Detecting Steganography

Marie-Jeanne Lagarde (236512)
Machine Learning and Optimization Laboratory, Ecole Polytechnique Fédérale de Lausanne, June 2018

***Abstract*—The aim of this project is to assess the non-detectability of slightly modified version of the linguistic steganography approach described in the paper Generating Steganographic Text with LSTMs [1] by classifying whether a tweet was generated by a regular user or by this steganographic method. For that purpose, both a fastText supervised classification and a user study are conducted.**

I. Introduction

The main advantage of steganography over cryptography alone is that it conceals the fact that a secret message is being sent and does not draw attention to itself, thus avoiding decryption attacks. As a consequence, using a steganographic method appears to be meaningful only if the hidden message remains undetectable against all possible adversaries. Using a slightly modified version of the linguistic steganography approach described in the paper Generating Steganographic Text with LSTMs [1] which hides secret messages in tweets, this project assesses the non-detectability of such a method. Non-detectability of a linguistic steganography method might seem relatively easy to evaluate for human operators: a message either looks human written or not. However, tweets are more complex to analyze as those short messages often contain spelling mistakes and more than 40% are classified as small talk [2]. Further, newly available efficient machine learning classification algorithms raise the question of whether a machine can tell if a sentence has been written by a human or not, thus compromising our steganographic model and automatizing the detection of embedded secret messages. This report is organized as follows: Section II sums up the slightly modified steganographic approach; Section III deals with the use of a machine learning classification algorithm to detect our steganographic method; Section IV summarizes the user study conducted; Section V explores possible improvements. Hereafter we define a *Fake* tweet to be one in which a secret message has been embedded by our linguistic steganography system and a *Real* tweet to be one which has been written by a real user on twitter.

II. Chosen Steganographic Approach

We choose for this project to assess the non-detectability of a slightly modified version of the linguistic steganography approach described in the paper Generating Steganographic Text with LSTMs [1] which we explain below.

A. Description of the Implementation

The general idea of this steganographic technique is for the sender to generate a chain of tweets which hide a secret message that only the receiver can decode thanks to a shared secret seed. Each character of the secret message that the sender wants to transmit is mapped to its ASCII representation which forms a bit string. We implement a Pytorch [3] recurrent neural network trained on a set of 75,291,137 tweets which vocabulary is equally split among n bins. Each $\log_2 n$ bits of the secret message bit string is then mapped to the corresponding bin in which the recurrent neural network selects the most probable word (the neural network being forced to pick a word from a particular bin will be designated hereafter as *under constraint*). The bins are generated randomly depending on the shared seed. These consecutive picks form several tweets that hide the secret message. At the receptor, each word is mapped back to its original bin and the original bit string is retrieved thanks to the shared seed. Finally, the bit string is mapped back to the original ASCII characters and the message is decoded and displayed at the receptor. Usernames are anonymized and being replaced by <user> and urls are being replaced by <url>.

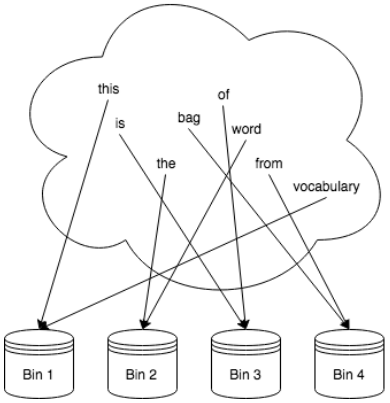

Fig. 1: Bins Scheme with replication factor 1

B. Improvement to the Original Design

In order to improve the quality of the produced tweets, we focus on three elements: the punctuation, the end of string signal <eos> and the capitalization. We also look into how to speed up the tweet generation. After several trials, it appears that the quality (based on human judgment) of the tweets improves when a word aggregated with a punctuation mark is considered as a single entry in the dictionary. For instance, for the tweet "<user> : New post: "Pontoon reflection" <url> <eos>", our dictionary contains the word "post:" rather than two separated words "post" and ":". Second, the usual behavior of our steganographic method described before is to stop when the secret message is totally embedded in the chain of tweets, even when an end of string symbol (<eos>) has not yet been reached. However, a non-finished tweet is easily recognizable by both a human operator and a classifier. As a consequence, we modify the original approach by creating an additional bin in which the neural network picks words only when it has finished embedding the secret message and this until it picks an end of string word. At the receiver,

words from this special bins are not decoded and this change does not affect the decoding of the secret message while it enhances the quality of the produced tweets. Third, the original method only produced lowercase tweets making it easily detectable and not realistic. We train the new model on capitalized tweets, where words with different capitalization are considered being different (for instance, "cat", "Cat" and "CAT" are different entries in our dictionary). This allows our neural network to produce tweets combining both lowercase and uppercase letters. Finally, the module Pickle is used to save and reload the corpus and the bin distribution depending on the chosen seed, allowing a large speed when generating tweets.

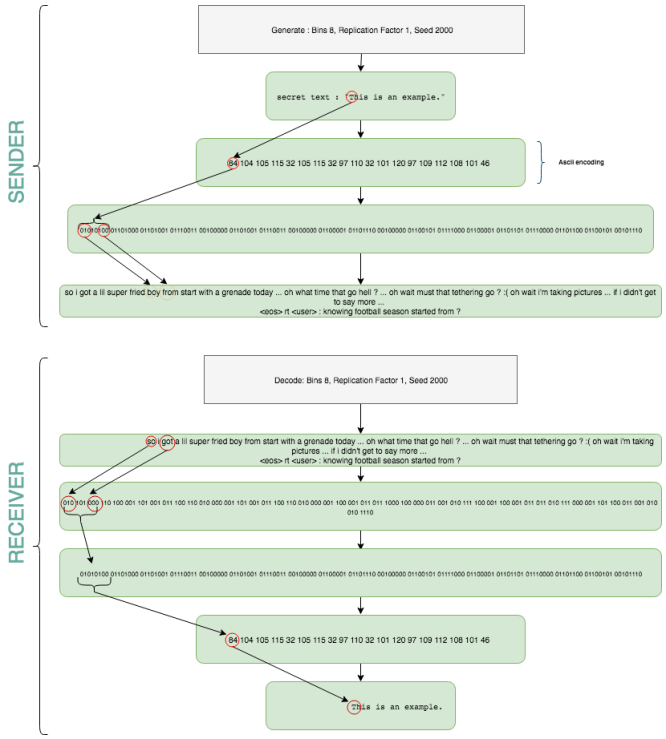

Fig. 2: Encoding Decoding Example

C. Hyperparameters Tuning

To produce the less detectable steganographic model possible, we search for the best hyperparameters to train the recurrent neural network (number of hidden layers, number of layers, number of epochs) and to generate tweets (number of bins, replication factor, number of common tokens and temperature). For each model, a set of tweets hiding a secret message (here after called *Fake* tweets) is generated and given to train the classifier along with the same number of existing tweets preprocessed in the same way (here after called *Real* tweets). The model selected is the one with the lowest result of well classified *Fake* tweets, in other words, the model for which fewer generated tweets are detected as embedding a secret message. Among twenty-two combinations of hyperparameters, the combination of following ones gives the most satisfying results (as described in section III subsection C) while keeping the ability to decode the original secret message: 600 hidden layers, 3 layers, 6 epochs, 2 bins, a replication factor of 1, 600 common tokens, a common bin factor of 2 and a temperature of 1.

III. Detecting Steganography Using a Binary Classifier

Machine learning classification algorithms for detection and analysis of steganographic embedded content in images are known to provide reasonable discrimination schemes [4], but researches about classification on steganographic embedded content in texts still shows limited results as most researches on text steganography are based on statistical approaches. The following part explores the uses of a supervised binary classifier to detect embedded content in tweets, in order to answer the question: can a machine tell if a tweet has been written by a human or not and thus compromise our steganographic method?

A. Choice of the Classifier

This project requires an efficient supervised binary classifier which should also be fast to train and to evaluate. Looking into machine learning classifying algorithms, fastText [5] appears to be a good candidate as it is designed for efficient learning of word representations and sentence classification. Further, it has shown significant results in text classification and is often on par with deep learning classifiers in terms of accuracy, and is many orders of magnitude faster for training and evaluation which makes it the best choice for our project.

B. Classifier Model Training

As described in section II subsection C, for each steganographic model, we build a set of 60000 tweets which contains 50% of our generated tweets (*Fake*) and 50% of existing tweets (*Real*) preprocessed in the same way. This set is then split: 85% goes into the training set and 15% goes into the validating set used for the supervised learning. The most accurate classification was obtained when tuning the hyperparameters of the fastText classifier with 35 epochs. Note that we do not look into bi-gram or tri-gram during this project, even though it could improve the classification results.

C. Results

Regarding the results, we are interested in selecting the steganographic model for which the classification of *Fake* tweets is the less accurate. It means that we select the model for which our classifier misclassifies the most *Fake* tweets as being *Real* ones. Indeed, we would like our classifier not to be able to decide whether a tweet comes from our steganographic model (*Fake*) or from a real user (*Real*). As a baseline, we evaluate the classification of tweets generated by our LSTM model with no steganographic approach (no constraints on bins for embedding a secret message), which is displayed in the first line of Table I. We believe that adding our steganographic approach to the LSTM, which constrains the choice of the next words to a specific bin, will make our generated tweets more detectable and thus our classifier would achieve a higher score. Table I confirms this hypothesis and presents the less accurate classification rates obtained. The lowest detection rate achieved for our generated tweets by the classifier when we want the message to be decodable at the receiver (number of bins being different from replication factor) is 0.80 as shown in Table I.

Temperature	Bins	Common bin factor	Number tokens	Replication Factor	Overall Classification	Classification of Fake Tweets Only
1	-	-	-	-	0.666	0.662
1	2	2	600	1	**0.697**	**0.80**
1	4	2	600	1	0.678	0.82

TABLE I: Less accurate classification rates

D. Possible Causes of Detection

As fastText classifier is not explicit about why it decides to label a given tweet as a *Fake* or *Real*, we make here a few hypothesis about possible detection causes. First of all, the length of the tweets appears to be a potential cause of detection. For that purpose, we try to design our steganographic model in a way that both median and average length of produced tweets equal those of real tweets from our LSTM training set. The temperature of the recurrent neural network also seems to have a large impact on the length of the tweets. By limiting the length of tweets to 120 symbols for both *Fake* and *Real* tweets, we reduce by 2% the overall accuracy of the classification but increase of 1% the accuracy of the classification of only the *Fake* tweets. As a consequence we do not apply this length limitation to our final model. Second, the frequency of retweets "RT" and url "<url>" varies a lot depending on our hyperparameters. This might have an influence on the classification and we thus need to ensure that both word appearance frequencies are close to the one of the real tweets from our LSTM training set.

IV. User Study

To assess the non-detectability of our steganographic method against human operators, we conduct a user study. This study should answer the following question: is a human operator able to differentiate *Real* tweets from *Fake* tweets generated by our steganographic method ?

A. Condition of the User Study

To conduct the study, we build 3 different sets of 40 tweets which contains 20 randomly selected *Fake* tweets generated by our most efficient steganographic model (temperature 1, 2 bins, a common bin factor of 2, 600 tokens and a replication factor of 1) and 20 randomly selected *Real* tweets (from the validation set of our LSTM model). The study was conducted on 28 men and 21 women aged from 19 to 63 years old. Each of them is asked to decide on a Google form whether a given tweet is *Fake* (generated by our steganographic method) or *Real* (written by a real user on twitter). From the answers, we compute the overall accuracy of the classification and the accuracy of the classification of only the *Fake* tweets.

B. Results

The goal of the user study is to assess the non-detectability of our steganographic method against a human operator. As we want our system to be non-detectable, which means that a human operator should not be able to decide whether a tweet hides a secret message (*Fake*) or was written by a regular twitter user (*Real*), the ideal results are the following: an accuracy of 50% on each set and on each *Fake* tweet set (sets that contain only the tweets generated by our steganographic method). Table II describes the results of the study. The average accuracy of the classification task (mean of the results of the 3 different sets submitted to participants) is 60% and the average accuracy of the classification when only looking at *Fake* tweet sets is 49%. These results appear to be close to the ideal ones defined above. The accuracy of the classification of each *Fake* tweets ranges from 87% accuracy for "<user> : 4 people followed me and 4 % by // by <url> " to 0% accuracy for "<user> : I'm giving up...". That highlights the quality of some tweets generated by our linguistic steganography system with hope for improvement as for some generated tweets the detection rate remains high.

Study Number	Overall Classification	Classification of Fake Tweets Only
1	0.5623161765	0.4141544118
2	0.6476041667	0.5420833333
3	0.5890625	0.5125
Average	0.59966094	0.48957925

TABLE II: Classification rates of the user study

C. Limitation of the user study

There are a few limitations in the user study we conduct that must be noticed. First of all, the number of participants remains small. Second, not all the participants are active users of twitter, which can introduce bias in the study as they are not used to the writing style. Finally, some tweets (both *Real* and *Fake* ones) are written in Spanish which is a language that not all participants master.

V. Suggested Improvements of the Steganographic Method

In this section, we explore a few solutions which could improve our linguistic steganography approach in order to reduce its detectability.

A. Model Improvement

The LSTM model used without any constraint (no constraint on bins for picking the next word) lowerbounds the achievable classification rate. By using a model which generates tweets that are less detectable when classified with real tweets, we would decrease the chance for our steganographic embedding to be detected. This would be possibly achievable with a longer training or a bigger dataset, with the goal to reach a 50% probability of our generated tweets to be classified as *Real* or *Fake* by both the classifier and human operators. Further, an improvement of the model would also allow the frequency of "RT" and "<url>" along with the mean and median average length to be closer to the one of the LSTM training set. However, due to computational power and time limitations, we do not consider this option in the scope of this project.

B. Automatization of Username

Considering an end-to-end application for secret communication purpose, an automatization script replacing "<user>" by a real twitter username and "@user" by a randomly chosen username among the accounts that the given username follows would make this application more realistic.

C. URL replacement

To handle the "<url>" replacement, a simple algorithm which randomly picks a website among a list of most popular websites worldwide could be set up. However, this simplistic approach would lead to the presence of irrelevant urls in the generated tweets. Two approaches are possible: either leave

the urls when training and consider them as words (entry in the dictionary), or replace the "<url>" word thanks to a script that exploits the generated tweets contexts to select a relevant url. Both options provide an url replacement process that would fit with the context of the generated tweet.

D. Generating Thematic Models

Considering again an end-to-end application for secret communication purpose, this system could be optimized by creating thematic models. The user would select a theme when embedding his secret message and generate a chain of consecutive tweets about this subject of his interest with his pseudonym, allowing the human adversary to be more easily fooled.

VI. Conclusion

In this project, we explore the detection of a slightly modified approach of the steganographic method described in the paper Generating Steganographic Text with LSTMs [1]. We conduct both a fastText classification and a user study to classify tweets as either *Fake* or *Real*, where a *Fake* tweet is one in which a secret message has been embedded by our linguistic steganography system and a *Real* tweet is one which has been written by a real user on twitter. We obtain an overall classification accuracy of 60% for the user study and of 70% for fastText algorithm. We further look into the classification of only the *Fake* tweets as we are interested in knowing if an adversary that captures a tweet which embeds a secret message can detect whether that it is a *Fake* one and thus compromise the steganographic process. We obtain a classification accuracy of 49% for the user study and of 80% for the fastText classification on the *Fake* tweet set. When comparing the two detection methods, it appears that our steganographic approach is hardly detectable by a human operator but still highly detectable by a machine learning classification algorithm. As a consequence, we offer some guidelines for critical thinking and improvements that will most likely decrease the detection rate of our steganographic embedding by a supervised classifier.

References

[1] Katerina Agyraki Tina Fang, Martin Jaggi. Generating steganographic text with lstms, 2017.

[2] Ryan Kelly. Twitter study august 2009, 2009.

[3] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[4] GUJAR SUJIT PRAKASH. Measures for classification and detection in steganalysis. http://clweb.csa.iisc.ernet.in/sujit/docs/ME_thesis.pdf, 2006.

[5] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.