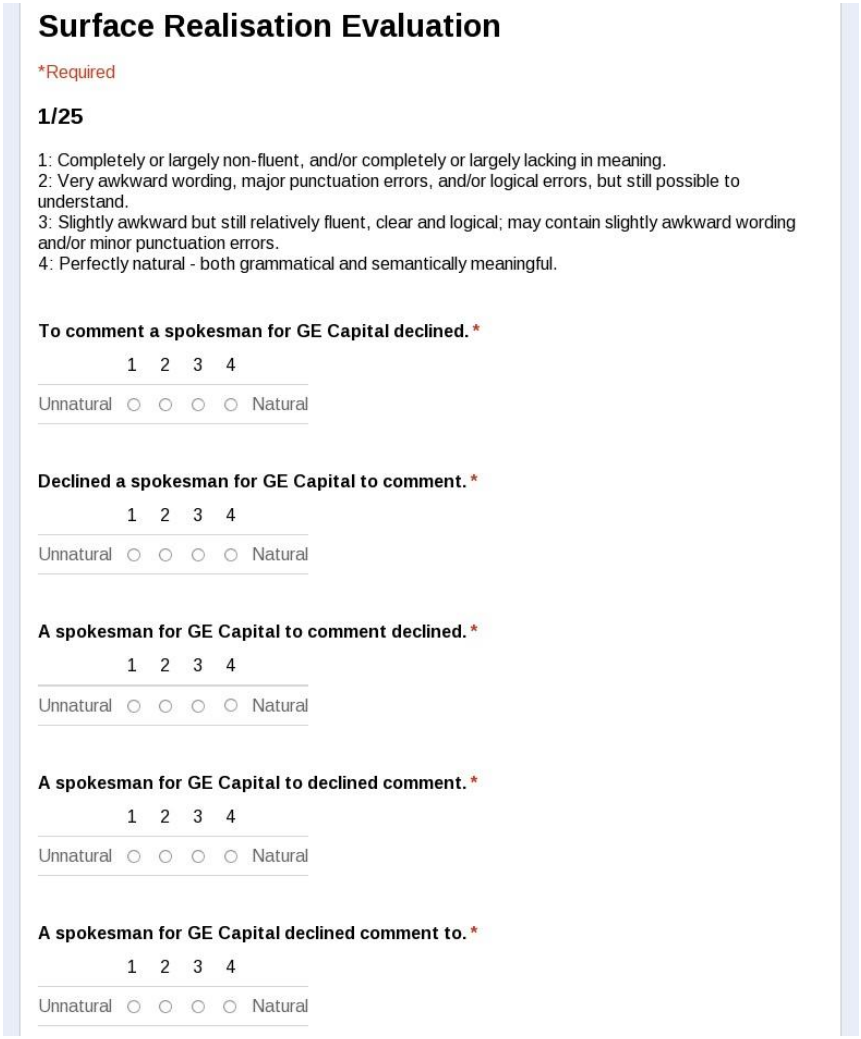


Figure 5.3: A screenshot of the word ordering annotation

Table 5.2 shows the guideline for each judgement score provided to the annotators. In addition, as part of the annotator instructions, I gave some example annotation sentences along with my judgements and explanations as shown in Table 5.3. After the introduction, 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 5.3 shows a screen capture of a set of five sentences presented to 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, I 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 my supervisor 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 2.3, I observe how well the BLEU score of a permutation correlates with the human judgement (for those multi-judged sentences, an average score is assigned). The BLEU metric (Papineni et al., 2002) was originally designed for automatic evaluation of machine translation. It measures the n-gram precisions of a translation given the reference sentence(s) and the final score is an interpolation of different n-gram precisions with a brevity penalty. Here I treat a permutation as a translation and its original sentence as the reference so that the BLEU score of a permutation can be obtained. Both Pearson's correlation coefficient (Pearson, 1920) and Spearman's rank correlation coefficient (Spearman, 1910) 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.

I divided the collected human judgement corpus 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 and for deciding a score/probability

	Training Set	Development Set	Test Set
Number of Positives	467	52	90
Number of Negatives	2,364	298	685

Table 5.4: Statistics of the experimental data sets

threshold in the proposed checkers. Note that the 425 multi-judged sentences are all included in the test set. Since the number of negatives is 7 times more than the number of positives in the training set, I added another 131 positives annotated by ourselves to the training set (but not the test set), in an attempt to address the imbalance. Table 5.4 presents the statistics of the data sets.

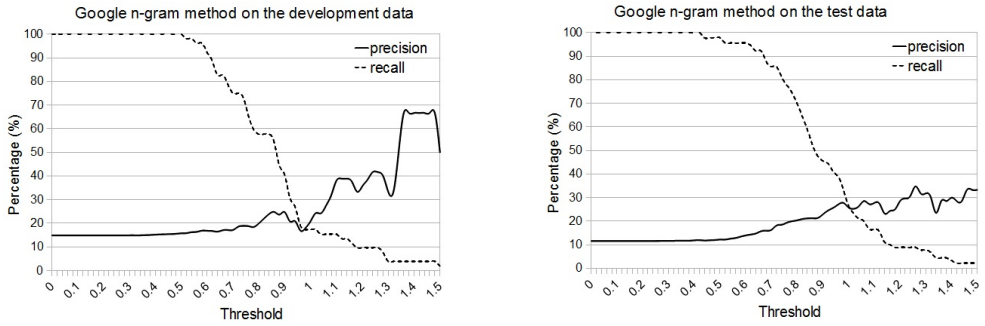
Now I know the class of each permutation in the training set, I can present the observed indicator functions to the MaxEnt classifier and find the optimal weight for each feature. In the next section, I describe the evaluation of the proposed baseline n-gram count method and the MaxEnt classifier and present the experiment results.

5.4 Experiments and results

I 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).

5.4.1 Experiments using the n-gram count method

I first evaluated the Google n-gram method on the development set. Figure 5.4(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 I 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 5.4(b) gives the precision and recall curves obtained by using the Google n-gram method on the test data. From the diagram one can see that, even when a threshold 1.26 is chosen, the best precision on the test set is only 34.8%,



(a) Results on the development set

(b) Results on the test set

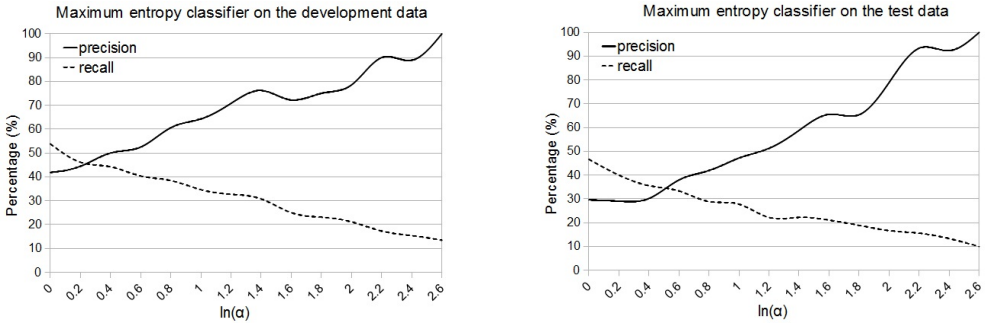
Figure 5.4: Performance of the n-gram count method

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 for 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 my 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.

5.4.2 Experiments using the maximum entropy classifier

Next I train a maximum entropy classifier using sentences in the training set. Each permutation in the training set is first represented by its indicator functions. A total of 5,815 indicator functions are extracted from the training set. As mentioned in Section 5.1.2, the idea of a MaxEnt model is to maximize the conditional entropy subject to a set of expectation constraints (with respect to different feature functions). Those constraints force the model to be consistent with the training data. However, in my training set the ratio of positives to negatives is about 1:5, and since there is less positive data, how well the model fits the positive data has less impact on the final model. To solve this issue, I duplicate the positives five times to balance the amount of positives and negatives in the training set, which is equivalent to assigning a higher



(a) Results on the development set

(b) Results on the test set

Figure 5.5: Performance of the maximum entropy classifier

weight to each positive data point so that positive and negative data can have similar influence in the weighted objective function. The trained weights are then used to calculate the probabilities of a test instance being positive and negative. As mentioned in Section 5.1.2, the system determines an instance as positive if:

$$\frac{exp\sum_{i=1}^{n}\lambda_i f_i(x, natural)}{exp\sum_{i=1}^{n}\lambda_i f_i(x, unnatural)} > \alpha$$

$$\Longrightarrow \sum_{i=1}^{n}\lambda_i f_i(x, natural) - \sum_{i=1}^{n}\lambda_i f_i(x, unnatural) > ln(\alpha)$$

I observe the precision and recall values of the classifier with different α values. Figure 5.5(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 5.3, 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, I 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 n-gram count method. Since the training data used in my 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, I 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 5.5(b). Note that I am not optimising on the test set; Figure 5.5(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.

With the proposed MaxEnt classifier, I can determine which permutation of a cover sentence is acceptable, and the passed permutations provide a possible covert channel for secret communication. In the following sections, I first demonstrate the proposed stegosystem based on word ordering, and then I show that the word ordering technique can be used in conjunction with existing translation-based encoding methods.

5.5 Word ordering-based stegosystem

As explained at the beginning of this chapter, a permutation that does not include all the cover words might lose important information conveyed by the original sentence. In addition, the secret recovery in the proposed steganography scheme relies on the sender and receiver using the same bag-of-words to generate the same list of permutations as demonstrated below. Therefore, only those permutations having the same length as the cover are considered in my stegosystem.

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. In addition, the number of secret

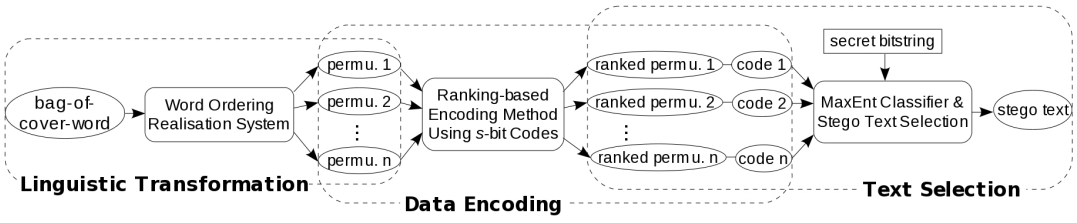


Figure 5.6: Framework of the proposed word ordering-based stegosystem

bits carried by a permutation must be fixed and known by the sender and the receiver.

Figure 5.6 illustrates the framework of the proposed word ordering-based stegosystem. During the data embedding, the sender first turns a cover sentence into a bag-of-words and then uses the 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 5.7 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 5.7 secret bitstrings 000 and 111 cannot be found when s is equal to 3. Finally, the text selection module chooses a permutation that represents the secret bitstring, and is determined as natural by the MaxEnt classifier, as the stego sentence. However, it may be the case that no permutation in the secret-bitstring group is natural. In this situation, the cover sentence will be used instead, and error detection codes (Klve, 2007) which enables the receiver to verify the extracted secret must be added to the transmission data.

To recover the secret bitstring, the receiver first transforms the received stego sentence into a bag-of-*stego*-words. Since I only consider permutations having the same length as the cover during embedding, the bag-of-*stego*-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.

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 5.7: Ranked sentence permutations and their secret bits

The payload of the system is controlled by the variable s, and the security level depends on the quality of the selected permutations. One of the differences between this word ordering-based stegosystem and the other two stegosystems proposed in the previous chapters is that the word ordering checker is integrated in the text selection module; while both the substitution checker and adjective deletion checker are applied during text transformation. The reason for the difference is that the proposed MaxEnt classifier requires information from the original text and can only be applied by the sender. In addition, the data encoding in the proposed word ordering-based stegosystem depends on the rank of a permutation; if the sender filters out bad permutations before data encoding, the rank of a permutation might be different from that derived by the receiver since the receiver cannot perform the word ordering check. In the next section, I demonstrate another word ordering-based stegosystem using a hash function as the coding method. In this system the MaxEnt classifier can be applied to the linguistic transformation module without affecting the secret recovery.

5.6 Using word ordering in translation-based embedding

In Section 2.2.4 I explained the hash function encoding used in existing translation-based stegosystems (Grothoff et al., 2005; Stutsman et al., 2006; Meng et al., 2011). This encoding method exploits a hash function to map a translation to a codeword,

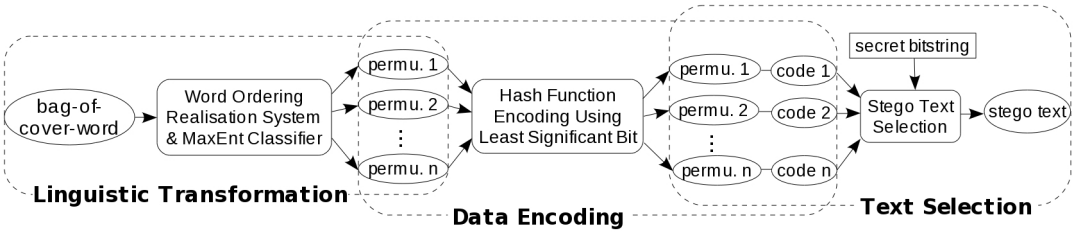


Figure 5.8: Framework of the stegosystem using word ordering and hash function encoding

which is independent of the rest of the alternatives; whereas the ranking-based encoding method described in the previous section considers the whole set of alternatives when assigning codewords. Since I can treat the word ordering technique as a "monolingual translation" that translates a cover sentence into different permutations, I can easily replace a machine translation system in a translation-based stegosystem with a word ordering realisation system.

Figure 5.8 illustrates the framework of using the word ordering technique in the translation-based embedding algorithm proposed by Grothoff et al. (2005). During the data embedding, the word ordering realisation system generates several permutations for a given cover sentence and the MaxEnt classifier is applied to eliminate unnatural permutations. Note that permutations are not required to have the same length as the original in this stegosystem. Next, a hash function maps each passed permutation to a bitstring, and the least significant bit of that bitstring is the the codeword of the permutation. Finally, a permutation 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. This basic embedding scheme has an upper bound of 1 bit per sentence. The problem faced by this stegosystem is the same as that in the previous system: it is possible that there is no permutation-carrying codeword that matches the secret bitstring since the generation of a desired bitstring cannot be guaranteed. Therefore, error detection codes must be used in this protocol when there is no feasible hash code available, which increase the size of transmission data.

A contribution of this work is to create a novel link between word ordering realisation

and linguistic steganography. In addition, I propose a maximum entropy classifier to determine the naturalness of a permutation. The evaluation results suggest that the proposed maximum entropy classifier can provide a high security level for the linguistic steganography application.

Chapter 6

Conclusions and Future Work

The aim of this thesis is to explore possible linguistic transformations for the steganography application. I have demonstrated the applicability of three different transformations: lexical substitution, adjective deletion and word ordering. As explained at the beginning of the thesis, transformations made to the cover text must be natural to an outside observer while the meaning of the cover text does not have to be preserved, which makes linguistic steganography a distinct task from other NLP tasks (e.g. text summarisation, paraphrasing or simplification) that not only require generated text to be linguistically acceptable but also convey the meaning of the original text. For each transformation, I proposed different checking methods to certify the sentence naturalness after the modification. In addition, I have demonstrated how to combine the transformations with the proposed encoding methods, and some existing encoding methods, in order to form different stegosystems. In the following section I summarise the main contributions of the thesis.

6.1 Contributions of the thesis

Generalising and summarising linguistic steganography: In Chapter 2, I first defined linguistic steganography as a combination of three independent modules: linguistic transformation, data encoding and text selection, the modularity of which is an important feature in designing linguistic stegosystems. Then I surveyed three major categories of linguistic transformations as well as various encoding methods that

have been exploited by existing linguistic stegosystems. In addition, I summarised the methods of evaluating imperceptibility and the payload capacity of existing systems. Chapter 2 provided the general framework and literature review for readers unfamiliar with linguistic steganography.

Lexical substitution checkers: Lexical substitution was the first linguistic transformation I explored. In Chapter 3, I described my n-gram count-based checker and the hybrid approach which combines the n-gram count method with contextual α-skew divergence. Both methods were first evaluated by a substitution ranking task. The n-gram count method and hybrid method achieved the GAP values of 49.7% and 50.8% on the task, respectively, which outperformed the GAP values of 38.6% and 42.9% derived by the Erk and Padó (2010) and Dinu and Lapata (2010) systems, respectively. The proposed methods were also evaluated by a naturalness classification task, which is more related to the steganography application than the ranking task. This time, the n-gram count method performed better than the hybrid method over all POS that I considered. Therefore, the n-gram count method was further assessed by a human evaluation. The results showed that the substitutions checked by the proposed n-gram count method received an average score of 3.33 out of 4; while without checking, the changed sentences got an average score of 2.82. Not only can the proposed checkers benefit substitution-based stegosystems, but also other applications that require the measurement of word similarity, such as document retrieval, machine translation and word sense disambiguation.

Adjective deletion checkers: In Chapter 4, I proposed two methods for checking deletable adjectives in noun phrases. The first approach exploits the Google n-gram corpus and the second method trains an SVM classifier. Both methods were evaluated against human judgements. The experimental results showed that the SVM classifier performed better than the n-gram count method and achieved a precision of 94.7% and 85% on the pilot study data and test data, respectively; while the precision baselines in the pilot study data and test data were 54.2% and 64.0%, respectively. Not only can the proposed checkers benefit deletion-based stegosystems, but also other NLP applications such as sentence compression, text simplification and text summarisation which usually aim at generating concise text and require the removal of unimportant words.

Word ordering checkers: The third linguistic transformation I explored is word ordering. I proposed a method using a MaxEnt classifier to determine the naturalness of a sentence permutation in Chapter 5. Again, the proposed MaxEnt classifier was evaluated by human annotated data and was compared with a baseline method based on the Google n-gram counts. The results showed that the MaxEnt classifier improved the performance of the baseline method significantly and achieved 93.3% precision with 15.6% recall on the test data; while the precision and recall values of the baseline method were 28.6% and 4.4%, respectively.

Human annotated data: Many NLP tasks require human judged data for evaluating a system, which is expensive to collect. In this thesis, I have collected three corpora where native English speakers were asked to evaluate the naturalness of sentences after undergoing lexical substitution, adjective deletion or word ordering. The corpora are available on the Web[1] and can be used as a gold standard to test other NLP systems.

Data encoding methods: From an information security perspective, I have proposed different novel data encoding methods which can be combined with other linguistic transformations because of the convenient modularity in linguistic steganography. For lexical substitution, I proposed the vertex colouring coding method; for word reordering, I proposed the ranking-based coding method. In addition, I developed a novel linguistic secret sharing scheme based on adjective deletion.

Stegosystem evaluation: Many existing works are proof-of-concept implementations with little practical evaluation of the imperceptibility or embedding capacity. Another contribution of the thesis is the practical evaluation of the proposed stegosystems. Although I did not directly evaluate the quality of the stego text, I have demonstrated the feasibility of using lexical substitution, adjective deletion and word ordering to generate natural sounding sentences and the frequency with which these transformations can be made to text.

[1]The human judgement corpora are available at www.cl.cam.ac.uk/∼cyc30 (last verified in June 2013).

6.2 Future work

All the proposed transformation checkers were evaluated against human judgements of sentence naturalness. For each evaluation, I used precision-recall curves to demonstrate the trade-off between imperceptibility and payload for the linguistic steganography application. From those diagrams, one can see that high precision values can be reached by the transformation checkers, which means a certain level of security can be achieved by integrating the proposed checking methods into the stegosystems. However, the corresponding recall values are around 20% which means many good transformations are ignored by the proposed checkers and therefore potential information carriers are wasted. Therefore, one possible extension of this thesis is to improve the recall, namely the embedding capacity of a stegosystem. For example, one may includes more features into machine learning classifiers, such as distributional semantic information from the target adjective and its argument. In addition, in this thesis I only evaluated the transformations in terms of the sentence-level naturalness rather than meaning retention and document-level coherence. Therefore, it would be interesting to see to what extent the proposed transformation checkers are useful for the security of linguistic steganography at the document-level.

Apart from the linguistic transformations discussed in this thesis, I would like to explore more manipulations that can meet the requirements of linguistic steganography. For example, in Chapter 4 I only focused on finding deletable adjectives in noun phrases. According to the pilot study, many adverbs and punctuation (e.g. comma) are redundant in text; for instance, "actually, I have already said: I cannot go" and "I have said I cannot go" convey similar meaning. Therefore, exploring more lexical redundancies in other POS is one possible future direction.

In addition, instead of finding redundancies in text, a system that can automatically insert unnecessary words into text may benefit linguistic steganography as well. So far many NLP systems have attempted to compress or summarise text by making text shorter. No automatic insertion system has being developed. For linguistic steganography, word insertion is a possible transformation and may create more alternatives for a cover text, increasing the payload capacity. I have carried out a preliminary study on automatic word insertion. The system first exploits the Google n-gram corpus to find possible words for an insertion position in context, and then uses a grammaticality

check and an n-gram count method, similar to that proposed for checking deletable adjectives, to measure how much the contextual n-gram count changes after inserting a particular word. Although some of the system output seems to be promising, I recognize the necessity for having a model to capture the semantic relation between an inserted word and the sentence. For example, my insertion system generates the sentence "well , the two cars are jam-packed bumper to bumper all the way from Beirut to Tyre" by inserting the word *two*. However, in reality two cars would not be jam-packed bumper to bumper and thus the modification can be easily spotted by a third party in a linguistic steganography scheme. My feeling is that in general word insertion is a harder problem than word deletion in the sense that word insertion requires more understanding of the added information in order to maintain the naturalness of the sentence in the context of the document and the world.

As mentioned in Section 2.3, there is no research on the practical issue of using different types of cover text for the steganography application. Thus, it would be interesting to see whether some types of cover text are better suited to linguistic steganography than others. Another interesting question that I have not addressed is whether some languages are easier to be modified than others, or whether some languages work better with particular linguistic transformations than others.

Linguistic steganography is a rather new research area, and further efforts are needed to develop more secure and efficient systems. The novel and original ideas provided in this thesis can benefit research in both computational linguistics and information security. It is hoped that my work can form the basis for more research devoted to linguistic steganography.

Bibliography

Atallah, M. J., McDonough, C. J., Raskin, V., and Nirenburg, S. (2000). 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. (2001). 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.

Bennett, K. (2004). Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text. Technical report, Purdue University.

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

Bergmair, R. (2004). Towards linguistic steganography: A systematic investigation of approaches, systems, and issues. Final year thesis, B.Sc. (Hons.) in Computer Studies, The University of Derby.

Bergmair, R. (2007). A comprehensive bibliography of linguistic steganography. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, California.

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.

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

Bohnet, B. (2009). Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 67–72, Boulder, Colorado.

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.

Briscoe, T. (2006). An introduction to tag sequence grammars and the RASP system parser. Technical report, Computer Laboratory, University of Cambridge.

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.

Callison-Burch, C. (2008). Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the EMNLP Conference*, pages 196–205, Honolulu, Hawaii.

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.-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. (2010). Linguistic steganography using automatically generated paraphrases. Masters Thesis. University of Oxford.

Chang, C.-Y. and Clark, S. (2010a). Linguistic steganography using automatically generated paraphrases. In *Proceedings of the Annual Meeting of the North American 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.

Chang, C.-Y. and Clark, S. (2012a). Adjective deletion for linguistic steganography and secret sharing. In *Proceedings of the 24th International Conference on Computational Linguistics*, Mumbai, India.

Chang, C.-Y. and Clark, S. (2012b). The secret's in the word order: Text-to-text generation for linguistic steganography. In *Proceedings of the 24th International Conference on Computational Linguistics*, Mumbai, India.

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, China.

Chapman, M., Davida, G. I., and Rennhard, M. (2001). A practical and effective approach to large-scale automated linguistic steganography. In *Proceedings of the 4th International Conference on Information Security*, pages 156–165, Malaga, Spain.

Chen, Z., Huang, L., Meng, P., Yang, W., and Miao, H. (2011). Blind linguistic steganalysis against translation based steganography. In *Proceedings of the 9th international conference on Digital watermarking*, pages 251–265, Seoul, Korea.

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.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cox, I., Miller, M., Bloom, J., Fridrich, J., and Kalker, T. (2008). *Digital Watermarking and Steganography*. Morgan Kaufmann Publishers Inc., second edition.

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

De Marneffe, M., MacCartney, B., and Manning, C. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth Conference on Language Resources and Evaluation*, volume 6, pages 449–454, Genoa, Italy.

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, Manchester, UK.

Dinu, G. and Lapata, M. (2010). Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172, Cambridge, MA.

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

Erk, K. and Padó, S. (2010). Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 92–97, Uppsala, Sweden.

Fellbaum, C. (1998). *WordNet: An electronic lexical database*. MIT Press, first edition.

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

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.

Gould, R. J. (1988). *Graph theory*. Benjamin/Cummings Pub. Co., Menlo Park, CA.

Grothoff, C., Grothoff, K., Alkhutova, L., Stutsman, R., and Atallah, M. J. (2005). Translation-based steganography. In *Proceedings of the 2005 Information Hiding Workshop*, pages 219–233, Barcelona, Spain.

Hearst, M., Dumais, S., Osman, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28.

Herodotus (1987). *The History*, chapter 7, page 279. University of Chicago Press. Translated by David Grene.

Hoover, J. E. (1946). The enemy's masterpiece of espionage. *The Reader's Digest*, 48:49–53. London 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.

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.

Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, Germany. Springer.

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, Maryland, USA.

Kahn, D. (1967). *The codebreakers: the story of secret writing*. Macmillan.

Kerckhoffs, A. (1883). La cryptographie militaire. *Journal des sciences militaires*, IX:5–83.

Khairullah, M. (2009). A novel text steganography system using font color of the invisible characters in Microsoft Word documents. In *Second International Conference on Computer and Electrical Engineering*, pages 482–484, Dubai.

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.

Kirk, R. E. (2012). *Experimental Design: Procedures for the Behavioral Sciences*. SAGE Publications, Inc, fourth edition.

Kishida, K. (2005). Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. Technical report, National Institute of Informatics.

Klve, T. (2007). *Codes for Error Detection*. Series on Coding Theory and Cryptology Series. World Scientific.

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

Kudo, T. and Matsumoto, Y. (2000). Japanese dependency structure analysis based on support vector machines. In *Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 18–25, Hong Kong.

Kullback, S. (1959). *Information theory and statistics*. John Wiley and Sons, NY.

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.

Lampson, B. W. (1973). A note on the confinement problem. *Communications of the ACM*, 16(10):613–615.

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.

Leopold, E. and Kindermann, J. (2002). Text categorization with support vector machines. How to represent texts in input space? *Machine Learning*, 46(1):423–444.

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.

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*, chapter 2 and 5. MIT Press.

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

McCarthy, D. and Navigli, R. (2007). SemEval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53, Prague, Czech Republic.

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, Calgary, Alberta, Canada.

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, Prague, Czech Republic.

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 International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, California.

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. Masters 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 International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, California.

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, Vancouver, BC, Canada.

Newman, B. (1940). *Secrets of German espionage*. Robert Hale Ltd.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318, Philadelphia, PA.

Pearson, K. (1920). Notes on the history of correlation. *Biometrika*, 13(1):25–45.

Pfitzmann, B. (1996). Information hiding terminology: Results of an informal plenary meeting and additional proposals. In *Proceedings of the First International Workshop on Information Hiding*, pages 347–350, Cambridge, UK.

Platt, J. C. (1999). *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods*, pages 61–74. MIT Press.

Por, L. Y., Fong, A. T., and Delina, B. (2008). Whitesteg: a new scheme in information hiding using text steganography. *WSEAS Transactions on Computers*, 7:735–745.

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, Philadelphia, PA, USA.

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

Schuler, K. K. (2005). *Verbnet: a broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania. Philadelphia, PA, USA.

Shahreza, M. S. (2006). A new method for steganography in HTML files. *Advances in Computer, Information, and Systems Sciences, and Engineering*, pages 247–252.

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

Sharoff, S. (2006). Open-source corpora: Using the net to fish for linguistic data. *International Journal of Corpus Linguistics*, 11(4):435–462.

Shen, L., Satta, G., and Joshi, A. (2007). Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, Prague, Czech Republic.

Shih, F. Y. (2008). *Digital watermarking and steganography: fundamentals and techniques*. CRC Press.

Simmons, G. J. (1984). The prisoners' problem and the subliminal channel. In *Advances in Cryptology: Proceedings of CRYPTO '83*, pages 51–67, Santa Barbara, California, USA.

Soderstrand, M. A., Jenkins, K. W., Jullien, G. A., and Taylor, F. J. (1986). *Residue number system arithmetic: modern applications in digital signal processing*. IEEE Press.

Søgaard, A. (2010). Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208, Uppsala, Sweden.

Spearman, C. (1910). Correlation calculated from faulty data. *British Journal of Psychology*, 3(3):271–295.

Spoustová, D. j., Hajič, J., Raab, J., and Spousta, M. (2009). Semi-supervised training for the averaged perceptron POS tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 763–771, Athens, Greece.

Stevens, G. (1957). *Microphotography: photography at extreme resolution*. Chapman & Hall.

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 International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6072, pages 97–105, San Jose, California.

Tissandier, G. (1874). *Les merveilles de la photographie*. Bibliothèque des merveilles: Hachette. Librairie Hachette & Cie.

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 International Conference on Security, Steganography,*

and Watermarking of Multimedia Contents, volume 5681, pages 441–452, San Jose, California.

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, California.

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.

Toutanova, K., Klein, D., Manning, C., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Edmonton, Canada.

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, Sydney, Australia.

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 International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, volume 6505, San Jose, California.

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.

Wayner, P. (1992). Mimic functions. *Cryptologia*, XVI(3):193–214.

Wayner, P. (1995). Strong theoretical steganography. *Cryptologia*, XIX(3):285–299.

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.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Williams, E. (1984). Grammatical relations. *Linguistic Inquiry*, 15(4):639–673.

Winstein, K. (1999). Tyrannosaurus lex. Open source.

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.

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.