Detection of Steganographic Techniques on Twitter

Alex Wilson and **Phil Blunsom** and **Andrew D. Ker**
Department of Computer Science
University of Oxford
Oxford, OX1 3QD, UK
{alex.wilson,phil.blunsom,andrew.ker}@cs.ox.ac.uk

Abstract

We propose a method to detect hidden data in English text. We target a system previously thought secure, which hides messages in tweets. The method brings ideas from image steganalysis into the linguistic domain, including the training of a feature-rich model for detection. To identify Twitter users guilty of steganography, we aggregate evidence; a first, in any domain. We test our system on a set of 1M steganographic tweets, and show it to be effective.

1 Introduction

Consider this: two isolated prisoners, communicating by letters scrutinised by the prison warden. They cannot write openly about escape plots, and the warden destroys any written in code. They must *hide* the true message within the letter, using *steganography*: the art of hiding information.

In *cover modification*[1] steganography, a *cover* object is tweaked so that it carries a hidden message: this is called *embedding*, the tweaked version is the *stego* object, and the message is the *payload* (Fridrich, 2009). The message should not be detectable to any observer (the standard terminology for this is the *warden*, taken from the prisoner metaphor) who knows the system is deployed, but does not know the original cover.

We are concerned here with *linguistic steganography*, in which the cover is a piece of text, and the message is embedded using textual transformations intended to preserve the meaning of the original: synonym substitutions, syntactical transformations, etc. Note that we are not concerned with the subset of linguistic steganography that hides in file formatting (e.g. white space, as in Por et al. (2008)), which has no security against an informed warden (in the case of hiding information by adding extraneous white space, the warden simply has to look for consistent irregular use of spaces to spot an active steganographer).

The field suffers from a number of issues: compared to images, text covers have low capacity (Chang and Clark, 2010); certain methods are weak against human attackers (Grosvald and Orgun, 2011) (most paraphrase systems cannot guarantee perfectly fluent stego objects); finally, authors are generally concerned with the performance of the transformation (whether they produce grammatically/semantically correct transformations), rather than whether the generated stego objects are detectable or not (e.g. Chang and Clark (2010)).

However, there is a new challenger in the field. We proposed a new linguistic stegosystem (Wilson et al., 2014) and verified its security against human judges, who were unable to distinguish genuine covers from manipulated stego objects.

This paper aims to attack CoverTweet statistically. We are in the shoes of the warden, attempting to classify stego objects from innocent[2] cover objects. We propose techniques new to linguistic steganalysis, including a large set of features that detect unusual and inconsistent use of language and the aggregation of evidence from multiple sentences. This last development, known in the steganographic literature as *pooled steganalysis* (Ker, 2007), represents a first in both linguistic and image steganalysis.

[1] There are other steganographic paradigms, not in scope. *Translation* based methods hide information in the automatic translation of the cover (e.g. Meng et al. (2011)). Cover *generation* methods automatically produce text containing the payload (e.g. Chapman et al. (2001)).

[2] It is usual to call steganographers and their output 'guilty' (with non-steganographers and unchanged cover objects being 'innocent'). This has the possibility of seeming politically charged, so we will use the term 'active' instead: be aware that this is not the usual terminology.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2564–2569,
Lisbon, Portugal, 17-21 September 2015.

2 Linguistic Stegosystems

T-Lex (Winstein, 1998) is the oldest available cover-modification based linguistic stegosystem. It uses a dictionary containing a small number of disjoint synonym sets extracted from WordNet (Miller, 1995). Each set is unambiguously ordered (e.g. alphabetically), then values are embedded by changing cover words for their synonyms. Due to the the small dictionary, the *capacity* of covers is only ~ 0.1 bits per sentence.

CoverTweet is a modern evolution of T-Lex. It hides information in tweets by applying paraphrase rules taken from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013), a set of 169M rules. The system applies suitable rules to a given cover, generating a set of possible stego objects. These are ranked by a distortion measure (derived from the probabilities of applied rules and from sentence probabilities given by a language model), and assigned a keyed hash. A human operator filters the options for fluency, and chooses the best stego object with the desired hash.

CoverTweet uses a subset of the PPDB, restricted to lexical and phrasal substitutions. Even with the reduced set of rules, 4 bits can be embedded per tweet, and it was proven secure against human judges (Wilson et al., 2014).

Twitter is a realistic setting for steganography. There is precedent for information hiding and mass monitoring on micro-blogging sites, such as the use of code words and government censorship on the Chinese website Sina Weibo (Chen et al., 2013). For this reason, we are attacking this setting.

There are many other linguistic stegosystems, using an array of different hiding methods (e.g. adjective deletion, word order, anaphora resolution: (Chang and Clark, 2012a), (Chang and Clark, 2012b), (Vybornova and Macq, 2007)). Approximately 1 bit of payload per cover sentence is usual, making CoverTweet the exception. Unfortunately for steganalysis literature, the vast majority of these require data that is unavailable (and too expensive to reproduce); beyond CoverTweet, the only system that can be evaluated is T-Lex.

3 Related Work

To our knowledge, there have been only five prior attempts at linguistic steganalysis on cover modification based systems; of these, four attack T-Lex, the other attacks an equivalent proprietary system. Taskiran et al. (2006) was the first, using n-gram language models to extract features from stego text, before training a support vector machine (SVM) on the features. We will adopt some of these features for our attack.

Subsequent work (Xin-guang et al. (2006); Yu et al. (2009); Chen et al. (2011); Xiang et al. (2014)) has used smaller models: they have all designed a *single* feature to exploit a 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. We will be utilising the latter here, in addition to the pooled steganalysis paradigm.

4 Proposed features

Below we describe four classes of proposed features for individual tweets. Each class will be evaluated individually, and in combination.

Basic features Including: word count (including tokenised punctuation); the mean and variance of the number of characters in each word; the total stop word count[3]; and the counts for each individual stop word. (131 features)

n-gram features Using a 5-gram model, for n from 1 to 5, the mean, variance and total log likelihood of the n-grams in the tweet. (15 features)

Word length Equivalent to the n-gram features, using a 10-gram model of word length. We expect the PPDB to replace common words with uncommon longer words, or multiple shorter words. (30 features)

PPDB features *Kerckhoffs' principle* (Fridrich, 2009), which states that a stegosystem must be secure even when the attacker (*us*) knows how the system works, introduces an interesting opportunity for linguistic steganalysis. As all linguistic stegosystems rely on automatic paraphrasing, and generally require a source of data, it is conservative to assume that the attacker has access to this data source. As the attacker, we can derive features from this data.

Applying this, we extracted features using the same subset of the PPDB that CoverTweet uses. To our knowledge, this is the first work to explicitly apply Kerckhoffs' principle to linguistic

[3]We use the Natural Language Toolkit (Bird, 2006) for the list of 127 stop words.

steganalysis. This set includes: a repeat of the n-gram features, but only taking into account n-grams that contain at least one word in[4] the PPDB; the mean and variance of character count for words in the PPDB; for phrase lengths j from 1 to 5, the proportion of j-grams in the tweet that are in the PPDB; likelihood of the most probable paraphrased sentence, extracted by generating sentences in the same way as CoverTweet. Finally, a substitution score: each word or phrase 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. (27 features)

4.1 Pooling the features

Individual linguistic stego objects carry a tiny payload, making detection an incredibly difficult task. In some cases, the stego tweet will be identical to the cover, making the task impossible. Although we do not hope to reliably identify stego tweets individually, the warden can still try to catch the active steganographers: through *pooled steganalysis* (Ker, 2007). Here, the warden considers all the evidence for each user, and aims to identify those actively using steganography. We combine features from batches of tweets, by the same user: taking the max, min, mean and variance of each feature. If performing steganography causes small consistent changes to a feature, this will be apparent in the mean. If it causes inconsistent changes, it may be apparent in the variance. If it occasionally produces an extreme result, the max and min will capture this. (812 features)

5 Method and Results

Here we take the role of an automatic censor, attempting to identify users on Twitter who are sending secret messages. We imagine a scenario where CoverTweet is openly available, and so can assume this is the system utilised by the active users.

5.1 Data and Embedding

From the Harvard TweetMap (Mostak, 2013), we gathered 72M English language tweets, from 1.2M users. Each tweet was canonicalised as follows: tokenised; made lowercase; usernames replaced by a uniform token; URLs replaced by a uniform token. We randomly selected 1000 users with 1000-2000 tweets to train and test a classifier. On the remaining tweets, we trained a 10-gram word length model, and a 5-gram language model, using SRILM (Stolcke, 2002) with Kneyser-Ney smoothing. For vocabulary, the language model was given every word in the PPDB, and every word in the set of tweets, including the removed tweets. We do not expect this to provide an unrealistic advantage to the censor: any word not in the PPDB cannot hide information.

We randomly took 10 users from the set of 1000, and produced 100 stego tweets for each, for three payload sizes: 1, 2 and 4 bits. The stego tweets were generated using CoverTweet, with a human operator selecting the most fluent option containing a desired (randomly generated) payload. If there were no fluent options, the tweet was skipped. If the tweet already contained the desired payload (if the hash value of the tweet already matched the message), it was left unchanged. We refer to this data as Manual CoverTweet (M-CT).

Due to the expensive nature of generating data with a human judge, and to assess the value of the human in-the-loop, we also automatically generated 1M stego tweets, by embedding data in 1000 tweets for each of the 1000 users. The same three payload sizes were used. Here, the tweet with the highest probability (provided by the language model and the PPDB) was selected. Again, the tweet was left unchanged if the tweet already contained the desired payload. Tweets were only skipped if there were no options with the correct payload. We refer to this as Automatic CoverTweet (A-CT). Finally, we embedded 1 bit in the same tweets using T-Lex, rejecting any that contained no words in the T-Lex dictionary; the result was approximately 100 tweets on average per user.

We split the users in half, training a linear ensemble classifier (Kodovskỳ et al., 2012) (designed to work with large feature sets for steganalysis) on feature instances from one half, testing it on the other. We only classified tweets by users for whom the classifier had no prior knowledge. All error rates are averaged over 10 different random user splits. For the M-CT data, where we have a much smaller set of data, we performed 10-fold cross validation, leaving one user out for testing each fold.

In each experiment we matched training and testing data: the training data was produced by the

[4]Where we say a word or phrase is 'in' the PPDB, we mean it features in one or more paraphrase rules that could have been applied to the original tweet.

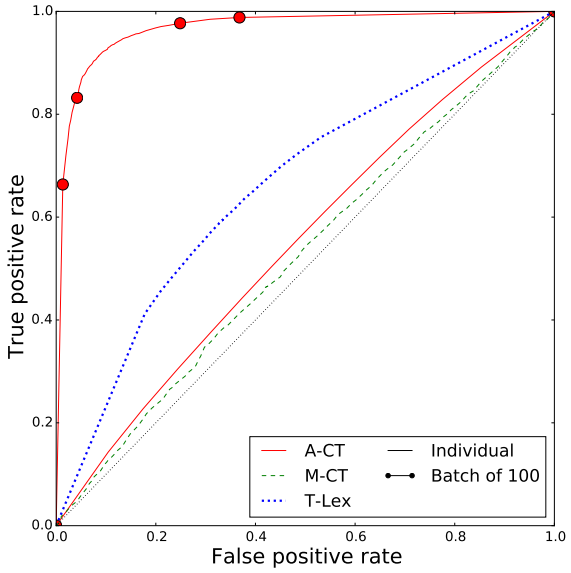

Figure 1: ROCs for each embedding method, for individual tweets or batches containing 1 bit.

	Batch size	AUC
A-CT	1	0.551
M-CT	1	0.509
T-Lex	1	0.667
A-CT	100	0.9631

Table 1: AUC values for the ROCs shown in Figure 1

same method as the testing data. For M-CT we tried an alternative scheme, by training on A-CT data, but testing on M-CT; this did not work reliably, and the accuracy of the resulting model went down as pooled batch size increased. The exploration of this phenomenon is left for later work.

5.2 A note on unchangeable covers

Some tweets cannot hide the payload, either when there are no paraphrase options for that hash, or when the human (for M-CT) vetoes all the options. This is the *non-shared selection channel* problem in steganography. Methods such as *syndrome trellis codes* (Filler et al., 2011) allow the unchangeable cover elements to be sent without compromising the message. These have not been applied to linguistic steganography, but we simulate their use: we remove the unchangeable tweets from the cover and stego sets, essentially giving the steganographer and detector the ability to ignore such tweets.

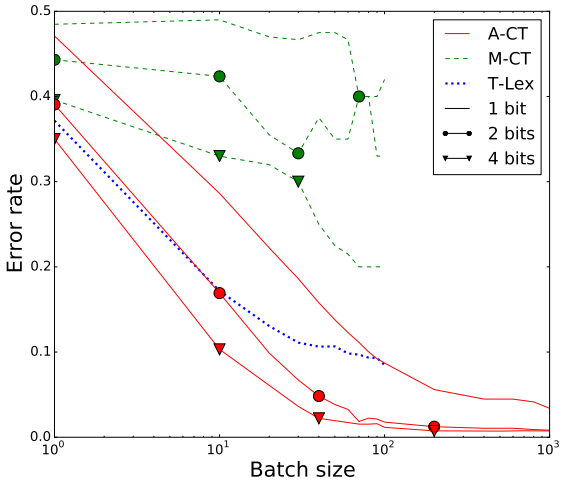

Figure 2: The effect of batch size on error rate. Batches of 100 are the maximum for M-CT and T-Lex, but we go up to 1000 with A-CT.

5.3 Results

As expected, the performance of the classifiers trained on individual tweets is poor (see Figure 1). In particular, the models trained on A-CT and M-CT data have very low accuracy on data with 1 bit payload, performing only slightly better than random guessing. T-Lex tweets prove slightly easier to detect. This does not mean that the systems are secure however. Though we cannot identify individual stego tweets, when we pool evidence we find we are able to train a model that can identify active users with high accuracy, for all data sets.

Figure 2 shows the change in error rate as the batch size (the number of tweets we pool) is increased. We can clearly see that increasing batch size improves accuracy. We also see that increasing payload size makes active users easier to spot. This is unsurprising: CoverTweet is forced to choose from fewer options when payload increases.

The experiment showed us that M-CT is the most secure stegosystem, though not immune to the effects of pooling. When combining features from 100 tweets, the classifier had an error rate of 0.21 on 4 bit M-CT data; data generated in the same way, with the same payload, was previously shown to be secure against human judges. With large batches, detection of A-CT is almost perfect.

To establish which class of features had the biggest effect on detection, we trained models on each combination of features described in Section 4. A subset of these combinations are shown

	b	n	p	w	b+n	n+p	p+w	b+n+p	p+n+c	all
A-CT (2 bits)	0.345	0.311	0.217	0.376	0.266	0.204	0.203	0.180	0.191	0.169
T-Lex (1 bit)	0.448	0.226	0.273	0.317	0.226	0.202	0.218	0.198	0.168	0.168

Table 2: Error rates for models trained on different combinations of feature class, for a batch of 10. The feature sets are as follows: basic (b), n-gram (n), PPDB (p) and word length (w).

in Table 2, for A-CT and T-Lex.We can see that for A-CT, the PPDB features easily outperform others, with an error rate of 0.217 when used alone; the warden's knowledge of the system is powerful. The second best is the n-gram set, though it performs better on T-Lex than on A-CT. This is most likely due to CoverTweet's use of the language model in the embedding stage: the system is attempting to minimise the distortion that these features are looking for. T-Lex has no such distortion measure, leaving it open to this sort of attack.

Basic and PPDB feature sets fare worse on T-Lex. The basic features are aimed at changes in word count and stop word usage: neither of these are affected by T-Lex substitutions. The PPDB features are at a disadvantage with T-Lex, as they are designed for CoverTweet. If T-Lex's data source were used instead of CoverTweet's, the performance would likely improve significantly.

The combination of PPDB and n-gram features on T-Lex gives us some interesting insight: despite the mismatch of substitution source, we still see an improvement over the n-gram features used on their own. This suggests that the warden does not need the exact data source as the steganographer for these features to be useful.

6 Conclusion

It was believed that linguistic steganography was weak against humans, but CoverTweet disproved it. We have shown that individual stego objects are seemingly also strong against statistical attacks. However, by pooling multiple pieces of evidence against a user, the warden can drastically improve detection rate. With each steganographic tweet sent, the user creeps closer to being caught. This is the first steganalytic classifier, in any domain, that successfully exploits pooled evidence. The design of steganographic systems must now take this type of attacker into account. It would interesting to determine whether human judges are capable of pooling large amounts of scant evidence: we conjecture not.

Results suggest that detection is improved by utilising rich-feature models; we only scratched the surface with regards to this. There are many avenues to explore, such as using multiple language models from which to extract features (this is the analogue of filter banks used in contemporary image steganalysis (Fridrich and Kodovskỳ, 2012)).

The security of a system should first be measured against a powerful (informed) attacker. We played this role by using features extracted using exact knowledge of the CoverTweet system (the PPDB features); this class of features was particularly effective against CoverTweet. The system should now be evaluated against a weaker attacker. We have seen that detection is still reliable when the warden knows the wrong system (T-Lex), but further experiments are required to determine exactly how detection rate is affected by mismatches in paraphrase sources, or language model.

References

Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Ching-Yun Chang and Stephen Clark. 2010. Linguistic steganography using automatically generated paraphrases. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 591–599. Association for Computational Linguistics.

Ching-Yun Chang and Stephen Clark. 2012a. Adjective deletion for linguistic steganography and secret sharing. In *COLING*, pages 493–510.

Ching-Yun Chang and Stephen Clark. 2012b. The secret's in the word order: Text-to-text generation for linguistic steganography. In *COLING*, pages 511–528.

Mark Chapman, George I Davida, and Marc Rennhard. 2001. A practical and effective approach to large-scale automated linguistic steganography. In *Information Security*, pages 156–165. Springer.

Zhili Chen, Liusheng Huang, Haibo Miao, Wei Yang, and Peng Meng. 2011. Steganalysis against

substitution-based linguistic steganography based on context clusters. *Computers & Electrical Engineering*, 37(6):1071–1081.

Le Chen, Chi Zhang, and Christo Wilson. 2013. Tweeting under pressure: analyzing trending topics and evolving word choice on sina weibo. In *Proceedings of the first ACM conference on Online social networks*, pages 89–100. ACM.

Tomáš Filler, Jan Judas, and Jessica Fridrich. 2011. Minimizing additive distortion in steganography using syndrome-trellis codes. *Information Forensics and Security, IEEE Transactions on*, 6(3):920–935.

Jessica Fridrich and Jan Kodovskỳ. 2012. Rich models for steganalysis of digital images. *Information Forensics and Security, IEEE Transactions on*, 7(3):868–882.

Jessica Fridrich. 2009. *Steganography in digital media: principles, algorithms, and applications*. Cambridge University Press.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *HLT-NAACL*, pages 758–764.

Michael Grosvald and C Orhan Orgun. 2011. Free from the cover text: a human-generated natural language approach to text-based steganography. *Journal of Information Hiding and Multimedia Signal Processing*, 2(2):133–141.

Andrew D Ker. 2007. Batch steganography and pooled steganalysis. In *Information Hiding*, pages 265–281. Springer.

Jan Kodovskỳ, Jessica Fridrich, and Vojtěch Holub. 2012. Ensemble classifiers for steganalysis of digital media. *Information Forensics and Security, IEEE Transactions on*, 7(2):432–444.

Peng Meng, Yun-Qing Shi, Liusheng Huang, Zhili Chen, Wei Yang, and Abdelrahman Desoky. 2011. Linl: Lost in n-best list. In *Information Hiding*, pages 329–341. Springer.

George A. Miller. 1995. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41.

Todd Mostak. 2013. Harvard TweetMap. accessed October 2013. `https://worldmap.harvard.edu/`.

Lip Y Por, TF Ang, and B Delina. 2008. Whitesteg: a new scheme in information hiding using text steganography. *WSEAS Transactions on Computers*, 7(6):735–745.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. pages 901–904.

Cuneyt M Taskiran, Umut Topkara, Mercan Topkara, and Edward J Delp. 2006. Attacks on lexical natural language steganography systems. In *IS&T/SPIE Electronic Imaging*, pages 120–129. International Society for Optics and Photonics.

Olga Vybornova and Benoit Macq. 2007. Natural language watermarking and robust hashing based on presuppositional analysis. In *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on*, pages 177–182. IEEE.

Alex Wilson, Phil Blunsom, and Andrew D Ker. 2014. Linguistic steganography on twitter: hierarchical language modeling with manual interaction. In *IS&T/SPIE Electronic Imaging*, pages 201–217. International Society for Optics and Photonics.

Keith Winstein. 1998. Lexical steganography through adaptive modulation of the word choice hash. `http://www.imsa.edu/~keithw/tlex`. *Unpublished.*

Lingyun Xiang, Xingming Sun, Gang Luo, and Bin Xia. 2014. Linguistic steganalysis using the features derived from synonym frequency. *Multimedia tools and applications*, 71(3):1893–1911.

Sui Xin-guang, Luo Hui, and Zhu Zhong-liang. 2006. A steganalysis method based on the distribution of characters. In *Signal Processing, 2006 8th International Conference on*, volume 4, pages 54–56. IEEE.

Zhenshan Yu, Liusheng Huang, Zhili Chen, Lingjun Li, Xinxin Zhao, and Youwen Zhu. 2009. Steganalysis of synonym-substitution based natural language watermarking.