Practical Linguistic Steganography using Contextual Synonym Substitution and a Novel Vertex Coding Method

Ching-Yun Chang*
University of Cambridge

Stephen Clark**
University of Cambridge

Linguistic steganography is concerned with hiding information in natural language text. One of the major transformations used in linguistic steganography is synonym substitution. However, few existing studies have studied the practical application of this approach. In this article we propose two improvements to the use of synonym substitution for encoding hidden bits of information. First, we use the Google n-gram corpus for checking the applicability of a synonym in context, and we evaluate this method using data from the SemEval lexical substitution task and human annotated data. Second, we address the problem that arises from words with more than one sense, which creates a potential ambiguity in terms of which bits are represented by a particular word. We develop a novel method in which words are the vertices in a graph, synonyms are linked by edges, and the bits assigned to a word are determined by a vertex coding algorithm. This method ensures that each word represents a unique sequence of bits, without cutting out large numbers of synonyms, and thus maintains a reasonable embedding capacity.

1. Introduction

In order to transmit information through an open channel without detection by anyone other than the receiver, a covert channel can be 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 cover medium are manipulated in an unexpected, unconventional, or unforeseen way so that, with steganographic transmission, the encrypted messages can be camouflaged in a seemly innocent medium and sent to the receiver with less chance of being suspected and attacked. Ideally, because the changes to the medium are so subtle, anyone not specifically looking for a hidden message is unlikely to notice the changes (Fridrich 2009).

* 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK. E-mail: Ching-Yun.Chang@cl.cam.ac.uk.
** 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK. E-mail: Stephen.Clark@cl.cam.ac.uk.

Submission received: 18 January 2013; revised version received: 3 June 2013; accepted for publication: 1 August 2013.

doi:10.1162/COLI_a_00176

In this article, we aim at concealing secret information in natural language text by manipulating cover words. The proposed steganography system replaces selected cover words with their synonyms, which is the mechanism used to embed information. In order to ensure the lexical substitutions in the cover text are imperceptible, the system uses the Google *n*-gram corpus (Brants and Franz 2006) for checking the applicability of a synonym in context. In addition, the system assigns codes to acceptable substitutes using a novel vertex coding method in which words are represented as vertices in a graph, synonyms are linked by edges, and the bits assigned to a vertex represent the code of a particular word. Our lexical substitution-based steganography system was previously published in Chang and Clark (2010b), in which the system was evaluated automatically by using data from the English lexical substitution task for SemEval-2007.[1] In this article, we extend the previous work by more closely addressing the practical application of the proposed system. We present results from a new human evaluation of the system's output and a simple computational steganalysis of word-frequency statistics which is a more direct evaluation for linguistic steganography. We also give an extended literature review which will serve as a useful introduction to linguistic steganography for those *Computational Linguistics* readers not familiar with the problem.

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 tablets by writing messages on the wood before applying a wax surface (Herodotus 1987). Another early recorded use of steganography occurred in ancient Greece when 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, in the Russo-Japanese War (1905) microscopic images were hidden in ears, nostrils, or under fingernails (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 postcards (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 documented that more than 100 text files were retrieved from images posted on publicly accessible Web sites.[2] According to the Steganography Analysis and Research Centre,[3] 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

1 http://www.dianamccarthy.co.uk/task10index.html.
2 http://www.justice.gov/opa/documents/062810complaint2.pdf.
3 http://www.sarc-wv.com/.

the least-significant-bit substitution technique. Because the difference between 11111111 and 11111110 in the value for red/green/blue intensity is likely to be undetectable by the human eye, the least-significant-bit can be used to hide information other than color, without being perceptable by a human observer.[4]

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 a 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 innocent 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 object**, and a message with embedded hidden information is called a **stego object**, where **object** stands for "text," "image," "audio," or whatever media is being used. The algorithms that Alice uses for creating the stego object and Bob uses 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 be unsuspicious according 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. Because 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 because 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 and, if necessary, to trace the source of the 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 use 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 or for the purpose of tamperproofing. 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.

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

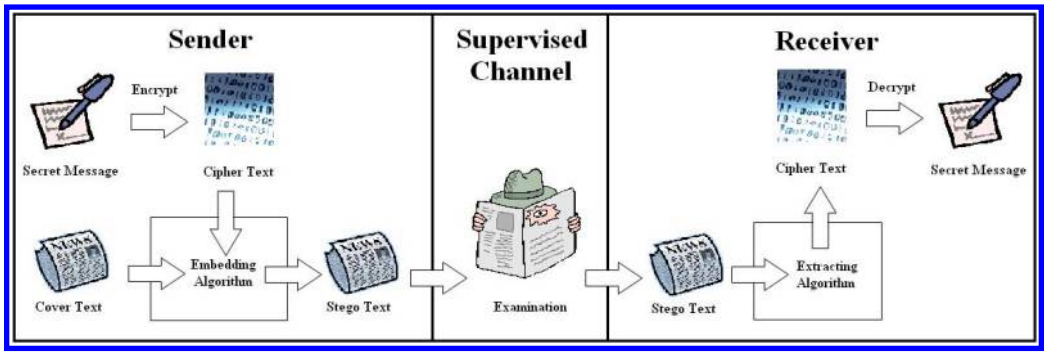


Figure 1
The linguistic steganography framework.

1.2 Linguistic Steganography

A key question for any stegosystem is the choice of cover medium. Given the ubiquitous nature of natural language and the 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 that 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 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*.[5] Next, the stego text passes the observer (human or computer), who does not object to innocuous messages passing between the sender 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

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

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 that are undetectable by an observer. Language has the property that even small local changes to a text (e.g., replacing a word with a word with similar meaning) may result in text that is anomalous at the document level, or anomalous with respect to the state of the world. Hence finding linguistic transformations that can be applied reliably and frequently is a challenging problem for linguistic steganography.

An additional 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. The current state-of-the-art techniques for automatically evaluating the fluency and grammaticality of natural language generation systems are based on techniques for evaluating the output of machine translation systems, such as comparing the *n*-grams in the machine translation output with those in a reference translation (Papineni et al. 2002; Zhang and Clark 2011). Although some computational steganalysis systems have been developed to identify stego text from innocent text using statistical methods, these 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, 2007b; Meral et al. 2007, 2009; Kim 2008, 2009; Chang and Clark 2010a, 2012a, 2012b), where a human assessor was provided with stego text and was asked to rate or improve the naturalness of the stego text.

1.3 Linguistic Stegosystem

Most existing stegosystems consist of three independent modules—linguistic transformation, encoder generation, and text selection—as shown in Figure 2. As explained earlier, 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 imperceptible to an observer. This first step of modification is called linguistic transformation. For example, suppose there is a cover sentence *I like biscuits with a cup of tea*, in which *biscuits* can be replaced by its synonym *cookies* without degrading the naturalness of the original sentence; hence *biscuits* can serve as an information carrier in this example (using synonym substitution as the transformation). After the linguistic transformation stage, there are two alternative sentences *I like biscuits with a cup of tea* and *I like cookies with a cup of tea*. According to the linguistic transformation used in a stegosystem, we can classify existing work into three major categories, which will be discussed in detail

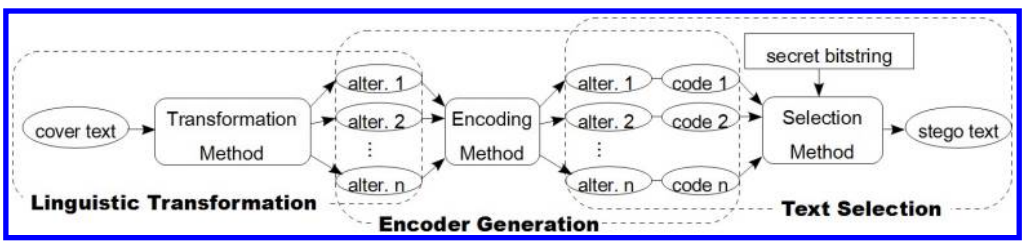


Figure 2
Three modules in a linguistic stegosystem.

in Section 2.1: 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 encoder generation. To continue the previous example, a simple encoding method would first sort the alternative sentences alphabetically and then encode the first alternative, namely, the one containing *biscuits* with bit 0, and the other sentence containing *cookies* with bit 1. The final phase is text selection, which chooses the alternative representing the secret bitstring as the stego text. Let us assume the secret bit is 1; therefore during the text selection phase, *I like cookies with a cup of tea* is chosen as the stego sentence in order to embed the secret. 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.

To recover the secret in the example, the sender and receiver must share the same linguistic transformation and encoder generation modules. The receiver first uses the transformation method to determine that, in the stego sentence, *cookies* can be replaced by its synonym *biscuits*; hence, two alternative sentences are derived. Next, the receiver uses the encoding method to assign codes to the two alternatives. Because the stego sentence *I like cookies with a cup of tea* is encoded by bit 1, the receiver knows that the secret is 1.

The convenient modularity between the linguistic transformation and encoder generation allows a transformation to be combined with different encoding algorithms, although the transformation may put some constraints on what encoding method can be used. For example, suppose we replace the synonym substitution-based transformation in the example with a translation-based transformation that takes a non-English cover sentence as input and outputs two different English translations as the alternatives. Assume the two translations are *I like biscuits with a cup of tea* and *I like cookies with a cup of tea*, and that the second alternative is the stego sentence. Now the receiver has to recover the secret from this stego sentence. However, the receiver does not know the original cover sentence, so it is unlikely that the receiver will be able to obtain the other translations used by the sender to derive the code assigned to the stego sentence. In this case, the translation-based transformation cannot work with block code encoding. A possible solution is to use hash function encoding, as explained in Section 2.2.4.

1.4 Overview

In this article we focus on linguistic steganography rather than watermarking, because we are 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 some practical security issues in the steganography application that we have chosen to ignore or simplify in order to focus on the underlying NLP techniques. For example, we assume the adversary in the espionage scenario is 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, we have ignored the possibility of steganographic attacks (Fridrich 2009), via an active warden who deliberately modifies messages in order to thwart any hidden communication. In addition, for the human evaluation of the security level of our stegosystem, we evaluate the naturalness of generated stego sentences. In other words, we do not investigate the document-level coherence of stego text because this requires sophisticated knowledge of natural language semantics and pragmatics which we consider to be outside the

scope of this work. Therefore, it is possible that a totally natural stego sentence can still raise suspicion if it contains words which stand out as being completely different from the rest of the paragraph. For the computational steganalysis, we use a simple statistical analysis proposed in Meng et al. (2010), which compares the frequency of high-frequency words in a stego text and in its original text.

The main objective of this article is to explore the applicability of lexical substitution to steganography. Lexical substitution 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. Another advantage of this transformation is that many languages are profuse in synonyms so there is a rich source of information carriers in a cover text. In this work we 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 and *n*-gram corpora.

There are two practical difficulties associated with hiding bits using lexical substitution. The first is that words can have more than one sense. In terms of WordNet (Fellbaum 1998), which is the electronic dictionary we use, words can appear in more than one synonym set (synset). 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. Our solution to this problem is a novel vertex coding 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. Our solution to this problem is to perform a contextual check that utilizes the Google *n*-gram corpus (Brants and Franz 2006). We evaluate the substitution checker using the data from the English lexical substitution task for SemEval-2007 and a human judgment corpus created specifically for this work.

For the proposed lexical substitution checker, the higher quality the passed substitutions are, the less suspicious the stego text may be. In addition, the more substitutions that pass the check, the more information carriers the stegosystem can use. Hence the evaluation of the proposed substitution checker can be seen as an indirect evaluation of the proposed stegosystem. For this reason, the performance of the proposed substitution checker is evaluated in terms of precision and recall in our automatic evaluation. Precision is the percentage of substitutions judged acceptable by the checker that are also in the gold standard of the SemEval-2007 English lexical substitution task; recall is the percentage of substitutions in the gold standard that 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. Apart from precision and recall evaluations, we also asked human judges to evaluate the naturalness of the substitutions that pass the proposed checker, which is a more direct evaluation of the imperceptibility of the steganography application.

A significant contribution of this article is to advertise the linguistic steganography problem to the NLP community. The requirement that any linguistic transformations maintain the grammaticality and meaning of the cover text makes the problem a strong test for existing NLP technology. In addition, the proposed substitution checker for

certifying sentence naturalness potentially benefits not only the steganography application, but also other NLP applications that require a measure of how natural a word is in a particular context. Another contribution of the work is the evaluation of the proposed stegosystems. The results suggest that it is possible to develop a practical linguistic steganography system based on lexical substitution with current NLP techniques.

The rest of this article is organized as follows. Section 2 reviews the current state of the art in linguistic steganography and the various linguistic transformations that have been used in existing stegosystems. This is a substantial literature review designed to introduce the problem to the *Computational Linguistics* reader. In Section 3, we describe our lexical substitution-based stegosystem, along with the method for checking substitution quality together with an empirical evaluation. In Section 4 we propose a novel vertex coding algorithm to solve the decoding ambiguity problem and demonstrate the proposed stegosystem using an example.

2. Background

This section reviews existing linguistic stegosystems. Under our interpretation of the term linguistic steganography, we are only concerned with stegosystems that make changes that are linguistic in nature, rather than operating on superficial properties of the text, for example, the amount of white space between words (Por, Fong, and Delina 2008), font colors (Khairullah 2009), or relying on specific file formats, such as ASCII or HTML (Bennett 2004; Shahreza 2006).

2.1 Linguistic Transformations

In the following, we describe 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 Transformations. 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). These can be used to perform lexical substitution. 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

Table 1
Synsets of the word *marry* in WordNet 3.1.

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

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 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 introducing much semantic difference because *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 is word-category disambiguation, which marks up a word with a particular 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, Satta, and Joshi 2007; Spoustová et al. 2009; Søgaard 2010) and are widely used in lexical substitution-based stegosystems (Chapman, Davida, and Rennhard 2001; Bolshakov 2004; Taskiran, Topkara, and Delp 2006; Topkara, Topkara, and Atallah 2006b; Topkara et al. 2006; 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" rather than "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 exactly the same synsets, and only uses these words for substitution. For example, *marry* and *wed* in Table 1 are interchangeable words because they are always synonyms even under different meanings. Any words that are not interchangeable are discarded and not available for carrying information. Winstein 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 replaced *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, Topkara, and Delp (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 Section 3, we describe how our 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 our earlier work (Chang and Clark 2010a) we developed a stegosystem exploiting a paraphrase dictionary (Callison-Burch 2008) to find potential information carriers, and used the Google n-gram corpus

Table 2
Some common syntactic transformations in English.

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.

and a combinatory categorial grammar (CCG) parser (Clark and Curran 2007) to certify the paraphrasing grammaticality.

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 lists some of the common syntactic transformations in English.[6]

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 performing lexical substitution directly on the text, the modifications are performed on the syntactic parse tree of a cover sentence. Murphy (2001), Liu, Sun, and Wu (2005), Topkara, Topkara, and Atallah (2006a), 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 these 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 uses a probabilistic grammar-based model to structure the stego text. Because 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. Compared to 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

6 The categories of transformations are adopted from Topkara, Taskiran, and Delp (2005).

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

Figure 3
Parts of the ontological semantics for *Afghanistan*.

```
assault–|–agent–nation–"United States"
        |–theme–nation–"Afghanistan"
```
(a)

```
assault–|–agent–nation–"United States"
        |–theme–nation–"Afghanistan"–|–has-representative–politician–"Mullah Mohammad Omar"
```
(b)

Figure 4
An example of the TMR tree modification taken from Atallah et al. (2002).

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 these 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 the sentences in different voices, such as active and passive, 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 ontological semantic 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 3, taken from Atallah et al. (2002), shows parts of the ontological semantics for *Afghanistan* that are structured in a tree-like hierarchy.[7] An ontology provides concepts that are used to define propositions in TMR. The TMR of a natural language expression can show information such as clause relationship, author attitude, and topic composition. It is constructed through mapping lexical items and events that are referred in the expression to their ontology concepts. Figure 4(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 4(b), yielding the alternative sentence *the United States are attacking Afghanistan, which is ruled by Mullah Mohammed Omar*. Vybornova and Macq (2007) also exploited the linguistic

7 Afghanistan was ruled by Mullah Mohammed Omar at the time of 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 not 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 utilized multiple translation systems to output alternatives for a cover sentence. Because Grothoff et al. (2005) and Stutsman et al. (2006) used multiple machine translation systems to generate alternative translations, which leads to a stego text containing a mixture of translations generated from different systems and each stego sentence may have different statistical distribution of features (e.g., percentage of high-frequency words), a simple comparison of the statistical distribution of features obtained from a normal text and from a stego text might be able 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. Because translations are from one system, each of them is more similar to the rest than that derived from another translation system.

Another of our papers (Chang and Clark 2012b) proposed a word-ordering-based stegosystem, where the word-ordering technique can be seen as a "monolingual translation" that translates a cover sentence into different permutations. Because not all the sentence permutations generated by a word-ordering system are grammatical and semantically meaningful, we developed a maximum entropy classifier to distinguish natural word orders from awkward ones.

Another possible semantic transformation for linguistic steganography is sentence compression (Dorr, Zajic, and Schwartz 2003; Cohn and Lapata 2008; Zhu, Bernhard, and Gurevych 2010). Different compressed versions of the original text provide various alternatives for a stegosystem. We developed a compression-based stegosystem that generates alternatives for a cover sentence by removing unnecessary adjectives in noun phrases (Chang and Clark 2012a). For example, *he spent only his own money* and *he spent only his money* almost express the same meaning. In order to certify the deletion grammaticality, we only accept a deletion that does not change the CCG categories of the words in the rest of the sentence. In addition, we propose two methods to determine whether the adjective in a noun phrase is necessary to the context. The first method uses the Google n-gram corpus, and the second method, which performs better, trains a support vector machine model that combines n-gram statistics, lexical association measures, entropy-based measures, and an n-gram divergence statistic.

2.2 Encoding Methods

In the previous sections we have explained different transformation methods for generating alternatives for an input text. This procedure is seen as the linguistic transformation module in Figure 2. After deriving alternatives for a cover text, the encoder generation module maps each alternative to a code that can be used to represent a secret bitstring. In this section, we introduce four encoding methods that assign bitstrings to the alternative candidates and have been used in existing stegosystems. In order to demonstrate each encoding method, we 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 *labor*, *task*, or *undertaking*. Note that, as mentioned earlier, 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. Because the synonym set {*labor*, *project*, *task*, *undertaking*} has cardinality $n = 4$, the block code method can use either one-bit or 2-bit codes to encode the words as shown in Figure 5. When one-bit codes are used, both *labor* and *task* represent code *0*, and both *project* and *undertaking* represent code *1*; when two-bit codes are used, the four words are assigned different codes *00*, *01*, *10*, and *11*. The advantage of using one-bit codes is that the cover word *project* needs to be replaced with its synonym only 50% of the time, whereas the two-bit scheme has a 75% chance of modifying the cover word, assuming the secret is a random bitstring. However, one-bit codes embed less information. 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. For example, if the block code method is applied to Table 1, *wed*, *tie*, and *splice* would have the same probability of replacing *marry* in the cover sentence *The minister will marry us on Sunday*. However, of these three alternatives only *wed* is allowed in this context; hence choosing *tie* or *splice* may arouse suspicion in others.

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 $a_n b_{n-1} b_{n-2}...b_0 + a_{n-1} b_{n-2} b_{n-3}...b_0 + ... + a_1 b_0 + a_0$, and any number can be uniquely expressed in mixed-radix form (Soderstrand et al. 1986).

Figure 6 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 {*labor*, *project*, *task*, *undertaking*} are encoded with *0*, *1*, *2*, and *3* with base 4. Therefore, the combinations of the substitutions yield the two-digit mixed-radix numbers from $0_2 0_4$ to $1_2 3_4$, which are equal to the decimal numbers 0 to 7. Assume the secret bitstring

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

We the charitable .

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

Figure 5
An example of the block code method.

	Code	Word		Code	Word	
We	0_2	*complete*	the charitable	0_4	*labor*	.
	1_2	*finish*		1_4	*project*	
				2_4	*task*	
				3_4	*undertaking*	

Figure 6
An example of the mixed-radix number method.

to be embedded is *110*, which can be seen as the binary number for six. Because *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 have security implications. To solve this issue, one can utilize variable-length code methods described in the next section.

2.2.3 Huffman Code Method. Figure 7 demonstrates the use of variable-length codes, in the form of the Huffman code (Huffman 1952), for encoding words in a synset. Assuming there is a utility score 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 have shorter codes whereas words with lower utility get longer codes. Thus, words frequently used by native speakers are more likely to be chosen by the stegosystem (assuming utility corresponds to frequency). In addition, when making a low-utility choice, the tree ensures that maximal (bitrate) benefit will be derived from that choice. The process shown in Figure 8 begins with leaf nodes each containing a word along with its associated probability. 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. 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. 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 words in the same synset and is encoded with the shortest code. Thus, it is more likely to match the secret bitstring.

One existing stegosystem that uses Huffman code encoding is proposed by Grothoff et al. (2005). Their system takes different translations of a cover sentence as alternatives, each of which is assigned a probability that represents the quality of the translation. Then a Huffman tree is built for the translations based on the probabilities, where poorer translations are at the bottom of the tree (with lower probabilities), and quality

	Code	Word	Prob.		Code	Word	Prob.	
We	0	*complete*	0.77	the charitable	110	*labor*	0.05	.
	1	*finish*	0.23		0	*project*	0.69	
					10	*task*	0.25	
					111	*undertaking*	0.01	

Figure 7
An example of the Huffman code method.

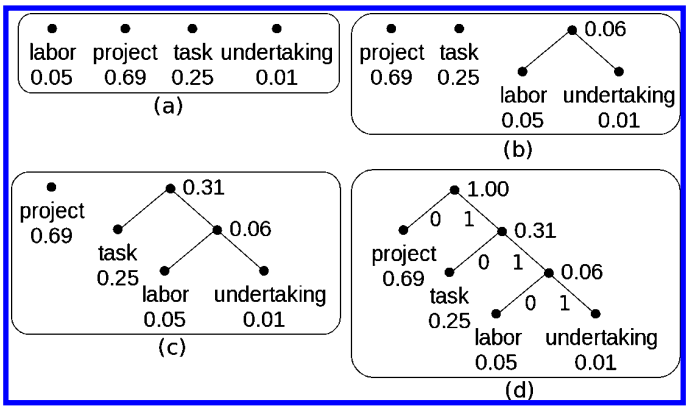


Figure 8
The process of constructing a Huffman tree.

translations are higher in the tree. In other words, poor translations have longer codes while better translations have shorter codes. This ensures that quality translations appear more often, and when a poorer translation (and thus potentially more perceptible sentence) appears, it transmits a maximal number of bits.

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 9(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

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 9
An example of the Wayner (1995) mimicry method.

secret bitstring. The procedure is repeated until a grammatical message is generated. In the embedding example given in Figure 9(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.

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 represents the secret bits to the receiver, it is unlikely that the receiver can retrieve the four best machine translations without knowing the original cover sentence. Thus, the secret recovery fails. For this reason, Stutsman et al. (2006), Meng et al. (2011), Venugopal et al. (2011), and Chang and Clark (2012b) 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. Stutsman et al. (2006) proposed a hash function encoding scheme which uses the first h bits of a translation hash bitstring as the header bits and the next b bits as the code represented by the translation, where h is shared between the sender and the receiver, and b is the integer represented by the header bits. For example, assume $h = 2$; a hash bitstring "1011..." has header bits 10 to indicate a $10_{(2)}$-bit code is carried by this translation, and the two-bits are 11. Among all possible translations of a cover sentence, the one with the combination of header and information-carrying bits for the given h representing the next b bits of the message is chosen as the stego sentence.

2.3 Stegosystem Evaluations

So far we 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 represents 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 a stegosystem.

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, Topkara, and Atallah

(2006a) and Topkara et al. (2006) used machine translation evaluation metrics BLEU (Papineni et al. 2002) and NIST (Doddington 2002), automatically measuring how close a stego sentence is to the original. Topkara, Topkara, and Atallah (2006a) 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 judgments. Meral et al. (2007, 2009) and Kim (2008, 2009) asked participants to edit stego text for improving intelligibility and style. The fewer edit-hits a transformed text received, the higher the reported security level. Murphy and Vogel (2007a, 2007b) first asked subjects to rate the acceptability (in terms of plausibility, grammaticality, and style) of the stego sentences on a seven-point scale. Then participants were provided with the originals and asked to judge to what extent meaning was preserved, also on a seven-point scale. In Chang and Clark (2010a) we asked participants to judge whether a paraphrased sentence is grammatical and whether the paraphrasing retains the meaning of the original. In Chang and Clark (2012a) we asked participants to annotate the naturalness of the resulting sentences after adjective deletions; and in Chang and Clark (2012b) we asked participants to rate the naturalness of sentence permutations on a four-point scale. For the work presented in this article, we also use human judgments to evaluate the proposed stegosystem, as this is close to the linguistic steganography scenario where we assume the adversary is a human acting passively.

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 of hidden message per bit transmitted or per language unit (e.g., per word or per sentence). Payload measurements can be theoretical or empirical. The theoretical payload measurement only depends on an encoding method and is independent of the quality of a stego text; 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, Taskiran, and Delp (2005) and Topkara, Topkara, and Atallah (2006b) achieved an average embedding payload of 0.67 bits per sentence, despite the large number of synonyms in English. In Chang and Clark (2012a) we showed that the payload upper bound of using the adjective deletion technique is around 0.4 bits per sentence if a deletion represents a secret bit. 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, Topkara, and Atallah (2006a) 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. Because the ontological semantic transformation is currently impractical, the empirical payload is not available for this transformation type. 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 of hidden message for every 100 bits of data transmitted.

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 storybook than in a car repair manual because there are usually many terminologies in a manual which cannot be changed or even cannot be found in a standard dictionary (assuming the system does not happen to have a detailed ontology of car parts). To the best of our knowledge, there is no study on the practical issue of using different types of cover text for the steganography application.

3. Lexical Substitution

In the following sections we introduce our linguistic stegosystem based on lexical substitution. In the original work on linguistic steganography, 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 5. 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 that 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 1. Figure 10 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 marry us on Sunday*. Note that we 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 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 two-bit codes. As can be seen in Figure 10, *marry* is encoded by two different codewords and thus the secret bitstring cannot be reliably recovered, because 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., they always appear together in the

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

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

Figure 10
An example of decoding ambiguity using lexical substitution.

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. As mentioned previously, 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.

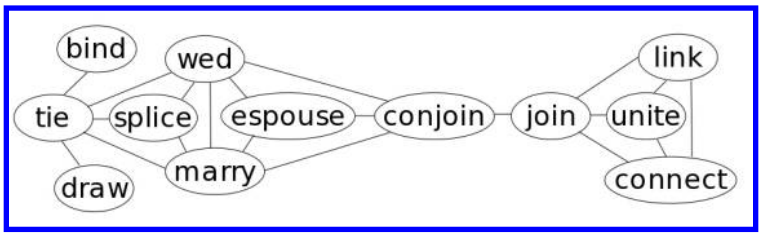
Another synonym substitution-based stegosystem was proposed by Bolshakov (2004), who applies **transitive closure** to overlapping synsets to avoid the decoding ambiguity. Applying transitive closure leads to a merger of all the overlapping synsets into one set which is then seen as the synset of a target word. Consider the overlapping synsets in Figure 10 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, our 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 both in the synonym transitive closure chain as well as acceptable to the context. More details of the proposed substitution checker will be described later. We then construct a synonym graph that 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, we use a novel vertex coding method inspired by vertex coloring to assign codes to every word in the graph.

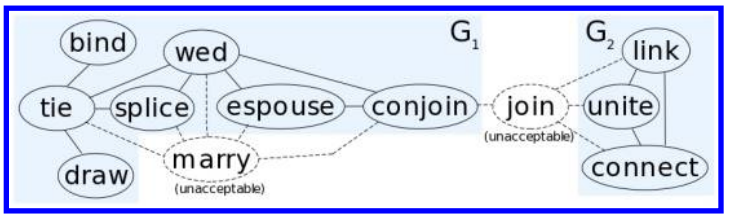
A crucial difference from Bolshakov's method is that in our approach the sender only considers words that are synonymous with the cover word as alternatives, even though the other words in the synonym graph can also fit into the context. The reason for also including non-synonymous words during the encoding is because the receiver does not know the cover word and, therefore, we 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 so that the sender knows the cover word and the receiver knows the stego word.

Figure 11(a) shows a synonym graph constructed from a synonym transitive closure chain that 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; in our method, however, we 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 we have not applied the substitution check in this example.

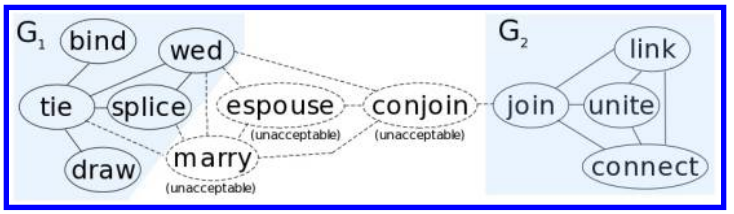
Now let us apply the substitution check to words in the synonym transitive closure chain, and suppose *join* and *marry* do not pass the check. Figure 11(b) shows the two disconnected synonym graphs G_1 and G_2 derived from the checked pool. The two



(a) An unchecked synonym graph



(b) Synonym graphs derived after substitution checking



(c) Another example of checked synonym graphs

Figure 11
Synonym graphs with and without the substitution check.

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. Because *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 those 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. Because 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.

Because 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 11(c) shows the corresponding synonym graphs of the remaining words. In this case, the applicable alternatives are either *wed* or *join* because 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.

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 must not only work with the proposed linguistic stegosystem, but can also be integrated into other synonym substitution-based applications to certify the transformation quality. The following sections are organized so that the basic substitution checker using the Google *n*-gram corpus (Brants and Franz 2006) is described first. Then we 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. We also perform a more direct evaluation of the imperceptibility of the steganography application by asking human judges to evaluate the naturalness of sentences. After explaining the linguistic transformation module in our stegosystem, we proceed with the encoder generation module and present the vertex coding method. Finally, we use an example to demonstrate the complete stegosystem.

3.1.1 n-*Gram Count Method (NGM).* The basic checking method, referred to as NGM, utilizes the Google *n*-gram corpus to calculate a substitution score for a candidate word in context based on Bergsma, Lin, and Goebel (2009). The Google *n*-gram corpus was collected by Google Research for statistical language modeling, and has been used for many tasks such as spelling correction (Carlson, Mitchell, and Fette 2008; Islam and Inkpen 2009), multi-word expression classification (Kummerfeld and Curran 2008), and lexical disambiguation (Bergsma, Lin, and Goebel 2009). It contains frequency counts for *n*-grams from uni-grams through to 5-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 5-grams around the word to be tested and uses the Minnen, Carroll, and Pearce (2001) tools 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 *decolor* will be corrected as *decoloring* because the original word is in the progressive tense.

After extracting contextual bi- to 5-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. We define a count function:

$$Count(w) = \sum_{n=2}^{5} log(f_n),$$

where *log(0)* is defined as zero. If *Count(w)=0*, we 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. We also need to use the most likely word, rather than the original cover word, because 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 with a high score is more suitable for the context, and we apply a threshold so that words having a score lower than the threshold are discarded.

Consider as an example the calculation of the substitution score for the candidate word *clever* as a possible replacement for the word *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, as shown in Figure 12. The derived f_n values can then be used to calculate *Count(clever)*, which is 27.5 in this example. Suppose the threshold is 0.9, and the max_{count} is 30 from the synonym transitive closure chain. The substitution score is as follows:

$$Score_{NGM}(clever) = \frac{Count(clever)}{max_{count}} = \frac{27.5}{30} = 0.92$$

which is greater than the threshold (0.9), 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 bigrams *the even* and *even of* have high frequency counts compared with those of the 4-grams and 5-grams. As a way of overcoming this problem, we consider the

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	

Figure 12
n-grams and their frequency counts for calculating $Score_{NGM}(clever)$.

	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

Figure 13
n-gram frequency distributions of *bright* and *clever* for calculating the contextual divergence.

n-gram distributional similarity between a most likely word and a candidate substitute in context using α-skew divergence as explained in the next section. We 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 our case the *n*-gram count distribution of the most likely word, and Q represents a model, in our 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 allows us to avoid the problem of zero probabilities, and in our method we use $\alpha = 0.99$. 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.

We use the following example to demonstrate how to calculate the contextual divergence between two words. Assume the most likely word is *bright* and a substitute to be considered is *clever*. First we need to derive the *n*-gram frequency distributions of both words. We divide each *n*-gram frequency by the total frequency to get C_{ni}, as shown in Figure 13, where i means the ith *n*-gram (e.g., C_{32} is the second trigram). For a word, C_{ni} should sum up to 1 (over all n, i). Then we can calculate the α-skew divergence of these two distributions:

$$S_\alpha(clever,bright) = \sum_n \sum_i C_{ni}^{bright} \cdot log(\frac{C_{ni}^{bright}}{\alpha C_{ni}^{clever} + (1 - \alpha) C_{ni}^{bright}}) = 0.014$$

Similar to the NGM method, we 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, suppose $max_{divergence}$ is 0.15 and therefore we can derive the contextual divergence-based substitution score:

$$Score_{DVG}(clever) = 1 - \frac{0.014}{0.15} = 0.91$$

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

$$Score_{NGM\_DVG}(w) = \lambda \cdot Score_{NGM}(w) + (1 - \lambda) \cdot Score_{DVG}(w)$$

where $0 \leq \lambda \leq 1$. The value of λ determines the relative weights of $Score_{NGM}(w)$ and $Score_{DVG}(w)$.

3.2 Ranking Task Evaluation

Both NGM and NGM_DVG assign a score to a word according to the context and the most likely word in the group of alternatives. In order to evaluate the performance of the proposed scoring methods, we apply our approaches to a ranking task that requires a system to rank a list of substitute 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 evaluation measure used is Generalized Average Precision (Kishida 2005). In this section we first describe the gold standard data used in this evaluation and then provide the results. We compare our results with three other models developed by Erk and Padó (2010), Dinu and Lapata (2010), and Ó Séaghdha and Korhonen (2011), all of which are designed for measuring word meaning similarity in context. Note that our substitution checkers do not aim at modeling word similarity, and therefore the result comparison is just trying to show that our substitution checkers are competitive. Later, we will evaluate the proposed checkers with the human annotated data and see whether our methods would be practical for linguistic steganography.

3.2.1 Data. For this evaluation, we use the SemEval-2007 lexical substitution data set as the gold standard. The original purpose of the data set was 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 the 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 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), Dinu and Lapata (2010) and Ó Séaghdha and Korhonen (2011), we pool together the positive substitutes for each target word, considering all contexts, and rank the substitutes using our scoring methods. For instance, assume in the gold standard there are only two sentences containing the target word *bright* as shown in Table 3. We merge all the substitutes of *bright* given by the annotators and derive a large candidate pool {*intelligent*, *clever*, *colorful*, *brilliant*, *gleam*, *luminous*}. We expect *intelligent* and *clever* to be ranked at the top of the list for the first sentence, with *colorful*, *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, we use the Generalized 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 our 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, we decided a λ value of 0.6 for the NGM_DVG method. We then applied the proposed NGM and NGM_DVG methods to rank pooled substitutes for each sentence in the test data. Table 4 summarizes the performances of our approaches, where mean GAP values are reported on the whole test data as well as for different POSs. We 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

Table 3
Two sentences in the SemEval-2007 lexical substitution gold standard.

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.	colorful(2), brilliant(1), gleam(1), luminous(1)

Table 4
GAP values (%) of the ranking task evaluation.

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
Ó Séaghdha and Korhonen (2011)	49.5	50.7	45.1	48.8	55.9

data. We then compare our results with those achieved by Erk and Padó (2010), Dinu and Lapata (2010), and Ó Séaghdha and Korhonen (2011). Erk and Padó 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 exemplar sentences most similar to it. Dinu and Lapata proposed a vector-space model that models the meaning of a word as a probability distribution over a set of latent senses. Ó Séaghdha and Korhonen also use probabilistic latent variable models to describe patterns of syntactic interaction. The best mean GAP values reported by Erk and Padó, Dinu and Lapata, and Ó Séaghdha and Korhonen are 38.6%, 42.9%, and 49.5% on the test data, respectively.

3.3 Classification Task Evaluation

Although the ranking task evaluation gives some indication of how reliable the proposed scoring methods are, for the steganography application we require a system that can correctly distinguish acceptable substitutes from unacceptable ones. Thus, we conduct a classification task evaluation which 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, we first describe the data and then explain the experimental setup and the evaluation results.

3.3.1 Data. We use the sentences in the gold standard of the SemEval-2007 lexical substitution task as the cover text in our experiments so that the substitutes provided by the annotators can be the positive data. Because we only take into consideration the single word substitutions, multi-word substitutes are removed from the positive data. Moreover, we use WordNet as the source of providing candidate substitutes in our stegosystem, so if a human-provided substitute does not appear in any synsets of its target word in WordNet, there is no chance for our stegosystem to replace the target word with the substitute; therefore, the substitute can be eliminated. Table 5 presents the statistics of the positive data for our experiments.

In addition to the positive data, we also need some negative data to test whether our methods have the ability to filter out bad substitutions. We extract the negative data for our 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

Table 5
Statistics of experimental data.

	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

into consideration. We 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.

Let us use an example to demonstrate our automatic negative data collection. In this example, we need to generate bad substitutions for a cover word *remainder* in the sentence *if we divide any number by 4, we would get 1 or 2 or 3, as the remainders*, given that the annotator-provided positives are *leftover* and *residual*. Table 6 lists the synsets of *remainder* found in WordNet 3.1. Because the synset {*remainder*, *balance*, *residual*, *residue*, *residuum*, *rest*} contains one of the positives whereas the other synsets do not, this synset is selected for our negative data collection. We 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 our checking methods can pick out bad substitutions 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 selected 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 the second author of this article. As can be seen from the annotation results shown in Table 7, most of the instances are true negatives, and only a few cases are incorrectly chosen as false negatives. Because 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 our experimental data. Also, it is

Table 6
Synsets of *remainder* in WordNet 3.1.

remainder (noun)
gloss: something left after other parts have been taken away synset: remainder, balance, residual, residue, residuum, rest
gloss: the part of the dividend that is left over when the dividend is not evenly divisible by the divisor synset: remainder
gloss: the number that remains after subtraction; the number that when added to the subtrahend gives the minuend synset: remainder, difference
gloss: a piece of cloth that is left over after the rest has been used or sold synset: end, remainder, remnant, oddment

Table 7
Annotation results for negative data.

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

more harmless to rule out a permissible substitution than to include an inappropriate replacement for a stegosystem in terms of its security. Table 5 gives the statistics of the automatically collected negative data for our experiments.

3.3.2 Experiments and Results. We evaluate the classification performance 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. 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, rejecting an acceptable substitute does not damage the quality of stego text. However, it will lower the payload capacity so more stego text transmission is needed in order to send the secret message, which may raise a security concern.

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, five-fold cross validation was used for the experiments. For each fold, 80% of the data is used to find the threshold value which maximizes the accuracy, and that threshold is then applied to the remaining 20% to get the final result.

We first test whether the proposed methods would benefit from using only longer n-grams. We 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 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 bigram to five-gram frequency counts performs the best across different POS and, therefore, is adopted in the NGM system and the NGM_DVG system. Next, we try weighting different sized n-grams in the proposed methods, as have Bergsma, Lin, and Goebel (2009) in related work. According to the preliminary experiments we conducted and the conclusion given by Bergsma, Lin, and Goebel, such a method does not do much better than the simple method using uniform weights for different sized n-grams.

Table 8 gives the results for the two checking methods and the average threshold values over the five folds. In addition, for each POS, a simple baseline is derived by always saying a substitute is acceptable. From the table we can see that both the NGM and the NGM_DVG systems have higher precision than the baseline, which performs well in terms of embedding capacity (100% recall) but at the expense of a lower security level (lower precision). However, in contrast to the results of the ranking task evaluation, the

Table 8
Performance of the NGM and NGM_DVG systems on the classification task.

	NGM				NGM_DVG				Baseline	
POS	Acc %	Pre %	Rec %	Thr	Acc %	Pre %	Rec %	Thr	Acc, Pre %	Rec %
noun	70.2	70.0	80.2	0.58	68.1	66.5	67.3	0.70	55.0	100
verb	68.1	69.7	79.5	0.56	64.8	65.7	66.7	0.70	58.0	100
adj	72.5	72.7	85.7	0.48	70.2	68.8	77.7	0.63	59.2	100
adv	73.7	76.4	80.1	0.54	68.0	66.4	75.9	0.63	58.9	100

NGM system slightly outperforms the NGM_DVG system. Because imperceptibility is an important issue for steganography, we would prefer a system with a higher precision value. Thus we adopt the NGM method as the linguistic transformation checker in our lexical substitution-based stegosystem.

In addition, we are interested in the effect of the threshold value on the performance of the NGM method. Figure 14 shows the precision and recall values with respect to different thresholds for each POS. From the charts we 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 we have presented the performance of our checking methods using two different automatic evaluations, the ranking task and the classification task. From the ranking task evaluation we can see that the *n*-gram distributional similarity does have the ability to further eliminate some bad substitutes after applying the basic *n*-gram

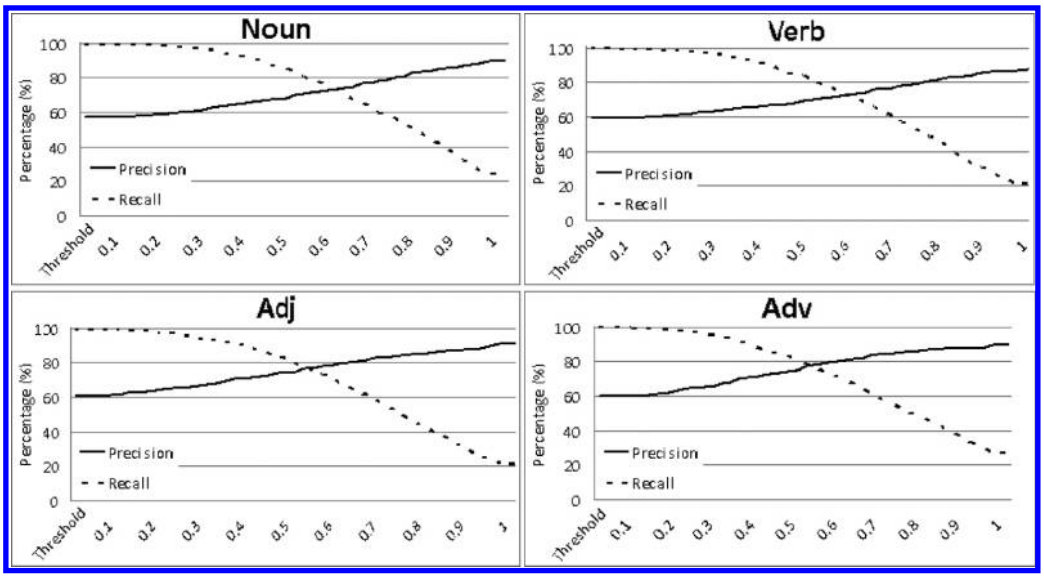


Figure 14
The performance of the NGM method under various thresholds.

method. However, when facing the classification task, which is more related to the steganography application, we 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 we can conclude that the more information we include in the checking method (i.e., using all counts from bi- to five-grams) the better the performance. This is similar to the conclusion given by Bergsma, Lin, and Goebel (2009).

3.4 Human Evaluation

We 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. Therefore, apart from the automatic evaluations, we conducted a more direct evaluation of the imperceptibility for the steganography application by asking human judges to evaluate the naturalness of sentences. In the following sections, we explain the evaluation data first and then describe the evaluation setup and results.

3.4.1 Data. We collected a total of 60 sentences from Robert Peston's BBC blog.[8] For each noun, verb, adjective, and adverb in a sentence, we 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.

We 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 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 because both of them receive the same number of substitutions. Table 9 shows lexical substituted sentences generated by our method and by the random method. We can see that our system replaces four words (in boldface) in the original sentence so the same number of words (in boldface) 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 our system. We refer to an original sentence as COVER, a version generated by our method as SYSTEM, and a version modified by the random method as RANDOM.

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 10. In this table, each row represents a set of annotation sentences for a group of judges, and we 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, we describe the aim of the annotation as shown in Figure A1 in the Appendix. Subjects are asked to rate the naturalness of each sentence

8 http://www.bbc.co.uk/news/correspondents/robertpeston/.

Table 9
Different versions of a cover sentence.

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-aggrandizing** 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 10
Latin square design with three groups of judges.

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

on a scale from 1 to 4 with score 1 meaning *Poor English* and score 4 meaning *Perfect English*. Each judgment score is explained followed by an example sentence. Figure A2 in the Appendix shows a screen capture of an annotation example presented to a subject. It is worth mentioning that we do not ask judges to spot changes in a text that has been run through our system because a spotted change does not necessarily imply that a sentence is unnatural; a spotted change might just be the result of the word preferences and style of an individual judge.

The annotation results show that our judges gave an average score of 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. We measure the significance level of our 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 two information carriers per sentence and each information carrier guarantees to represent at least one bit. These results show that our stegosystem achieves better payload capacity than existing lexical substitution-based stegosystems which have achieved 0.67 bits per sentence (Topkara, Taskiran, and Delp 2005; Topkara, Topkara, and Atallah 2006b). In addition, in terms of security, the results suggest that, with the proposed checking method, the quality of stego sentences is improved compared with the random substitution baseline.

Table 11
The average number of words for each word frequency rank (rank 1 is the lowest frequency and rank 15 is the highest frequency).

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cover	961.1	56.7	35.5	29.3	36.7	45.5	55.4	49.2	48.4	39.6	31.2	23.9	19.8	14.5	239.2
stego	973.8	56.2	35.2	29.2	36.3	46.0	55.7	49.2	47.9	40.3	30.6	23.6	20.6	14.7	238.7

3.5 Computational Analysis of Word-Frequency Distribution

In the previous sections we evaluated our system using human judges and now we test the system using a simple statistical analysis that compares the number of high-frequency words in stego text with that in cover text. We followed the methodology used by Meng et al. (2010), who show that there are fewer high-frequency words in translation-based stego text than in normal text. In the following sections we first describe the evaluation data and then give the evaluation setup and results.

3.5.1 Data. We randomly collected 1,000 pieces of text from 1,000 sections of the British National Corpus[9] as the cover text, each of which is about 20k bytes. Then for each piece of cover text, the corresponding stego text was generated using the same stegosystem setting as that used to generate the human evaluation data described in Section 3.4.1. This results in a total of 152,268 cover words being replaced with synonyms in the 1,000 pieces of stego text. As mentioned in Section 2.3, one of the standard payload measurements is to calculate the number of hidden bits per bit transmitted. From this data, we can calculate the lower bound embedding rate of our stegosystem as 0.09 bits of hidden message per every 100 bits of data transmitted if one substitution only embeds one bit.

3.5.2 Evaluation Setup and Results. In order to calculate word frequency counts, words in both the cover text and the stego text are first lemmatized using the Minnen, Carroll, and Pearce (2001) tools. In each piece of text, the obtained word frequency count is then mapped to a rank r between 1 and 15, where rank 1 corresponds to low frequency and 15 corresponds to high frequency. The mapping is defined as:

$$r = \left\lfloor \frac{frequency - min}{\frac{max - min}{15}} \right\rfloor + 1$$

where *max* and *min* is the maximum and minimum word frequency counts in the text, respectively, and the rank of *max* is 15. Next, we calculate the number of words for each frequency rank in each text. Finally, we calculate the average number of words for each frequency rank for both the cover text and the stego text as shown in Table 11, where the average number of words has been multiplied by r^2 like the results reported by Meng et al. (2010).

From Table 11 we can see the two vectors are very close, and the only apparent difference is that in the stego text there are more low-frequency words than in the cover text. However, unlike the conclusion derived by Meng et al. (2010), our results do not

9 http://www.natcorp.ox.ac.uk/docs/URG/.

show a substantial difference in the frequency of high-frequency words between the cover text and the stego text. A possible explanation is that a translation-based stegosystem may combine translations output from different machine translation systems, hence the usage of frequent words may not be consistent, whereas our stegosystem uses the same substitution checker for each synonym replacement so there is a certain consistency achieved in stego text.

After giving both the results of human and computational evaluations in the previous sections, now we would like to show an example of what a typical stego text will look like. The following paragraphs are generated by the proposed NGM method with the substitution score threshold equal to 0.9, where 24 words have been substituted:

> *The whistleblower, who yesterday gave me the entire recording, told me that the Telegraph's deletion of these sections about Mr Murdoch was a commercial decision, prompted by the fact that the Telegraph - like Mr. Cable - would rather News Corporation does not end up as 100% owner of BskyB.*
>
> *I of course set this to the Telegraph. And quite late in the day, at 19:19 last night to be accurate, the Telegraph's external media adviser sent me a statement attributed to an unidentified "spokesman for the Daily Telegraph." The statement reads:*
>
> *"It is complete nonsense to suggest that the Daily Telegraph did not publish comments from Vince Cable on the Rupert Murdoch takeover of BSkyB for commercial reasons. It was an editorial decision to focus this morning on Cable's comments on the Coalition because they cost of wider interest to our readers."*
>
> *Well, some would say that was a somewhat eccentric editorial decision for an editor, Tony Gallagher, widely regarded as one of the sharpest in the business. I rang Mr. Gallagher to discuss this, but he directed me to the Telegraph's national PR spokesperson.*
>
> *Also, you may have found that the Telegraph has not even put out any clear and unequivocal statement that it was ever planning to publish Mr Cable's remarks about Mr Murdoch (though it has now released them, after they were set out by the BBC).*
>
> *Maybe I am being a bit naive and ridiculous to think any of this matters. Maybe most of you believe that what we do as reporters is so plain and constantly subject to commercial interference that there is no special benefit to be gained from asking the Telegraph to explain itself in this case.*
>
> *But really that's not been my experience in 27 years as a hack. And I still think the question of what news organisations put into the public domain, and how they do it, matters.*

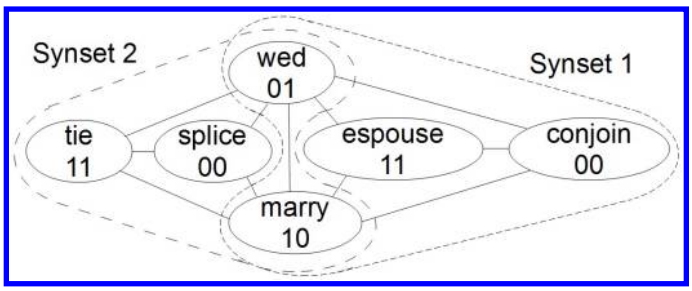
Readers can consult Appendix B for those 24 changes made by the proposed NGM method.

4. Vertex Coding Method

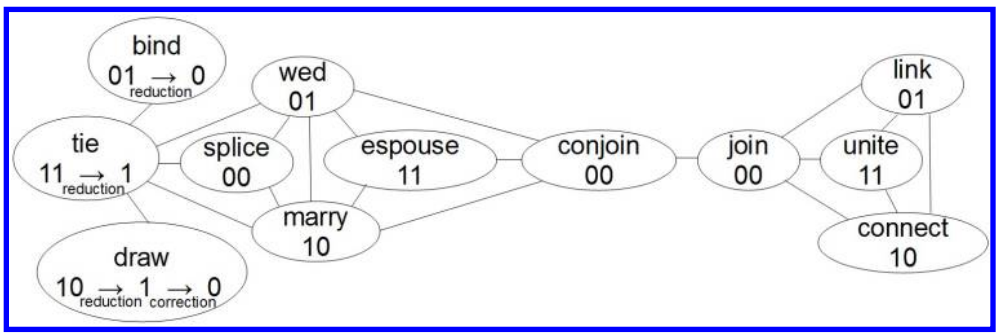
As described earlier, the proposed stegosystem extends the original synset by adding words in a synonym transitive closure chain while retaining the synonymous relationships between words using a synonym graph representation. The proposed NGM checker effectively controls the size of a synonym graph according to the context. After constructing the graph, namely, obtaining all the good alternatives for a cover word, the encoder generation module needs to assign codes to every word in the graph. In this section, we explain the coding method used in our stegosystem.

The aim of the proposed coding method is to convert an input synonym graph into a coded graph so that each vertex, namely, word, is encoded by a particular code. The method is inspired by the classic vertex coloring problem in graph theory (Gould 1988), where a coloring of a graph is a labeling of the graph's vertices with colors subject to the condition that no two adjacent vertices share the same color. However, in our proposed coding method, adjacent vertices are allowed to have the same code as long as each vertex is able to handle the prefix string of any secret message. A vertex can achieve this by either using its own code or a neighbor's code, as long as there is a guarantee that at least the first secret bit of the prefix can be embedded no matter which word in the graph is the cover word.

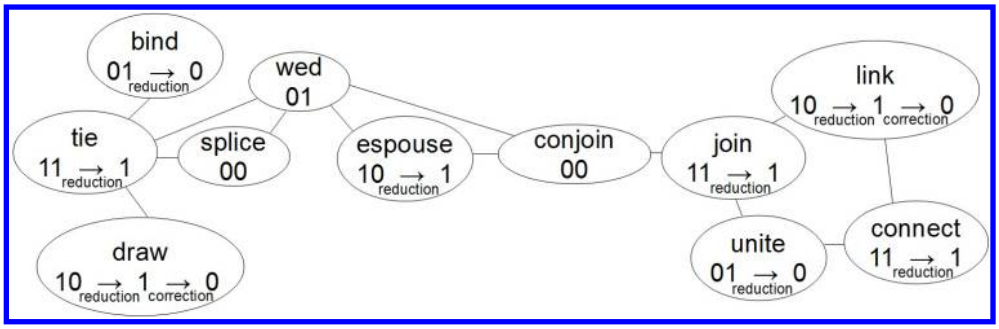
Let us first consider coding the synonym graph of the two joint synsets from Figure 10. Because both of the synsets have a size of four, which means the synsets can exhaust up to a two-bit coding space, four different two-bit codewords 00, 01, 10, and 11 are used to code the graph, as shown in Figure 15(a). As we can see, each word



(a)



(b)



(c)

Figure 15
Examples of coded synonym graphs.

in the graph has access to all the two-bit codewords. This means that the fundamental requirement of the coding method is satisfied: No matter what the target word is in the graph, any two-digit prefix of a secret message can be accommodated. In addition, the problematic word *marry* receives a unique codeword no matter which synset is considered, which means the secret recovery process will not encounter an ambiguity because the receiver can apply the same coding method to derive identical codewords used by the sender.

Next, let us consider coding the synonym graph in Figure 11(a). Again, four two-bit codewords are used because the maximum synset size is four in the synsets that make up this graph, and a coded version of the graph is shown in Figure 15(b). Note that it is acceptable to have *conjoin* and *join* encoded by the same codeword 00 because both of them have access to all the two-bit codewords. However, both *bind* and *draw* have only one neighbor, which means that only two codewords can be accommodated by these nodes, namely, bits 0 and 1. Therefore, instead of using two-bit codewords, the most significant bits are used to code these words and the neighbor, a process we call **codeword reduction**. In this example, the codewords of *bind*, *draw*, and *tie* are reduced to 0, 0, and 1, respectively. After codeword reduction, the vertex *draw* can only access codeword 1 so a further change is needed: The vertex's codeword is changed to 0 in order to accommodate either secret bit 0 or 1, a process we call **codeword correction**.

Note that the final coded graph, after codeword reduction and correction, satisfies the fundamental requirement that all vertices can represent some prefix of the secret message. Note also that some vertices can represent a longer prefix than others. For example, if the next part of the secret message to be embedded is 11, and the target word is *splice*, then *tie* would be chosen as the stego word, covering only the first bit of the prefix. However, if the target word is *wed*, then *espouse* would be chosen as the stego word, covering two-bits of the prefix. In general the secret embedding procedure will choose to cover as many bits of the secret message as possible at each point.

99.6% of synsets in WordNet have a size of less than eight, which means that most of the synsets cannot exhaust more than a two-bit coding space (i.e., we can only encode at most two bits using a typical synset). Therefore, we restrict the maximum codeword size in our coding method to two bits. The proposed method always starts coding a graph with two-bit codewords even if the maximum synset size of a graph is less than four, and then adjusts the assigned codewords by codeword reduction and codeword correction.

Figure 15(c) shows a coded synonym graph where the maximum synset size is three. We first want to make sure that *wed*, *tie*, *conjoin*, and *join* have access to all the two-bit codewords because they all have at least three neighboring vertices. Those vertices that have less than three neighbors are randomly assigned one of the four codewords such that no two adjacent vertices have the same codeword. After the two-bit codeword encoding, for a vertex that has only two neighbors, we first check whether the two neighbors are encoded by the same codeword. If they are, which means the vertex and its neighbors can accommodate only two codewords, then codeword reduction is applied to the vertex and both of its neighbors. For example, both the neighbors of *link* are encoded by codeword 11 so the codewords of *link* and its neighbors are replaced by the most significant bits. Then codeword correction is applied to *link* to ensure the access of both bit 0 and bit 1. Similarly, the codeword of *unite* is replaced by its most significant bit, but *unite* does not need codeword correction in this case.

However, if the two neighbors have different codewords, then the vertex has access to only three two-bit codewords and one of the two-bit codewords is missing. In this case the codeword that has the same most significant bit as the missing one must be

reduced by undergoing codeword reduction. For example, the two neighbors of *splice* have different two-bit codewords, and the missing codeword is 10. Among *splice*, *wed*, and *tie*, *tie* has the same significant bit from the missing codeword, so the codeword of *tie* is reduced to bit 1. Note that *splice* can now handle any message prefix: If the first bit is a 0, then two bits will be embedded (using either *splice* or *wed*); if the first bit is a 1 then only that one bit will be embedded (using *tie*). Similarly, the codeword of *espouse* is changed to bit 1.

Finally, *bind* and *draw*, which have only one neighbor, are adjusted as described for Figure 15(b). It is worth noting that, even though the maximum synset size in this example is three, it is possible to have a word carrying a two-bit codeword in the coded graph.

Figure 16 describes an algorithm for assigning two-bit codewords to each node in an input synonym graph (which is run before applying the processes of codeword correction and reduction). The *node* data structure has four fields: *word* is the label of

```
INPUT: a synonym graph G that has its nodes stored in Q1 and Q2
OUTPUT: a coded synonym graph G_coded using 2-bit codewords

struct node
        string word
        string code = null
        set_of_codes cannot_access = {00, 01, 10, 11}
        set_of_nodes neighbors

function AssignCode(w, code_set)
        IF code_set is empty THEN
                w.code = random({00, 01, 10, 11})
        ELSE
                w.code = random(code_set);
        END IF
        delete w.code from w.cannot_access
        FOR every n in w.neighbors
                delete w.code from n.cannot_access
        END FOR

Q = [Q1, Q2]
WHILE Q is not empty
        w = pop(Q)
        IF w.code is null THEN
                AssignCode(w, w.cannot_access)
        END IF
        FOR every n in w.neighbors
                IF n.code is null THEN
                        IF w.cannot_access is empty THEN
                                AssignCode(n, n.cannot_access)
                        ELSE IF n.cannot_access is empty THEN
                                AssignCode(n, w.cannot_access)
                        ELSE
                                AssignCode(n, w.cannot_access ∩ n.cannot_access)
                        END IF
                END IF
        END FOR
END WHILE
```

Figure 16
Algorithm for coding a synonym graph using two-bit codewords.

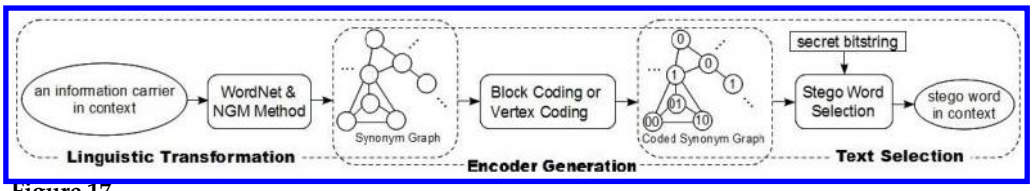


Figure 17
Framework of the proposed lexical substitution-based stegosystem.

the node corresponding to a word in a synset; *code* is the codeword of the node and is initialized to *null*; *neighbors* contains a set of neighboring nodes; and *cannot_access* records the two-bit codewords that cannot be accessed by the node and is initialized to 00, 01, 10, 11. Because we want a node to have access to all the two-bit codewords where possible, nodes that have at least three neighbors are stored in a priority queue Q_1. In this group of nodes, a node having three neighbors does not allow any of its neighbors to carry the same code so we assign codes to three-neighbor nodes and their neighbors first. Therefore, nodes in Q_1 are sorted by the number of neighbors such that nodes having the least neighbors are at the front of the queue. The other nodes are stored in a queue Q_2 such that nodes having two neighbors are at the front of the queue.

The function *AssignCode(w, code_set)* randomly chooses a codeword from *code_set*, or from the four two-bit codewords if *code_set* is empty. Then the function removes the chosen codeword from *w.cannot_access* and from the *cannot_access* sets of w's neighbors. The two-bit codeword assigning procedure first loops through all the nodes in Q_1 and then Q_2. For each node, the procedure first checks if a codeword has already been assigned; if not, it calls the *AssignCode* function. This assigns an appropriate code, as described earlier, and modifies the *cannot_access* field of both the node and the node's neighbors (since the new code is now accessible to both the node and its neighbors). It then loops through each of the node's neighbors, using the *AssignCode* function to assign a code to each neighbor if it does not already have one. Note that the set of available codes passed to the function depends on the *cannot_access* sets from both the node and the neighbor.

After all the nodes have been assigned two-bit codes using this algorithm, codeword reduction and codeword correction as previously described are applied to revise improper codewords.

4.1 Proposed Lexical Stegosystem

Figure 17 illustrates the framework of our lexical substitution-based stegosystem. Note that we asume that WordNet has been pre-processed by excluding multi-word synonyms and single-entry synsets. Table 12 shows the statistics of synsets used in our

Table 12
Statistics of synsets used in our 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

stegosystem. A possible information carrier is first found in the cover sentence. We define a possible information carrier as a word in the cover sentence that belongs to at least one synset in the pre-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, we assign codes to each word in the synonym graph(s). During the encoder generation 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 coding method 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.

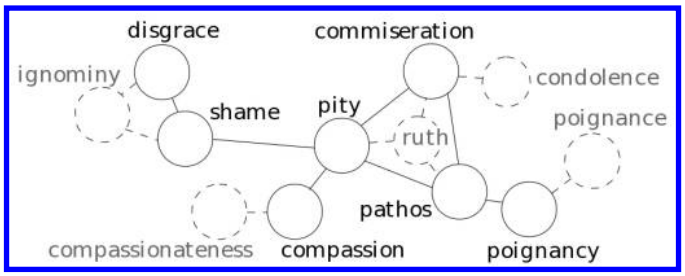
Repeating a comment made earlier, we use the transitive closure chain of WordNet containing the target word as a simple method to ensure that both sender and receiver encode the same graph. It is important to note, however, that the sender only considers the synonyms of the target word as potential substitutes;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

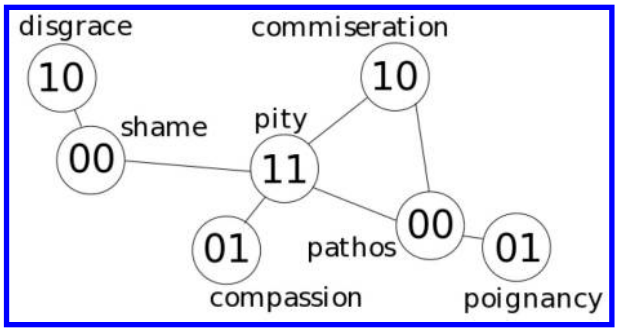
We demonstrate how to embed secret bit 1 in the sentence *it is a shame that we could not reach the next stage*. A possible information carrier *shame* is first found in the sentence. Table 13 lists the synsets in the synonym transitive closure chain extracted from WordNet. The score of each word calculated by the NGM method is given in parentheses. For the purpose of demonstrating the use of vertex coding, we select a low threshold score of 0.27. The output of the synonym graph is shown in Figure 18(a). Because the remaining words do not belong to the same synset, the vertex coding method is then used to encode the words. Figure 18(b) shows the coded synonym graph in which each vertex is assigned one of the four two-bit codewords; Figure 18(c) is the graph after applying codeword reduction and codeword correction. Although both *disgrace* and *pity* are encoded by 1, *pity* is chosen to replace the cover word because it has a higher score. Finally, the stego text is generated, *it is a pity that we could not reach*

Table 13
Synsets of *shame* in the synonym transitive closure chain with substitution scores.

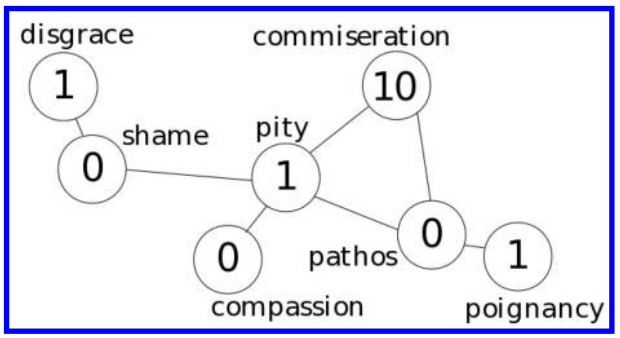
cover sentence: It is a *shame* that we could not reach the next stage.
{pity (0.97), shame (1.0)}
{shame (1.0), disgrace (0.84), ignominy (0.24)}
{commiseration (0.28), pity (0.97), ruth (0.13), pathos (0.31)}
{compassion (0.49), pity (0.97)}
{condolence (0.27), commiseration (0.28)}
{compassion (0.49),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



(a) The synonym graph of the synsets in Table 13



(b) Coded synonym graph using the four 2-bit codewords



(c) Coded synonym graph after codeword reduction and codeword correction

Figure 18
Synonym graphs generated by the proposed stegosystem.

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 an ignominy that we could not reach the next stage*. The same coded synonym graph as Figure 18(c) will be constructed because 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 with *disgrace* represents secret bit 1. In other words, a change must be made in order to embed a secret bit in this case.

In cryptography, Kerckhoffs's principle (Kerckhoffs 1883) 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 our steganography scheme, the secret key is the score threshold in the NGM method, and

the private randomizer is the one that assigns codes in the *AssignCode* function in the proposed vertex coding methods. The score threshold decides the size of a synonym graph, and the randomizer 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 code randomizer.

5. Conclusions and Future Work

One of the contributions of this work is to develop a novel lexical substitution-based stegosystem using vertex coding that improves the data embedding capacity compared to existing systems. The vertex 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 our stegosystem to ensure information security. The proposed stegosystem was automatically evaluated using the gold standard from the SemEval2007 lexical substitution task as well as a human evaluation. From the evaluation results we may conclude that our substitution-based stegosystem has achieved a reasonable level of security while reaching the payload capacity of around two bits per sentence.

In this work, we only evaluated the lexical substitution 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 substitution checkers are useful for the security of linguistic steganography at the document-level. In addition, apart from the linguistic transformations discussed in Section 2.1, we would like to explore more manipulations that can meet the requirements of linguistic steganography. As mentioned in Section 2.2, 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 we 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.

Research in linguistic steganography requires experience and knowledge of both information security and NLP. In addition, due to the complexity of language, there have been more studies on image or video steganography than linguistic steganography. Hence linguistic steganography is a relatively new research area, and further efforts are needed to develop more secure and efficient systems. The novel and original ideas provided in this article can benefit research in both computational linguistics and information security. It is hoped that our work can form the basis for more research devoted to linguistic steganography.

Appendix A. Screenshots of the Web-Based Human Annotation

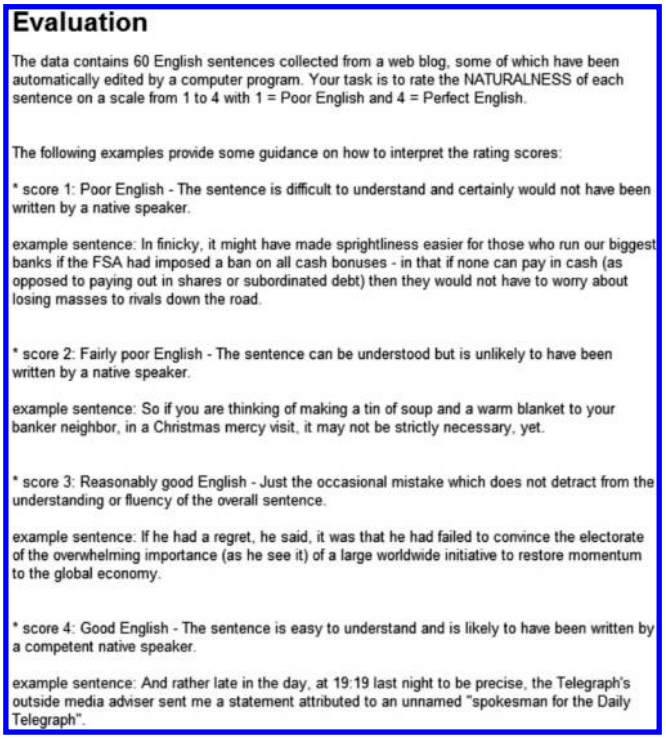


Figure A1
The introduction and guidelines for the lexical substitution annotation.

Appendix B. Original Text and the Substituted Words

The following text is part of the Robert Peston's BBC blog article titled "Unanswered questions about Cable"[10] where the 24 words in boldface are selected by the proposed NGM method as information carriers and are replaced with their synonyms as shown in the example in Section 3.4.2:

> *The whistleblower, who yesterday gave me the **full** recording, told me that the Telegraph's **omission** of these sections about Mr Murdoch was a commercial decision, **motivated** by the fact that the Telegraph - like **Mr** Cable - would rather News Corporation does not end up as 100% owner of BskyB.*
>
> *I of course **put** this to the Telegraph. And **rather** late in the day, at 19:19 last night to be **precise**, the Telegraph's external media adviser sent me a statement attributed to an **unnamed** "spokesman for the Daily Telegraph." The statement **says**:*
>
> *"It is **utter** nonsense to suggest that the Daily Telegraph did not publish comments from Vince Cable on the Rupert Murdoch takeover of BSkyB for commercial reasons. It was an editorial decision to focus this morning on Cable's comments on the Coalition because they **were** of wider interest to our readers."*

10 `http://www.bbc.co.uk/blogs/thereporters/robertpeston/2010/12/unanswered_questions_about_cab.html`.

*Well, some would say that was a **slightly** eccentric editorial decision for an editor, Tony Gallagher, widely regarded as one of the sharpest in the business. I rang **Mr** Gallagher to discuss this, but he directed me to the Telegraph's **internal** PR spokesperson.*

*Also, you may have **noticed** that the Telegraph has not **yet** put out any clear and **unambiguous** statement that it was ever planning to publish Mr Cable's remarks about Mr Murdoch (though it has now **published** them, after they were **put** out by the BBC).*

*Maybe I am being a bit naive and **silly** to think any of this matters. Maybe most of you **think** that what we do as reporters is so **obviously** and constantly subject to commercial interference that there is no **particular** benefit to be gained from asking the Telegraph to explain itself in this case.*

*But **actually** that's not been my experience in 27 years as a hack. And I still think the question of what news organisations put into the public domain, and how they do it, matters.*

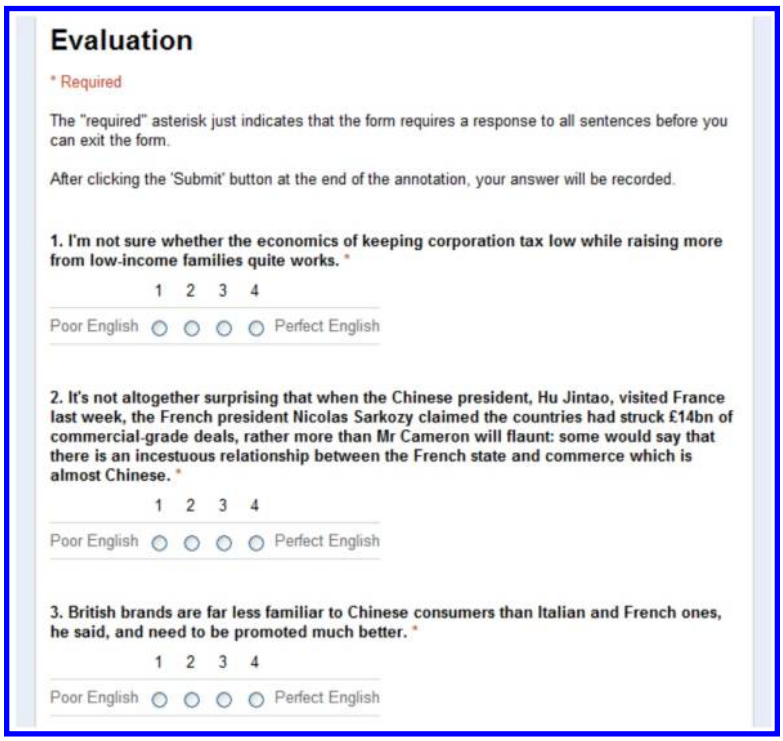


Figure A2
A screen capture of the lexical substitution annotation.

Acknowledgments

We would like to thank Dr. Laura Rimell, the anonymous reviewers, and all the contributing volunteer annotators for useful comments and their effort and time. This work was largely carried out while Ching-Yun Chang was a Ph.D. student at the Cambridge University Computer Laboratory, where she was supported by the Studying Abroad Scholarship from the Ministry of Education in Taiwan.

References

Atallah, Mikhail J., Craig J. McDonough, Victor Raskin, and Sergei Nirenburg. 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.

Atallah, Mikhail J., Victor Raskin, Michael C. Crogan, Christian Hempelmann, Florian

Kerschbaum, Dina Mohamed, and Sanket Naik. 2001. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Proceedings of the 4th International Information Hiding Workshop*, pages 185–199, Pittsburgh, PA.

Atallah, Mikhail J., Victor Raskin, Christian F. Hempelmann, Mercan Karahan, Umut Topkara, Katrina E. Triezenberg, and Radu Sion. 2002. Natural language watermarking and tamperproofing. In *Proceedings of the 5th International Information Hiding Workshop*, pages 196–212, Noordwijkerhout.

Bennett, Krista. 2004. Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text. CERIAS Technical Report 2004-13, Purdue University, Lafayette, IN.

Bergmair, Richard. 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, Richard. 2007. A comprehensive bibliography of linguistic steganography. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, pages W1–W6, San Jose, CA.

Bergsma, Shane, Dekang Lin, and Randy Goebel. 2009. Web-scale *n*-gram models for lexical disambiguation. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, pages 1,507–1,512, Pasadena, CA.

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

Brants, Thorsten and Alex Franz. 2006. Web 1T 5-gram corpus version 1.1. Linguistic Data Consortium, Philadelphia, PA.

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

Carlson, Andrew, Tom M. Mitchell, and Ian Fette. 2008. Data analysis project: Leveraging massive textual corpora using *n*-gram statistics. Technical Report CMU-ML-08-107, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Chang, Ching-Yun and Stephen Clark. 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, CA.

Chang, Ching-Yun and Stephen Clark. 2010b. Practical linguistic steganography using contextual synonym substitution and vertex color coding. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1,194–1,203, Cambridge, MA.

Chang, Ching-Yun and Stephen Clark. 2012a. Adjective deletion for linguistic steganography and secret sharing. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 493–510, Mumbai.

Chang, Ching-Yun and Stephen Clark. 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*, pages 511–528, Mumbai.

Chapman, Mark and George I. Davida. 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*, pages 335–345, Beijing.

Chapman, Mark, George I. Davida, and Marc Rennhard. 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.

Chen, Zhili, Liusheng Huang, Peng Meng, Wei Yang, and Haibo Miao. 2011. Blind linguistic steganalysis against translation based steganography. In *Proceedings of the 9th International Conference on Digital Watermarking*, pages 251–265, Seoul.

Clark, Stephen and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computation Linguistics*, 33(4):493–552.

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

Cox, Ingemar, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. 2008. *Digital Watermarking and Steganography*.

Morgan Kaufmann Publishers Inc., second edition.
Dinu, Georgiana and Mirella Lapata. 2010. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1,162–1,172, Cambridge, MA.
Doddington, George. 2002. Automatic evaluation of machine translation quality using *n*-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145, San Diego, CA.
Dorr, Bonnie, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*, pages 1–8, Edmonton.
Erk, Katrin and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 92–97, Uppsala.
Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.
Fridrich, Jessica. 2009. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press.
Gould, Ronald J. 1988. *Graph Theory*. Benjamin/Cummings Publishing Co., Menlo Park, CA.
Grothoff, Christian, Krista Grothoff, Ludmila Alkhutova, Ryan Stutsman, and Mikhail J. Atallah. 2005. Translation-based steganography. In *Proceedings of the 2005 Information Hiding Workshop*, pages 219–233, Barcelona.
Herodotus. 1987. *The History*. University of Chicago Press. Translated by David Grene.
Hoover, J. Edgar. 1946. The enemy's masterpiece of espionage. *The Reader's Digest*, 48:49–53. London edition.
Huffman, David A. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.
Islam, Aminul and Diana Inkpen. 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 1,241–1,249, Singapore.
Kahn, David. 1967. *The Codebreakers: The Story of Secret Writing*. Macmillan.
Kerckhoffs, Auguste. 1883. La cryptographie militaire. *Journal des Sciences Militaires*, IX:5–83.
Khairullah, M. D. 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, Mi-Young. 2008. Natural language watermarking for Korean using adverbial displacement. In *Multimedia and Ubiquitous Engineering*, pages 576–581, Busan.
Kim, Mi-Young. 2009. Natural language watermarking by morpheme segmentation. In *First Asian Conference on Intelligent Information and Database Systems*, pages 144–149, Dong Hoi.
Kirk, Roger E. 2012. *Experimental Design: Procedures for the Behavioral Sciences*. SAGE Publications, Inc, fourth edition.
Kishida, Kazuaki. 2005. Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. Technical Report NII-2005-014E, National Institute of Informatics, Tokyo.
Kullback, Solomon. 1959. *Information Theory and Statistics*. John Wiley and Sons, New York.
Kummerfeld, Jonathan K. and James R. Curran. 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.
Lampson, Butler W. 1973. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615.
Lee, Lillian. 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, MD.
Liu, Yuling, Xingming Sun, and Yong Wu. 2005. A natural language watermarking based on Chinese syntax. In *Advances in Natural Computation*, volume 3612, pages 958–961, Changsha.
McCarthy, Diana and Roberto Navigli. 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.
Meng, Peng, Liusheng Hang, Zhili Chen, Yuchong Hu, and Wei Yang. 2010. STBS: A statistical algorithm for steganalysis of translation-based steganography. In *Proceedings of the 12th International Conference, Information Hiding*, pages 208–220, Calgary.

Meng, Peng, Yun-Qing Shi, Liusheng Huang, Zhili Chen, Wei Yang, and Abdelrahman Desoky. 2011. LinL: Lost in n-best list. In *Proceedings of the 13th International Conference, Information Hiding*, pages 329–341, Prague.

Meral, Hasan M., Emre Sevinc, Ersin Unkar, Bulent Sankur, A. Sumru Ozsoy, and Tunga Gungor. 2007. Syntactic tools for text watermarking. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, pages X1–X12, San Jose, CA.

Meral, Hasan Mesut, Bülent Sankur, A. Sumru Özsoy, Tunga Güngör, and Emre Sevinç. 2009. Natural language watermarking via morphosyntactic alterations. *Computer Speech and Language*, 23(1):107–125.

Minnen, Guido, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7:207–223.

Murphy, Brian. 2001. Syntactic information hiding in plain text. Masters Thesis, Trinity College Dublin.

Murphy, Brian and Carl Vogel. 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*, pages Z1–Z9, San Jose, CA.

Murphy, Brian and Carl Vogel. 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, pages Y1–Y12, San Jose, CA.

Newman, Bernard. 1940. *Secrets of German Espionage*. Robert Hale Ltd.

Ó Séaghdha, Diarmuid and Anna Korhonen. 2011. Probabilistic models of similarity in syntactic context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1,047–1,057, Edinburgh.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.

Pfitzmann, Birgit. 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.

Por, Lip Y., Ang T. Fong, and B. Delina. 2008. WhiteSteg: A new scheme in information hiding using text steganography. *WSEAS Transactions on Computers*, 7:735–745.

Schuler, Karin Kipper. 2005. *Verbnet: A Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

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

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

Shen, Libin, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, Prague.

Shih, Frank Y. 2008. *Digital Watermarking and Steganography: Fundamentals and Techniques*. CRC Press.

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

Soderstrand, Michael A., Kenneth W. Jenkins, Graham A. Jullien, and Fred J. Taylor. 1986. *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. IEEE Press.

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

Spoustová, Drahomíra "Johanka," Jan Hajič, Jan Raab, and Miroslav Spousta. 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.

Stevens, Guy William Willis. 1957. *Microphotography: Photography at Extreme Resolution*. Chapman & Hall.

Stutsman, Ryan, Christian Grothoff, Mikhail Atallah, and Krista Grothoff. 2006. Lost in just the translation. In *Proceedings of the*

2006 ACM Symposium on Applied Computing, pages 338–345, Dijon.

Taskiran, Cuneyt M., Mercan Topkara, and Edward J. Delp. 2006. Attacks on lexical natural language steganography systems. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, pages 97–105, San Jose, CA.

Topkara, Mercan, Giuseppe Riccardi, Dilek Hakkani-Tür, and Mikhail J. Atallah. 2006. Natural language watermarking: Challenges in building a practical system. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, pages 106–117, San Jose, CA.

Topkara, Mercan, Cuneyt M. Taskiran, and Edward J. Delp. 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, CA.

Topkara, Mercan, Umut Topkara, and Mikhail J. Atallah. 2006a. Words are not enough: Sentence level natural language watermarking. In *Proceedings of the ACM Workshop on Content Protection and Security*, pages 37–46, Santa Barbara, CA.

Topkara, Umut, Mercan Topkara, and Mikhail J. Atallah. 2006b. 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.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 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.

Venugopal, Ashish, Jakob Uszkoreit, David Talbot, Franz Och, and Juri Ganitkevitch. 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 1,363–1,372, Edinburgh.

Vybornova, M. Olga and Benoit Macq. 2007. A method of text watermarking using presuppositions. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, pages R1–R10, San Jose, CA.

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

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

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

Winstein, Keith. 1999. Tyrannosaurus lex. Open source. Available at `http://web.mit.edu/keithw/tlex`.

Zhang, Yue and Stephen Clark. 2011. Syntax-based grammaticality improvement using CCG and guided search. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1,147–1,157, Edinburgh.

Zhu, Zhemin, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1,353–1,361, Beijing.