DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-074

Avoiding detection on twitter: embedding strategies for linguistic steganography

Alex Wilson, Andrew D. Ker
Department of Computer Science, Oxford University, United Kingdom

Abstract

Any serious steganography system should make use of coding. Here, we investigate the performance of our prior linguistic steganographic method for tweets, combined with perfect coding. We propose distortion measures for linguistic steganography, the first of their kind, and investigate the best embedding strategy for the steganographer. These distortion measures are tested with fully automatically generated stego objects, as well as stego tweets filtered by a human operator. We also observed a square root law of capacity in this linguistic stegosystem.

Introduction

The area of linguistic steganography has encountered little resistance in the form of steganalytic attacks. Of the attacks that have been published, the vast majority have focussed on attacking *cover generation* systems; an arguably unnecessary task, given these systems are fatally weak to human judges [1].

As for *cover modification*, a scant five papers (see 'Related Work') have attacked this type of system, and most system publications themselves have at best a passing reference to security, choosing instead to focus on the accuracy of the linguistic transformation (perhaps conflating syntactic and semantic correctness of the transformed sentence for security: see [2] for an example).

We attempted to address this with the *CoverTweet* system [3]. Designed to hide in *tweets* (140 character messages published on the so-called *micro-blogging* website Twitter), we employed human judges to verify its security; these judges were unable to distinguish genuine covers from manipulated stego objects.

Having established security against humans, we subsequently developed a statistical attack [4]. Individually, the generated tweets were hard to detect, but by adopting the *pooled steganalysis* paradigm, we found that looking at sequences of tweets vastly improved detection rates.

Steganography can be improved by employing *source coding* [5]. This technology, not previously deployed in linguistic steganography, solves the selection channel problem (not all sentences/tweets are suitable for hiding), improves efficiency (fewer changes are made), and allows us to quantify and minimize total distortion. In previous work we noted that the selection channel problem was significant for linguistic steganography.

Here, we follow up on our prior work [3, 4], introducing the first linguistic distortion measures. We examine how the attack performs when minimizing distortion through coding, and look at embedding strategies for the steganographer.

Related Work

The oldest available cover modification based linguistic stegosystem is *T-Lex* [6]. It uses a dictionary containing a number of disjoint synonym sets (extracted from WordNet [7]) to hide information. The synonyms in each set are unambiguously numbered, and payload embedded by changing cover words for the synonyms with numbers that convey the desired payload. In the domain of interest to this paper (tweets), T-Lex has a capacity of approximately 0.14 bits per tweet.

Subsequent systems have used a range of transformations to hide information (e.g.: adjective deletion [2], word order [8], and anaphora resolution [9]). Their implementations are not available or make use of annotated data sets that are not available.

There have also been attempts to build upon the synonym substitution method of T-Lex, such as [10], which uses a graph labelling technique to assign unambiguous values; this allows for non-disjoint synonym sets. CoverTweet [3] is another modern evolution of T-Lex, and generalises synonym substitution to paraphrasing, allowing for multi-word substitutions. The payload is conveyed through a keyed hash of the entire tweet, again allowing non-disjoint synonym sets, and can use a human operator to filter the transformed cover for fluency. This system is described in detail in the next section.

As mentioned, before our recent work there had been only five prior attempts at cover modification based linguistic steganalysis. Four of the attacks were against T-Lex, and the other an equivalent proprietary system [11].

The first such attack [12] used language models to extract features from stego text, before training a support vector machine (SVM) on the features. Its features are similar to some of those we use later. On generated stego sentences containing an average of 0.67 bits per sentence, this attack achieved an average error rate of 0.38 (0.15 false negative rate, and 0.61 false positive).

Subsequent work [13, 14, 15, 11] has used smaller models: they have all designed a *single* feature to exploit a particular weakness, and used this (or the mean and variance of it) to train a classifier for attack. Analysis of results, especially the effect of embedding rate on detection, has been lacking or non-existent. This focus on individual features echoes the early work on image steganalysis, which has since shifted towards feature-rich models.

CoverTweet

CoverTweet embeds information by using lexical and phrasal substitution rules taken from the *Paraphrase Database* (PPDB) [16]. The PPDB is a set of 169M paraphrase rules, automatically extracted from a number of bilingual corpora. Each pair of corpora were searched for non-English phrases that translate to two or more English ones; these English phrases are assumed to be paraphrases of each other. A rule in the PPDB contains a source string, a target string, and the estimated probability of the target being an appropriate substitution for the source (based on the

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-074

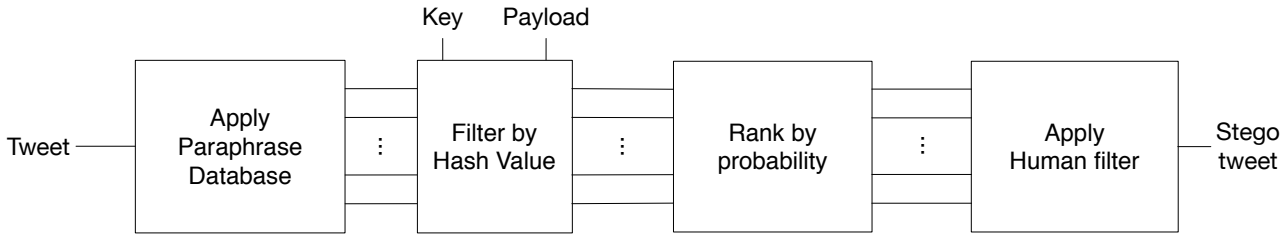


Figure 1. *The steps of CoverTweet.*

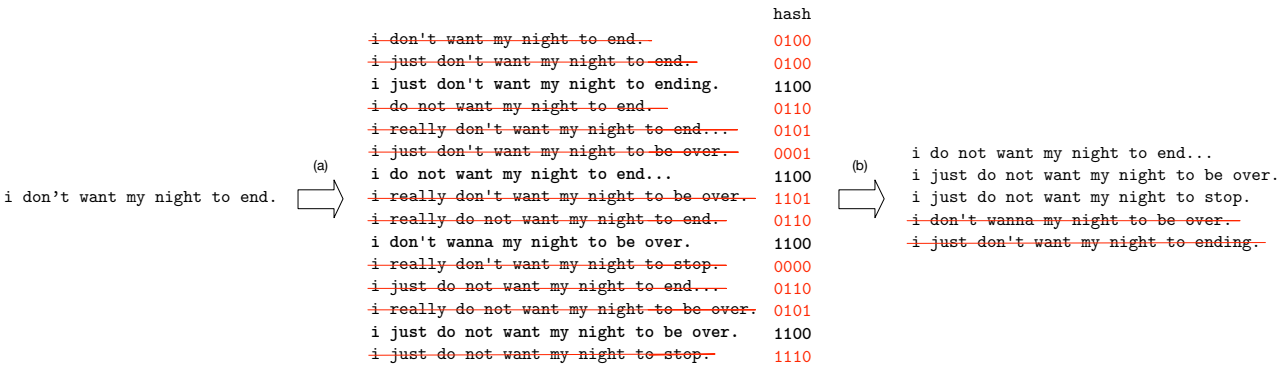


Figure 2. *Example of a tweet being transformed. The paraphrase database generates all possible stego objects for a given cover (a). These are then filtered according to hash value, and ordered by language model (b). The final list is filtered by a human, who goes on to choose the stego tweet to transmit. In this example the desired payload is 1100.*

number of times the pair is observed).

To use CoverTweet, the steganographer generates a cover object (a tweet), which is canonicalized and tokenized, then the system generates all possible stego sentences by applying all applicable PPDB rules. These possible stego objects are assigned values of a fixed number of bits with a hash function, then ranked by probability $\Pr(s|c)$, where s is the stego object and c is the cover object. See Figure 1 for a diagram of the system. This probability is derived using Bayes' rule:

$$\Pr(s|c) \propto \Pr(c|s)\Pr(s)$$

where $\Pr(c|s)$ is taken from the PPDB, and $\Pr(s)$ is provided by an n-gram language model for English text.

In computational linguistics, an n-gram is a sequence of n words, $w_1,\ldots,w_n$. An n-gram language model provides estimates for the probability of these sequences, often trained by counting instances of each n-gram in a large corpus (applying a smoothing method to the counts to avoid overfitting). The probability of a sentence, made up of a sequence of words $w_1,\ldots,w_T$, is approximated as:

$$\Pr(w_1,\ldots,w_T) \approx \Pr(w_1)\prod_{i=2}^{T}\Pr(w_i|w_{i-n},\ldots,w_{i-1})$$

To the reader familiar with steganographic literature, n-gram language models might be better recognised as $n-1^{\text{th}}$ order Markov models.

A fully automatic stegosystem could, at this point, rank all possible stego tweets (that have the correct hash) and transmit the most probable. However, this system often creates non-fluent tweets, and therefore the CoverTweet system uses a human operator to select the best option. An example of the embedding on a short tweet is shown in Figure 2.

In the original work [3] we found individual stego tweets, containing 4 bits of payload, were secure against human judges. However, those experiments required the removal of all tweets where the operator found no fluent option with the correct hash. The overall payload rate was much lower than 4 bits per tweet. In this paper we will address the selection channel problem.

Coding and Distortion

Assigning values with a hash function means that CoverTweet attempts to hide the same length of payload in every tweet. However, some sentences have more fluent paraphrases than others, so the true capacity should vary from tweet to tweet.

Furthermore, when the system could not embed a desired payload in a given tweet (if the tweet does not have any stego options with that hash value), the system discarded it and moved onto the next. In the initial attack on steganographic tweets, we made the assumption that such tweets could be recognised by the system, and therefore also by the detector (by Kerckhoffs' principle); the discarded tweets were excluded from any experiments.

In practice this assumption is unrealistic and impractical; some tweets without any options will be able to transmit payload, and some with many options might not have exactly the right option for the desired message. In addition, a tweet may have all its options filtered out by the human, which would be undetectable to the proposed system.

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-074

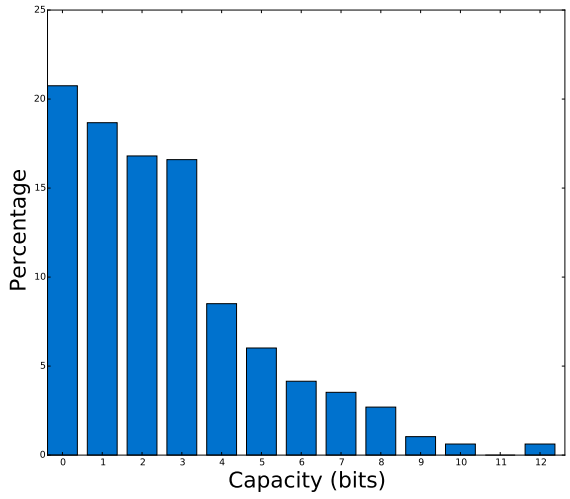


Figure 3. *Capacity of 1000 manually filtered cover tweets.*

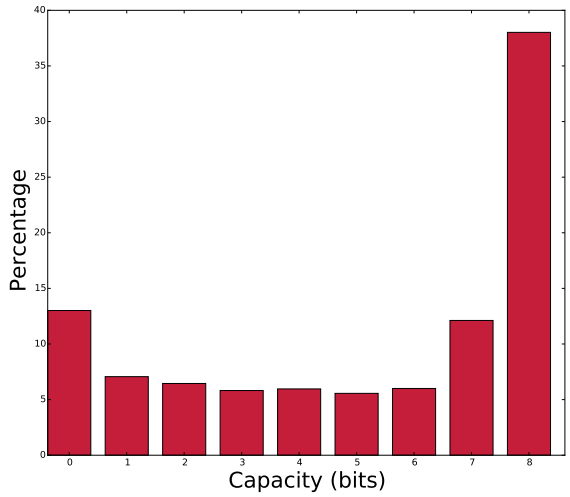


Figure 4. *Capacity of unfiltered cover tweets, capped at 8 bits.*

In a lossless communication system, the theoretical capacity of a tweet with k paraphrases (including itself) is $log_2 k$ bits. We examined the distribution of this capacity in 1000 tweets, with paraphrases both before and after human filtering. The results are shown in Figures 3 and 4; for the fully automatic paraphrases we capped the number of options at 256 (it could grows into the millions for some tweets, most of which were completely unusable). On average the capacity for manually filtered tweets is 2.8 bits and 21% of tweets cannot carry any payload; the average capacity for unfiltered tweets is 5.4 bits, with 13% unable to carry payload.

The proper solution is *source coding*. With this, we spread the payload across a number of tweets, treating the unusable tweets as noise. When we embed data in a cover object, any changes to the cover have a cost, or distortion; embedding a symbol in a tweet that does not have an option for it is defined as having an infinite distortion. Coding aims to minimise an additive distortion measure, enabling us to avoid these 'missing' symbols. Note that, due to the low capacity per tweet, spreading the payload across multiple tweets is already essential.

We denote the cost of embedding symbol j in cover i as d_{ij}, and p_{ij} the probability of this change being made by the steganographer. The average capacity of object i is the entropy of p_i

$$H(p_i) = -\sum_j p_{ij} \log_2 p_{ij}$$

and the average cost is

$$E(d_i) = \sum_j d_{ij} p_{ij}.$$

There are two ways to frame the task of minimising the embedding distortion:

- Payload Limited Sender (PLS): embedding a fixed average payload m while minimising average distortion, $\min E(d)$ s.t. $\sum_i H(p_i) = m$.
- Distortion Limited Sender (DLS): fixing the average distortion to a value D while maximising average payload, $\max H(p)$ s.t. $\sum_i E(d_i) = D$.

Both options have the same solution for p_{ij}:

$$p_{ij} = \frac{e^{-\lambda d_{ij}}}{\sum_j e^{-\lambda d_{ij}}}$$

for some constant λ, determined by either m or D. In practice we could use *syndrome trellis coding* to perform this embedding, but for our purposes we need only *simulate* it [17].

To simulate the embedding of a fixed payload across all tweets by one user, we find the value of λ that makes the total payload $\sum_i H(p_i)$ equal to our desired message size m. $H(p_i)$ is monotonic with respect to λ, so we can find it with binary search. For each object, we then change it to option j with probability p_{ij}.

The task of measuring and minimising distortion has never before been applied to linguistic steganography. We use four linguistic distortion measures, of which three are novel.

1. **Binary** A very basic distortion measure: 0 if the cover is unchanged, 1 otherwise.
2. **Probability** The log likelihood ratio of the cover object probability $\Pr(c)$ and the stego object probability $\Pr(s)$. Probabilities are estimated using an n-gram language model. The distortion is
   $$d_s = -(log \Pr(s) - log \Pr(c)).$$
3. **Edit distance** The minimum number of word-level edits (deletions, insertions and substitutions) required to turn the cover object into the stego object. For example, turning "`those are nice socks`" into "`those are not nice gloves`" has an edit distance of 2: one insertion (`not`) and one substitution (`socks` for `gloves`)

4. **Feature distortion** The Euclidean distance between the feature vector for the stego object and the cover object. This distortion measure is inspired by the HUGO image stegosystem which minimizes change to pixel co-occurrences [18]. Some of our features cannot be extracted for every stego object in reasonable time. Instead, we approximate these. The features (and approximations when necessary) are described in the following section.

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-074

Cover `ok but i doubt you'll be able to carry me`

Edit `ok but i doubt you will be able to carry me`
Prob. `ok but i don't think you'll be able to carry me`
Feature `okay but i doubt you will be able to carry me`

Figure 5. *A cover tweet with the best stego versions, according to the edit distance, probabiity and feature distortion measures*

A more ideal distortion measure might be the posterior likelihood, based on language model and paraphrase probabilities, that a given tweet is from an active steganographer: $\Pr(x|\text{active})$. Letting C be the set of all possible covers, this would be calculated as: $\Pr(x|\text{active}) = \sum_{c \in C} \Pr(c) \Pr(x|c)$. The subset of C for which $\Pr(x|c)$ is not zero can be produced by using the PPDB. Unfortunately this is expensive to calculate, and infeasible to do so for every possible stego tweet arising from a given cover.

Figure 5 shows an example tweet where the best substitution is different for two of the distortion measures.

Linguistic Steganalysis

We follow the main paradigm of steganalysis and train a model on features extracted from individual tweets. For the attack we use four classes of features, used to train a linear classifier ensemble [19]. These are the same features proposed in our previous work [4]. Descriptions of the features follow, along with the reasoning behind them.

Basic features. Simple statistical features, including word count, mean and variance of the number of characters in each word, and statistics for *stop word* usage. Stop words are common function words such as `the`, `as` and `just`; we use a list of 127 such words, and include features of individual counts for each stop word, along with a total count. The PPDB contains many rules that insert or remove these words, and these features are intended to capture that.

***n*-gram features**. Features based on probabilities of n-grams in the tweet, provided by a language model. For n from 1 to 5, the mean, variance and total log likelihood of each n-gram. Words will often be changed for less likely options during embedding; these features (inspired by the features in [12]) are designed to capitalize on this fact.

Word length features. We trained a 10-gram model of word length on the same data used to train the 5-gram language model. This allowed us to estimate probabilities of word length sequences. Using this model, we extract features equivalent to the n-gram features: for n from 1 to 10, the mean, variance and total log likelihood of each word length sequence of length n in the tweet. The distribution of word lengths is likely to be changed during embedding, due to the addition of stop words and the substitution of common short words for less common longer alternatives.

PPDB features. Justified by Kerckhoffs' principle, the attacker has access to the PPDB, CoverTweet's source of paraphrase rules. The PPDB is used to extract a number of features from each tweet: the mean, variance and total log likelihood of n-grams (again for n from 1 to 5) that contain at least one word that has an entry in the PPDB; the mean and variance of character counts for words in the PPDB; the proportion of words in the tweet that have entries in the PPDB. We apply CoverTweet's transformation to each stego object, extracting the likelihood of the most probable paraphrased sentence. Finally, a substitution score: each word or phrase in the tweet, with an entry in the PPDB, is replaced by its most likely substitute, according to the language model; the max, min, mean and variance of the log likelihoods for the resulting sentences are used.

When calculating the feature distortion it is infeasible to calculate the substitution score, and the likelihood of the most probable paraphrased sentence for each possible stego tweet. For our experiments we approximated the latter by using the likelihood of the most probable stego option for each *cover* tweet, meaning that each stego object for a given cover has the same value for this.

Although the PPDB features are squarely aimed at CoverTweet, we found that they were still viable for attacking T-Lex, a system with an entirely different paraphrase source. This may be in part due to the T-Lex source (a very small subset of the WordNet resource [7]) likely being subsumed by the much larger PPDB; indeed, overlap between large enough paraphrase sources is to be expected, meaning that these features might in general be useful against substitution based stegosystems. It is also worth noting that the PPDB features could be easily adapted to use any paraphrase source, should a different system be the target.

Pooled Steganalysis

We employ the *pooled steganaylsis* paradigm [20]. Instead of distinguishing individual innocent tweets from guilty, we pool together multiple pieces of evidence for each user, and distinguish innocent *actors* (here, twitter users) from guilty actors. For a set of tweets (of a fixed batch size), we calculate the maximum, minimum, mean and variance for each feature. The ensemble classifier is then trained on these features. Although individual steganographic tweets were difficult to detect in our initial experiments, pooling tweets caused a dramatic increase in detection rate.

Results

Data

We gathered 72M English language tweets, from 1.2M users, from the Harvard TweetMap [21]. Each tweet was canonicalised as follows: split into tokens, where tokens can be single words, punctuation, or a combination of the two (e.g. `this`, `-` and `s` are all tokens); made lowercase; usernames replaced by a uniform token; URLs replaced by a uniform token.

We randomly selected 500 users with 1000-2000 tweets to form our training and testing set. On the remaining tweets, we trained a 10-gram word length model and a 5-gram language model.

For 10 users from the set of 500, we generated every stego option for 100 cover tweets. The options were produced with CoverTweet, with a human operator removing non-fluent options. We refer to this data as Manual CoverTweet (M-CT).

Generating the manual data is very costly, so it was necessary to also generate data automatically. We used CoverTweet to embed up to 8-bits in each cover (for all 500 users) automatically selecting the stego object with the highest probability according to the language model. Finding the paraphrase with highest probability is an NP-complete problem [22], so we employ a greedy algorithm working from left to right in the tweet, pruning the substitutions with the lowest probability (provided by the PPDB) in

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-074

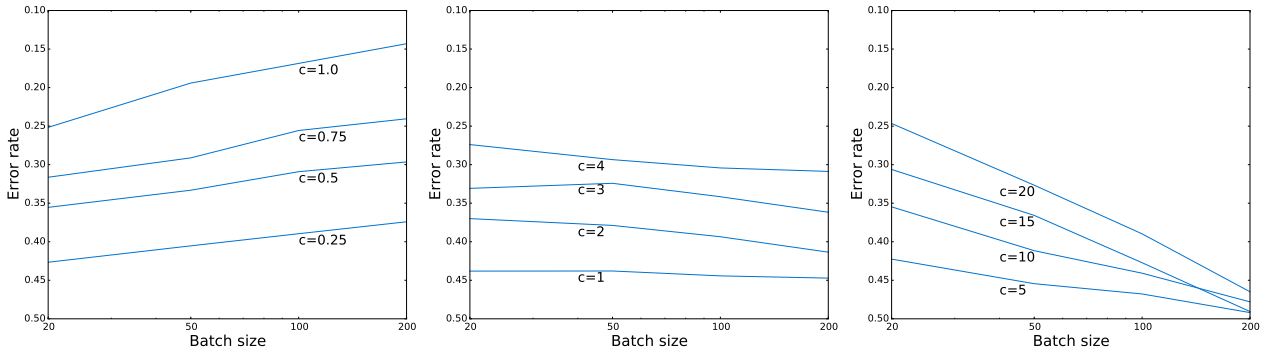


Figure 6. *Error rate vs. batch size for different mixes of cover and stego. Payload length: proportional to cover size, proportional to square root of cover size, constant in cover size.*

order to reduce the search space. We refer to this data as Automatic CoverTweet (A-CT).

We will measure capacity of the stegosystem (which includes coding) in *average bits per tweet*. For each A-CT user, we simulated the embedding of 1000, 2000 and 4000 bits across the 1000 tweets, corresponding to an average payload of 1, 2 and 4 bits per tweet. We used each of the four distortion measures we have described. For each M-CT user, we simulated the embedding of 100 and 200 bits across the 100 tweets; after human filtering, total capacity was too low to embed with an average payload of 4 bits per tweet.

We will benchmark the security of the system by the performance of the ensemble classifier [19], using default parameters. The error rate reported is the minimum value of the average false positive and false negative rates.

Square Root Law

It is known that there is a sublinear relationship between the total payload and the number of cover objects [23]. The square root law of steganography states that if a steganographer embeds a total of M bits into N cover objects, then:

- If $M/\sqrt{N} \to \infty$ as $N \to \infty$, then there is a pooled detector which, for sufficiently large N, comes arbitrarily close to perfect detection.
- If $M/\sqrt{N} \to 0$ as $N \to \infty$, then the performance of any pooled detector must become arbitrarily close to random for sufficiently large N.

This is a theoretical result, but it has been observed robustly in image steganography. As a step toward serious linguistic steganalytic research, we will show empirically that this holds also for this linguistic stegosystem, adding to the growing evidence for the square root law.

We created batches of our data containing a mix of N stego and cover objects, with a total payload size of M bits. First, we let $M = cN$, a linear payload size, and trained our classifier on these batches, for a range of N. Next, we made $M = c$, so that the payload size is constant. Finally, we set $M = c\sqrt{N}$. Figure 6 shows the results, which confirm the theoretical predictions: when the payload size is proportional to total cover size, the detection rate increases with the batch size; when the payload is constant, detection rate decreases; finally, when the payload is proportional to the square root of the total cover size, detection rate stays approximately the same.

Embedding Strategy

When embedding in tweets, the steganographer can choose to spread the payload out across all cover objects, or to embed a higher payload in fewer objects, leaving some unchanged. An advantage of the former is that the coding efficiency gain is greater [17], but with the latter there are, in theory, fewer opportunities to make a non-fluent error. In the context of batch steganography in images, this was studied in [24]. With our linguistic steganalysis attack, we wish to study the same question.

Initially just using the data generated with the binary distortion, we randomly mixed stego and cover objects for each A-CT user to create batches of tweets with three different embedding strategies, containing the same total payload of 1000 bits in 1000 tweets: 1 average bit per tweet in 100% of the covers; 2 average bits per tweet in a randomly-chosen 50% of the covers; 4 average bits per tweet in 25% of the covers. We trained and tested the classifier on features extracted from batches of the tweets (for batch sizes up to 200). The results are shown in Figure 7(a).

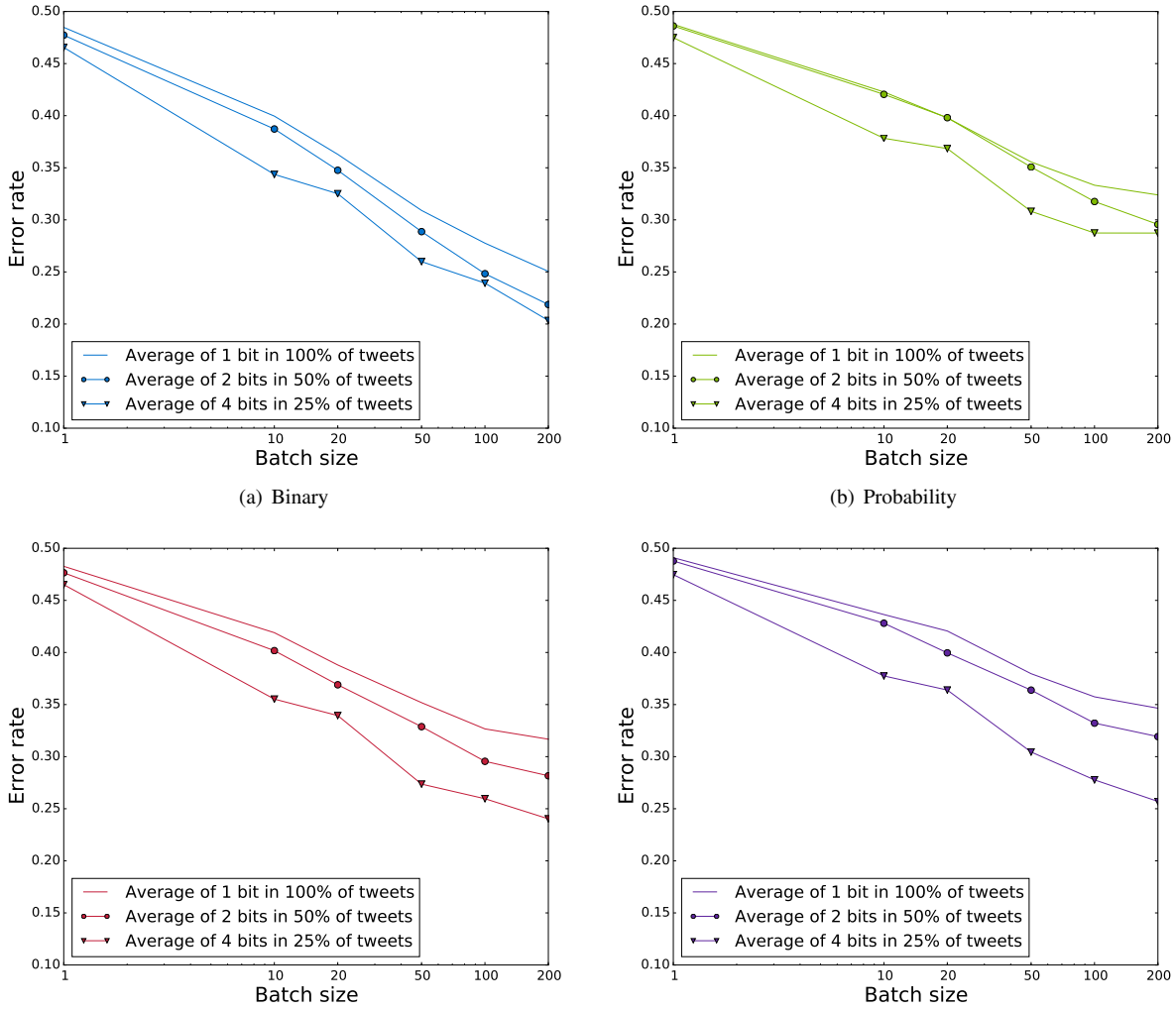
On individual tweets, detection is near random for all average payload sizes; this echoes the findings of the original work, where individual tweets proved difficult to detect by humans. On all pooled results, concentrating the payload made it slightly easier to detect: for example at batch size 100, the error rate was 0.28 using 100% of covers, and 0.24 when the same payload is concentrated into 25% of them.

Distortion

We repeated the previous experiment with the more refined distortion measures. Figures 7(b)-(d) show the results for the experiment with probability, edit distance, and feature vector distortions.

Using refined distortions makes the system more secure. In other respects, the classifier behaved similarly: in all cases, spreading the payload out across all tweets is harder to detect, and a larger pool of evidence (larger batches) makes for more accurate detection. The data embedded with the feature vector distortion is the hardest to detect; this was to be expected, given it works to explicitly minimise the distance between the feature vector for stego objects and their corresponding cover object. It therefore

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-074



(a) Binary (b) Probability

(c) Edit Distance (d) Feature

Figure 7. *Error rates for classifiers trained and tested on A-CT data, for each distortion type and range of batch sizes.*

assumes knowledge of the enemy's detector. The next most effective distortion measure was probability, with edit distance after that.

When this attack was originally used against un-coded stego, the error rates were significantly lower. For example, on batches of 100 tweets each containing exactly 1 bit of payload per tweet, the classifier achieved an error rate of 0.1. Even with the simple binary distortion, the coded batches are far more secure: against a batch of 100 containing an average 1 bit per tweet, the error rate is 0.28.

Manual data

We have seen how coding can affect detection of automatically generated tweets, but the A-CT data is flawed: some automatic tweets are easily detectable to a knowledgeable human. For example, the A-CT data contains tweets such as "`if i go thereafter she willingness go`" (originally "`if i go then she will go`"), and "`you should try to come rearward into town...`" (`rearward` originally being `back`). Because of this, we are most interested in the M-CT data, which has been filtered by a human so that only valid substitutions are used; as already noted, we know manually filtered individual tweets are secure against humans. When we first presented these linguistic steganalysis features [4], the attack was significantly worse at detecting manually generated tweets (at a batch size of 100, the classifier achieved an error rate of 0.42 on manual data vs. 0.1 on automatic).

With the M-CT data for our 10 users, we trained the classifier on data from 9 users, and tested on the remaining 1. This was repeated 10 times, holding back a different user for testing each time, and we calculated the average error rate across the users. We augmented the testing data with additional covers from other users that were not seen in the testing set, giving us a good estimate of false positive rate; however, our false negative rate is based on a very small number of testing samples.

The results are shown in Table 1. Due to the small amount of training and testing data, it is difficult to draw any conclusions. We are hesitant to make any claims based on these noisy results, other than: we do not have enough manually filtered training data. However, we do have vast amounts of *automatic* data.

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-074

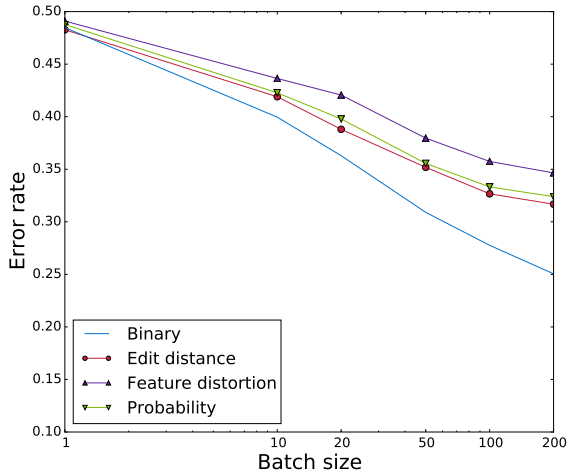


Figure 8. *A comparison of the most secure embeddings for each distortion type, with A-CT data.*

		Batch Size		
		5	10	20
Binary	1 bit in 100%	0.498	0.461	0.413
	2 bits in 100%	0.492	0.463	0.442
Edit distance	1 bit in 100%	0.462	0.486	0.395
	2 bits in 100%	0.498	0.469	0.462
Probability	1 bit in 100%	0.436	0.470	0.425
	2 bits in 100%	0.485	0.471	0.497
Feature	1 bit in 100%	0.470	0.461	0.492
	2 bits in 100%	0.492	0.471	0.458

Error rates for classifiers trained and tested on M-CT data, with a range of batch sizes, distortions and average payloads.

We trained the classifier on all the A-CT data, minus the A-CT data for the users for whom we had manual data. The testing set was made from all M-CT data, with the addition of all cover instances from the same users; this gives 10000 cover tweets and 1000 stego tweets for testing. For each average payload and distortion type for the M-CT data, we used the same for the A-CT training data (so we were training with tweets embedded with the same distortion that we were testing with).

Figure 9 shows the results of these experiments. Note that we do not include results for an average of 2 bits in 50% of the tweets. Seemingly due to the distribution of distortion on the manual data, and the presence of a small number of very high capacity tweets, manual data with an average of 1 bit embedded in every tweet contains a very high percentage of unchanged tweets; raising the average number of bits to 2, and hiding in only 50% of the data actually *increases* the number of changes made by a small amount. The strategy is no longer a viable one, as the entire point of it is to require fewer changes overall.

The manual data is far more secure than its A-CT counterpart, in all cases. It is difficult to differentiate between the distortion measures, or indeed between the batches coded to contain an average of 1 bit per tweet and those containing an average of 2 bits per tweet. With a few exceptions (mostly at the highest batch size), the classifier performs not much better than random guessing; it is unknown whether this is entirely due to the lack of data, or from the distortion minimization effect of coding.

It is possible that the distortion measure of the training data has an effect, and M-CT could be better detected by training on A-CT data embedded with a different measure (after all, the human filter is itself a kind of distortion measure). We explored this option but did not find anything significant to report. The main problem here is, of course, one of *domain adaptation* or *model mismatch* [25, 26]. In-domain (manually filtered) data is prohibitively expensive to produce, for any Warden who wishes to deploy linguistic steganalysis, but we have an abundance of automatic data. This is a problem well known to the steganographic community, and finding a solution in this instance is beyond the scope of the work here. The key to improving detection of manual data may lie in the generation of better quality automatic data.

Conclusions

Prior work on linguistic steganography has suffered from the non-shared selection channel problem. It resulted in loss of capacity or artificial experimental results. However, it can be solved by source coding, which we have employed here.

With human-filtered data, we found that the perfectly-coded capacity of tweets was approximately 2.8 bits (disregarding security for the moment). This is favourable quantity compared with the 4 bits per tweet reported in [3], which could only use approximately 48% of tweets. The gain is due to coding.

We hope this work acts as a starting point for serious linguistic steganalysis. Here, we proposed three linguistic distortion measures; the first of their kind. Minimising these distortions through the simulation of perfect coding let us evaluate them against a previously developed attack. For automatically generated stego objects, the effect of the distortion measures was clear, but the performance of each measure on manually filtered stego was difficult to judge. The creation of better automatic data for training, and additional manual data for testing is vital to allow for further work on these, and additional, distortion measures.

The features we proposed in [4] are only a first step to linguistic steganalysis. One could imagine *rich features* based on diverse language models, combinations of n-gram models for different n (and trained on different sources), different paraphrase sources, domain-adapted models for each user, and we hope that these advances will be forthcoming. In addition, further evaluation is needed. The performance of the attack on systems other than CoverTweet, and on non-tweet domains, is of interest: at present the security of new systems is not satisfactorily evaluated, in part due to there being no available attack designed for more than one stegosystem.

References

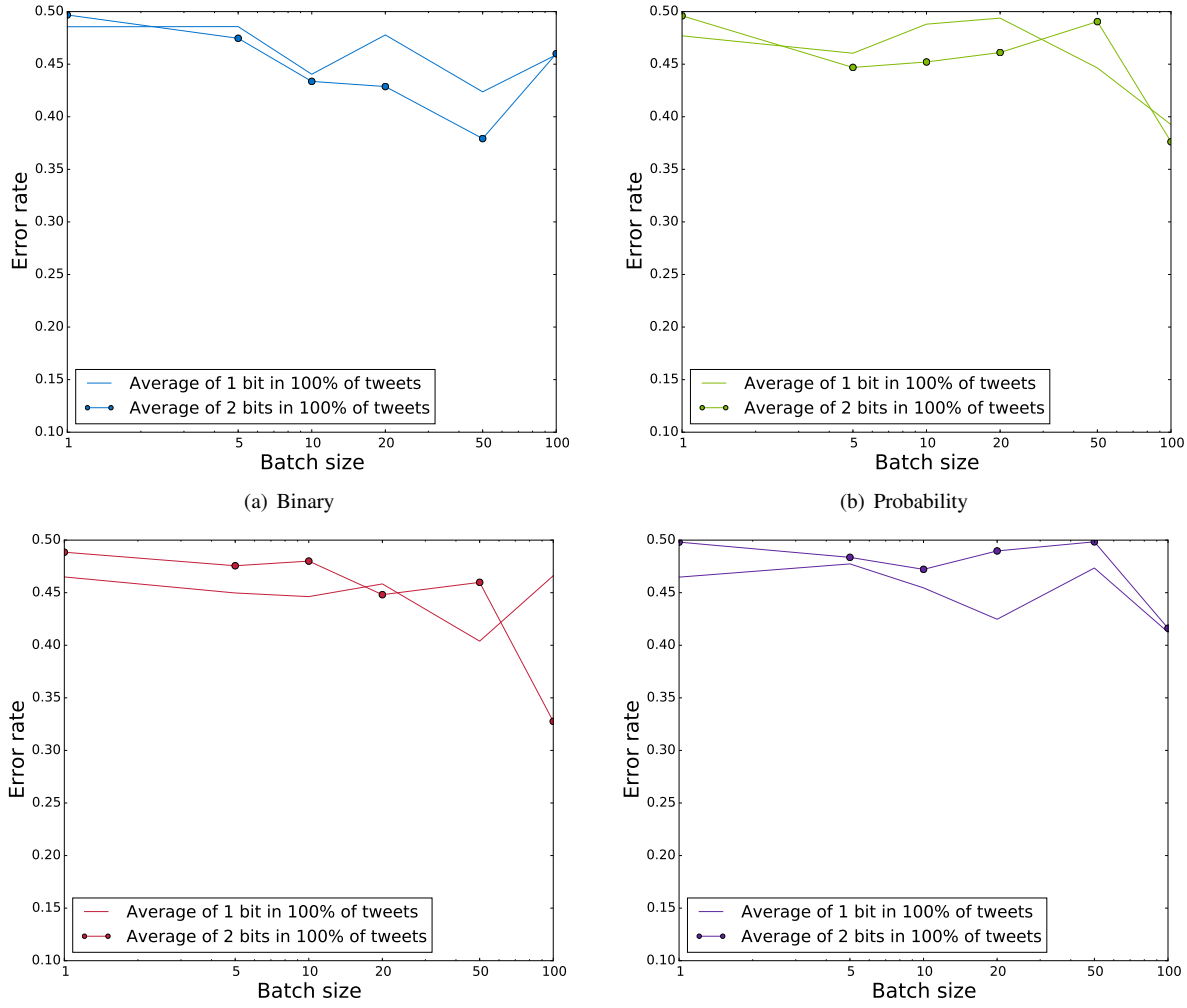
[1] M. Grosvald and C. O. Orgun, "Free from the cover text: a human-generated natural language approach to text-based steganography," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 2, pp. 133–141, 2011.

[2] C.-Y. Chang and S. Clark, "Adjective deletion for linguistic steganography and secret sharing.," in *COLING*, pp. 493–510, 2012.

[3] A. Wilson, P. Blunsom, and A. D. Ker, "Linguistic steganography on twitter: hierarchical language modeling with manual interaction," in *IS&T/SPIE Electronic Imaging*, pp. 201–217, International Society for Optics and Photonics, 2014.

[4] A. Wilson, P. Blunsom, and A. Ker, "Detection of steganographic

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-074



(a) Binary

(b) Probability

(c) Edit distance

(d) Feature

Figure 9. *Error rates for classifiers trained on A-CT data and tested on M-CT data, for each distortion type and range of batch sizes.*

techniques on twitter," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2564–2569, Association for Computational Linguistics, 2015.

[5] J. Fridrich, *Steganography in digital media: principles, algorithms, and applications*. Cambridge University Press, 2009.

[6] K. Winstein, "Lexical steganography through adaptive modulation of the word choice hash. `http://www.imsa.edu/~keithw/tlex`," *Unpublished*, 1998.

[7] G. A. Miller, "Wordnet: A lexical database for english," *COMMUNICATIONS OF THE ACM*, vol. 38, pp. 39–41, 1995.

[8] C.-Y. Chang and S. Clark, "The secret's in the word order: Text-to-text generation for linguistic steganography.," in *COLING*, pp. 511–528, 2012.

[9] O. Vybornova and B. Macq, "Natural language watermarking and robust hashing based on presuppositional analysis," in *Information Reuse and Integration, IEEE International Conference on*, pp. 177–182, IEEE, 2007.

[10] C.-Y. Chang and S. Clark, "Practical linguistic steganography using contextual synonym substitution and vertex colour coding," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 1194–1203, Association for Computational Linguistics, 2010.

[11] L. Xiang, X. Sun, G. Luo, and B. Xia, "Linguistic steganalysis using the features derived from synonym frequency," *Multimedia tools and applications*, vol. 71, pp. 1893–1911, 2014.

[12] C. M. Taskiran, U. Topkara, M. Topkara, and E. J. Delp, "Attacks on lexical natural language steganography systems," in *IS&T/SPIE Electronic Imaging*, pp. 120–129, International Society for Optics and Photonics, 2006.

[13] S. Xin-guang, L. Hui, and Z. Zhong-liang, "A steganalysis method based on the distribution of characters," in *Signal Processing, 2006 8th International Conference on*, vol. 4, pp. 54–56, IEEE, 2006.

[14] Z. Yu, L. Huang, Z. Chen, L. Li, X. Zhao, and Y. Zhu, "Steganalysis of synonym-substitution based natural language watermarking," 2009.

[15] Z. Chen, L. Huang, H. Miao, W. Yang, and P. Meng, "Steganalysis against substitution-based linguistic steganography based on context clusters," *Computers & Electrical Engineering*, vol. 37, pp. 1071–1081, 2011.

[16] J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, "PPDB: The

DOI: 10.2352/ISSN.2470-1173.2016.8.MWSF-074

paraphrase database.," in *HLT-NAACL*, pp. 758–764, 2013.

[17] T. Filler, J. Judas, and J. Fridrich, "Minimizing additive distortion in steganography using syndrome-trellis codes," *Information Forensics and Security, IEEE Transactions on*, vol. 6, no. 3, pp. 920–935, 2011.

[18] T. Pevnỳ, T. Filler, and P. Bas, "Using high-dimensional image models to perform highly undetectable steganography," in *Information hiding*, pp. 161–177, Springer, 2010.

[19] J. Kodovskỳ, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 2, pp. 432–444, 2012.

[20] A. D. Ker, "Batch steganography and pooled steganalysis," in *Information Hiding*, pp. 265–281, Springer, 2007.

[21] T. Mostak, "Harvard TweetMap. accessed October 2013. `https://worldmap.harvard.edu/`," 2013.

[22] K. Knight, "Decoding complexity in word-replacement translation models," *Computational Linguistics*, vol. 25, no. 4, pp. 607–615, 1999.

[23] A. D. Ker, T. Pevnỳ, J. Kodovskỳ, and J. Fridrich, "The square root law of steganographic capacity," in *Proceedings of the 10th ACM workshop on Multimedia and security*, pp. 107–116, ACM, 2008.

[24] A. D. Ker and T. Pevny, "The steganographer is the outlier: realistic large-scale steganalysis," *Information Forensics and Security, IEEE Transactions on*, vol. 9, no. 9, pp. 1424–1435, 2014.

[25] A. D. Ker, P. Bas, R. Böhme, R. Cogranne, S. Craver, T. Filler, J. Fridrich, and T. Pevnỳ, "Moving steganography and steganalysis from the laboratory into the real world," in *Proceedings of the first ACM workshop on Information hiding and multimedia security*, pp. 45–58, ACM, 2013.

[26] A. D. Ker and T. Pevny, "A mishmash of methods for mitigating the model mismatch mess," in *IS&T/SPIE Electronic Imaging*, pp. 90280I–90280I, International Society for Optics and Photonics, 2014.