DEGREE PROJECT IN COMPUTER ENGINEERING,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2017

Detecting Sockpuppets in Social Media with Plagiarism Detection Algorithms

FREDRIK ALBREKTSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

DEGREE PROJECT IN COMPUTER ENGINEERING, FIRST CYCLE
STOCKHOLM, SWEDEN 2017

Detecting Sockpuppets in Social Media with Plagiarism Detection Algorithms

Fredrik Albrektsson

Degree Project in Computer Engineering, DD142X
Supervisor: Iolanda Leite
Examiner: Örjan Ekeberg

KTH, CSC, 2017-06-05

Abstract

As new forms of propaganda and information control spread across the internet, the need for novel ways of identifying them increases as well. One increasingly popular method of spreading false messages on microblogs like Twitter is to disseminate them from seemingly ordinary, but centrally controlled and coordinated user accounts – sockpuppets. In this paper we examine a number of potential methods for identifying these by way of applying plagiarism detection algorithms for text, and evaluate their performance against this type of threat. We identify one type of algorithm in particular – that using vector space modeling of text – as particularly useful in this regard.

Identifikation av Strumpdockor inom Social Media med Plagiatkontrollalgoritmer

Abstrakt

Allteftersom nya former av propaganda och informationskontroll sprider sig över internet krävs också nya sätt att identifiera dessa. En allt mer populär metod för att sprida falsk information på mikrobloggar som Twitter är att göra det från till synes ordinära, men centralt kontrollerade och koordinerade användarkonton – på engelska kända som "sockpuppets". I denna undersökning testar vi ett antal potentiella metoder för att identifiera dessa genom att applicera plagiatkontrollalgoritmer ämnade för text, och utvärderar deras prestanda mot denna sortens hot. Vi identifierar framför allt en typ av algoritm – den som nyttjar vektorrymdsmodellering av text – som speciellt användbar i detta avseende.

Table of Contents

1 Introduction

As those who spread propaganda, half-truths, and so-called fake news find new ways of using the internet to send out their message, the ability to identify the tricks of their trade becomes increasingly important. One method often used on microblogging platforms such as Twitter is signal boosting [1] of a message, to ensure that a given message will be spread as far and as wide as possible with the help of sockpuppet accounts [2]. This is generally done by posting an identical or nearly identical message from several user accounts controlled by a single user. By automatically correlating posts from suspicious user accounts and analyzing their message contents it should be possible to determine if they are controlled by a common user. One way of performing this analysis is by applying the same algorithms as those used in detecting plagiarism in written texts on the messages from these user accounts.

1.1 Problem Statement

The purpose of this paper is to determine if it is possible to use plagiarism detection algorithms to determine if two Twitter accounts are colluding to spread a message. In other words, we seek to see if algorithms used to detect a high level of similarity between texts can be used to cross-check tweets collected from two different accounts and find identical or near-identical messages being sent out. We also do not want it to generate a large number of false positives, i.e. flagging message pairs as being similar when they are not. To be considered a success we require that an algorithm must be able to correctly identify 75% of similar message pairs, and generate no more than 20% false positives.

1.2 Scope and Constraints

This work is limited to attempting to apply plagiarism detection algorithms to messages written in the English language. We try to cover a variety of methods and algorithms commonly used in plagiarism detection, but limit ourselves to testing only the most common in this paper.

2 Background

In this section we outline some background on the subject matter to explain what we're dealing with and why this research is necessary. We then highlight existing pertinent research and discuss how it relates to our task at hand.

2.1 Twitter

Launched in 2006, the microblogging site Twitter currently boasts over 300 million active users and sees more than a billion unique monthly visits to sites with embedded Tweets. With offices worldwide and 79% of their users outside the United States, the platform has become a major social networking player on par with sites such as Facebook and Reddit. Users are limited to 140 character posts called Tweets, which are visible to their followers and can be re-tweeted for further spread. Posts are said to *go viral* when they are repeatedly passed on in this manner, or when the sentiment expressed becomes a talking point across a large section of a community.

2.1.1 Twitter API

Twitter exposes a number of functions to external developers via their APIs (application programming interfaces) including a REST API [12] accessible over HTTP. An API is generally a set of protocols and tools for developers to interface with and make use of an existing program or service. The Twitter API can be used with or without login information; developers with a login can access all the actions of a user logged in to the Twitter web interface (posting messages, reading profile data, and so on), while those without a login can perform actions that do not require one (reading messages from unprotected accounts, etc.). Some rate limits apply, such as how many messages can be collected in a given timeframe.

2.2 Propaganda, Troll Farms & Fake News

While propaganda has existed in many forms over the years, a key element often considered necessary for success is the repetition of an idea. Signal boosting [1] of messages on social media is one way to create this kind of repetition, ensuring both the spread of an idea and that it keeps appearing for other users.

An application of this can be seen in Adrian Chen's article from The New York Times [3], in which he documents a series of disinformation campaigns originating out of a troll farm in St. Petersburg with a focus on pro-Kremlin propaganda and falsifying news about terrorist attacks abroad. While the term *internet troll* originally meant someone who posted provocative messages on internet message boards to cause polarization and infected arguments, a troll farm such as the one discussed by Chen is a corporate propaganda machine with employees coordinating social media posts and falsified news coverage. In the article he further describes it as follows:

> *The agency had become known for employing hundreds of Russians to post pro-Kremlin propaganda online under fake identities, including on Twitter, in order to create the illusion of a massive army of supporters; it has often been called a "troll farm."*

All fake news do not come from these farms however, nor are the troll farms and creators of fake news themselves limited to Russia. The US presidential election of 2016 saw an upswing in the use of the expression, e.g. with news outlets like NPR [4] publishing articles about one creator of fake news using them as a political tool against those who don't fact-check.

2.3 Plagiarism Detection

With the ever-increasing sophistication of plagiarism employed by students, researchers and in various forms of industries, an increasing need for methods to detect plagiarism is likewise needed. Most schools and universities apply some form of plagiarism detection for text when managing student assignments, with systems such as Turnitin, Urkund, Plagiserve and the like comparing texts and detecting similar passages that warrant closer inspection.

The most common approach (for text comparison) is to compare the entire contents of documents, analyzing the content for example with a word-by-word comparison, by way of fingerprinting or with statistical measures such as word frequency. More advanced solutions such as calculating the Euclidean distance between two documents can also be applied. Details on these are discussed in the following sections on related work.

2.4 Related Work

This subsection discusses existing research in three main areas: rudimentary techniques for detecting plagiarism, some more advanced algorithms to do the same, and how to work with lexical data from microblogs such as Twitter.

2.4.1 Plagiarism Detection Techniques

A variety of algorithms are used in various existing software to assess whether a document is likely to be plagiarized wholly or in part. Some key solutions discussed by Maurer et al. [5] are word-by-word comparisons between documents, identifying and searching for characteristic paragraphs or phrases, and style analysis.

A word-by-word analysis algorithm may use direct comparisons between texts or it can use fingerprinting in a manner similar to that originally suggested by Manber [6]. In the latter case snippets of text taken from one document are used as fingerprints, which are then searched for in the second document. The presence of a fingerprint in both texts thus indicates that the texts share some similarities. Note that the original method was not designed only for texts, and instead operated on raw bytes by selecting anchors (points of reference) and picking some amount of data surrounding these as fingerprints. One method of choosing these fingerprints before searching for them for them in the second document is to selectively pick them based around some unique feature of the text, while another method is to generate fingerprints for all possible substrings in a text and selecting a subset of these. As Manber points out both methods have their advantages and disadvantages. The former requires the anchors or fingerprints to be adjusted to the text to some degree, and may be more or less effective depending on language used or even writing style of the author. The latter, fingerprinting all possible substrings in a text, does not allow for any adjustments at all, and may cause bad fingerprints to be chosen due to the nature of randomized substring selection.

Maurer et al. [5] point out that many researchers have improved on Manber's work, some by utilizing online search tools such as Google to cross-reference a fingerprint not only with local databases but also all those indexed by these search engines.

Searching for characteristic phrases is a solution that has been automated by Niezgoda & Way [7] but still requires a relatively large amount of manual work. A phrase related to the key concepts of the paper is chosen and, when done manually, input to a search engine such as Google to find any verbatim uses of the same phrase in other papers or articles. The software developed by Niezgoda & Way allows for the search to be automated, but still requires a manual selection of this characteristic phrase by the examiner. As this method relies on searching for the phrase verbatim, successfully finding it elsewhere shows that the paper examined is indeed plagiarized. The drawback is however that it will only detect copy-paste plagiarism, meaning any use of paraphrasing, reordering words or other techniques that change the wording of the phrase without affecting the meaning of it will bypass these detection methods.

In discussing style analysis of an author, or stylometry, Maurer et al. refer to the work of Eissen & Stein [8] who in turn identify five main categories of stylometry that can be applied to detect plagiarism. Stylometry itself is the application of analysis within a text or between several texts by the same author to identify unique traits that, when suddenly missing or replaced in portions of a document, suggest that passage is likely to have been written by another author.

The categories of stylometric features identified by Eissen & Stein [8] were broken down into five categories:

- Text Statistics, operating at the character level
- Syntactic Features, measuring writing style on a sentence level
- Part-of-Speech Features, quantifying the use of word classes
- Closed-Class Word Sets, counting special words
- Structural Features, reflecting text organization

Expounding on these categories, Stein et al. [9] mention the strengths and limitations related to them. Structural features such as average paragraph length, for example, requires a considerably longer passage than lexical features such as average word length.

2.4.2 Advanced Algorithms for Plagiarism Detection

Separate from the main subject of detection methods, Maurer et al. [5] briefly discuss some of the more sophisticated techniques available for plagiarism detection. A mathematical approach to pattern recognition, using a vector space model to determine angular similarity (or cosine similarity) between two texts is given as an example, calculating a similarity measure using the following formula:

$$S_{cos}(A,B) = \frac{VectorA \cdot VectorB}{|Vector\ A||Vector\ B|}$$

Where the length of a vector is calculated in the regular fashion:

$$|Vector\ \bar{\upsilon}| = \sqrt{Vector\ \bar{\upsilon} \cdot Vector\ \bar{\upsilon}}$$

This does however require cleaning up a text to remove common words like pronouns and prepositions, and Maurer et al. noted at the time of their writing that given a large vector space this solution quickly becomes impractical to use in computations.

Generating the vectors themselves is done by building a vocabulary from both texts being compared, and then creating a numeric vector representing each message such that each occurrence (in that message) of the ith word of the vocabulary is represented by incrementing the ith value of the vector. For clarity, let us take an example of two texts and convert them to vectors:

Text A: “Not a system to censor content”

Text B: “But a system to create context”

Text A	Text B	Vocabulary
not	but	a
a	a	but
system	system	censor
to	to	content
censor	create	context
content	context	create
		not
		system
		to

This results in Vector A = {1, 0, 1, 1, 0, 0, 1, 1, 1} and Vector B = {1, 1, 0, 0, 1, 1, 0, 1, 1}.

Still more advanced solutions are discussed by Bao et al. [10] for calculating symmetric and asymmetric similarity, with one example being a variation on the angular distance formula above. It takes word weights into account as follows:

$$S_{cos}(A,B) = \frac{\sum_{i=1}^{n}[\alpha_i^2 \times F_i(A) \times F_i(B)]}{\sqrt{\sum_{i=1}^{n}[\alpha_i^2 \times F_i^2(A)] \times \sum_{i=1}^{n}[\alpha_i^2 \times F_i^2(B)]}}$$

where α_i is the word weight vector, $F_i(A)$ & $F_i(B)$ are the frequency of the ith word in the respective documents and n is the length of the text (i.e. the number of words).

2.4.3 Lexical Normalization

Peculiar to Twitter are certain abbreviations, special characters and writing styles that come from the 140-character limit imposed on posts and the tagging features available. Tags using hashes (e.g. #FakeNews) are commonly used to mark the post as belonging to a topic, as is tagging or addressing other users with an @-sign (e.g. @POTUS for the account of the President of the United States).

This makes Twitter messages a special case in natural language processing, and they generally require some cleaning up to work well with natural language processing algorithms. For this type of lexical normalization we looked at the work of Bilal Ahmed [11] who breaks the process down into a few steps. Each message is separated into words (tokens), with each token matching a pre-existing wordlist being considered a correctly spelled word where no normalization is necessary. Tokens that do not match any known word is passed through regular expression (regex) filters to identify the presence of Twitter-specific characters such as hashes, which if present also mark the token as not needing normalization.

Remaining tokens are then assumed to be misspellings or shortened forms of dictionary words and normalized according to a series of techniques such as Levenshtein distance, Soundex technique etc. As these specific solutions are outside the scope of what we will use in this paper we will not discuss their specifics here.

2.5 Conclusion

A number of techniques and algorithms are available for detecting similarities in text, each with their own strengths and weaknesses related to the short messages found in microblogs like Twitter.

The use of characteristic phrases discussed by Niezgoda & Way [7] requires manual selection of such a phrase, making it inapplicable when searching through a large volume of messages without knowing what we are trying to find. Most forms of stylometry also fails to be of use when texts are as short as Tweets are, given that comparing structural features like average paragraph length simply is not applicable. Some may however be of use, such as counting special words or lexical features like average word length. Even those provide poor correlation between single messages however, and are better suited to analyzing if different accounts are used by the same author.

Running comparison data through internet search engines like Google is unlikely to provide much benefit when comparing Twitter accounts, as this is more likely to reveal cross-posts to other platforms or sources used than similarities between accounts and messages.

In regards to the lexical processing necessary only a minor part of what Bilal Ahmed [11] discusses is really relevant here. Since our goal is to detect similarities in messages there should be little need for normalizing abbreviations or shorthand. What is relevant to us is tokenizing Tweets correctly and using regex filters to find Twitter-specific characters.

Among the more direct methods discussed a word-by-word comparison is the easiest to implement and doesn't suffer much from the brevity of Tweets, but it is easily defeated by minor changes in a text. Fingerprinting is considerably more robust in that a minor offset of words is less likely to hinder detection. The drawback is that the 140-character limit on Tweets gives relatively little space for anchor and fingerprint selection, meaning there's a higher risk of false positives.

The best solution should be calculating the angular similarity of messages using a vector space model as this method is not adversely affected by the brevity of the texts used, nor does it require tuning or input to find similarities. While Maurer et al. [5] commented that this method was not always computationally feasible, advances in processing power since then mean this is no longer an issue. The primary weakness here is the need for more pre-processing of the text than other methods. The more advanced formula discussed by Bao et al. [10] does need a word weight vector as well, adding a level of complexity to the implementation.

3 Methods

This section covers how data was collected and datasets created, how messages were parsed prior to performing our tests, and which specific algorithms were used as well as how they were implemented.

3.1 Collecting Tweets

Using the REST API exposed by Twitter we acquired an authentication token (or OAuth token, representing a login for our program) for application-only level authorization, i.e. without user login credentials. By using this we could download the timeline of non-protected user accounts, providing up to the last 200 Tweets from them. Retweets were not included, by requesting only original (non-retweeted) messages via the Twitter API. As the timeline is sent in JSON format by Twitter it was decoded and stored as a collection of Tweets, each containing the text of that Tweet as well as metadata like a message ID, the relevant username, and so on.

3.2 Tokenizing & Processing Messages

Before applying any plagiarism detection algorithms the text from each message was normalized and tokenized in different ways to make processing possible with the various algorithms. Normalization was done with regular expressions (regex) to remove hashtags (# signs), user tagging (@username strings), web address links, and replacing all forms of whitespace with regular spaces. For algorithms that required no additional processing the text was broken down into tokens by separating words by using whitespace, special characters and anything else that is not a letter of the (unicode) alphabet as a delimiter. For the angular distance model it was necessary to remove common words prior to tokenization. To this end we passed the normalized text through a filter of the 50 most common words in the English language (“the”, “of”, “to”, “and”, etc.) before breaking the text into tokens as above.

3.3 Datasets

The last 200 timeline messages (minus retweets) were scraped from two different user accounts (the National Aeronautics and Space Administration, @NASA and CNN’s breaking news feed, @CNNBRK) selected to have a minimal overlap in messages. Any similar messages between these sets were manually removed, so as to have two clean sets as a control group.

A number of messages were then crafted and inserted into these sets, to be able to test the effectiveness of the plagiarism detection algorithms. These consisted of three types: messages meant to be more or less easily detectable, messages that may be detected incorrectly (false positives) by certain algorithms, and messages that would attempt to defeat the detection algorithms. The crafted messages were also varied in length to be short (less than 50 characters), medium length (50-100 characters) and long (100-140 characters). This formed our dataset for all further testing. By combining these messages with the control group data we hoped to add some noise to the dataset that might help us find unexpected results.

The group of messages crafted to be easy to detect as having a high similarity focused on having only minor differences between messages, such as adding hashtags or usertags or having a single word replaced or added. These were the messages used when considering whether an algorithm was capable of solving the problem, as mentioned in the problem statement, as these are the kinds of messages we would expect to find in real world applications.

The group of messages used to test each algorithm’s likelihood of generating false positives did so by having a high degree of syntactic similarity but with different meanings, as well as by attempting to test for word weight bugs.

Those messages that were designed to be more difficult to find were mainly created by paraphrasing, either manually or by passing messages through internet paraphrasing tools.

See appendix A for a list of crafted messages used.

3.4 Algorithm Applications

The most common plagiarism detection algorithms were tested against our dataset, including word-by-word comparisons, fingerprinting and calculating the angular distance between messages. This subsection covers how these were implemented.

3.4.1 Whole Message Comparison

To test that data collection had not failed prior to tokenization, the text from whole messages were compared to see that identical messages from separate sources were indeed detectable as being the same.

3.4.2 Word-By-Word Comparison

To compare messages word-by-word we iterated over each pair of messages, comparing if the i:th token of each message were identical. Each occurrence of identical tokens in a given position was counted, allowing us to calculate the number of co-occurent words relative to the message length (of the longer message when they were not equal) as a measure of similarity. Only messages detected as having a 50% similarity or greater were included in the results.

3.4.3 Fingerprinting

For fingerprinting we used a method similar to what Manber [6] referred to as a random selection of fingerprints from a given text. While Manber’s work used a mathematical approach to be able to apply it to any form of UNIX document or file, we used the fact that we would only operate on text strings to instead apply direct text comparison when performing our tests.

For each message from the first user account a short snippet of normalized (but not tokenized) text (1/4 the length of the message) was selected at random. Each message from the second user account was then tested to see if it contained that same snippet of text.

Due to the random nature of this fingerprinting it was necessary to perform it repeatedly against our dataset to test how well it performed. Data from 10’000 repetitions were collected and analyzed for how many messages were correctly identified and how many false positives were detected.

3.4.4 Vector Space Model

The model for calculating angular distance referred to by Maurer et al. [5] was used when performing tests with vector space models. Note that this algorithm does not take into account word weights, as we wanted to test the more common form of these algorithms. For each pair of messages from the two user accounts in our dataset a vocabulary was built, using their list of tokens where common words had been removed. The messages were then converted to numerical equal-length vectors by comparing their contents to the shared vocabulary.

These vectors were then used as input for the angular distance model, returning a similarity score value between 0 and 1 representing the cosine of the angle between the vectors. Results were filtered to include only those with a similarity score greater than 0.5 so as to prevent obvious false positives while also allowing for messages that were not quite identical be detected.

4 Results

For each algorithm tested the results were collected and are presented below as a graph of how well they handled the different types of messages we crafted, both in terms of correct identifications and false positives generated.

4.1 Whole Message Comparison

Identical messages were correctly identified. Any changes to either text being compared caused this method to find no similarity between them, as was intended. As such this method could only identify those messages in our dataset that had no differences between them, and nothing else.

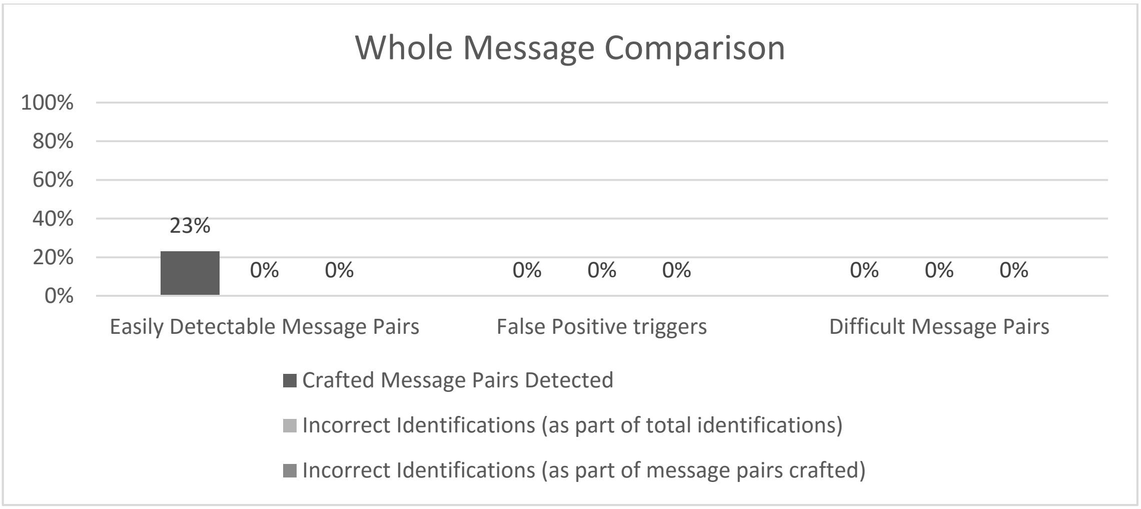

Fig. 1: Accuracy results of comparing whole messages. For each dataset we show the number of intentionally similar message pairs correctly identified (left bar), number of false positives generated relative to the number of message pairs flagged as similar (middle bar) and the number of false positives generated relative to the number of actually similar messages.

4.2 Word-By-Word Comparison

As predicted this algorithm succeeded in detecting messages identical up to the i:th word as well as those with identical words in the same i positions. Any offset (such as an extra word inserted at the start of an otherwise identical message) did however cause it to fail as the words compared were then offset throughout the messages. Replacing words with their synonyms (as many paraphrasing tools do) could cause it to fail when too many words were replaced, but it generally performed well against machine-based paraphrasing. Manual paraphrasing did however defeat it easily.

When testing for easily identifiable similar messages it correctly identified 69% of messages, while tests for false positives caused it to incorrectly identify 50% of messages. Against messages crafted to evade detection it could only identify 20% of messages as similar.

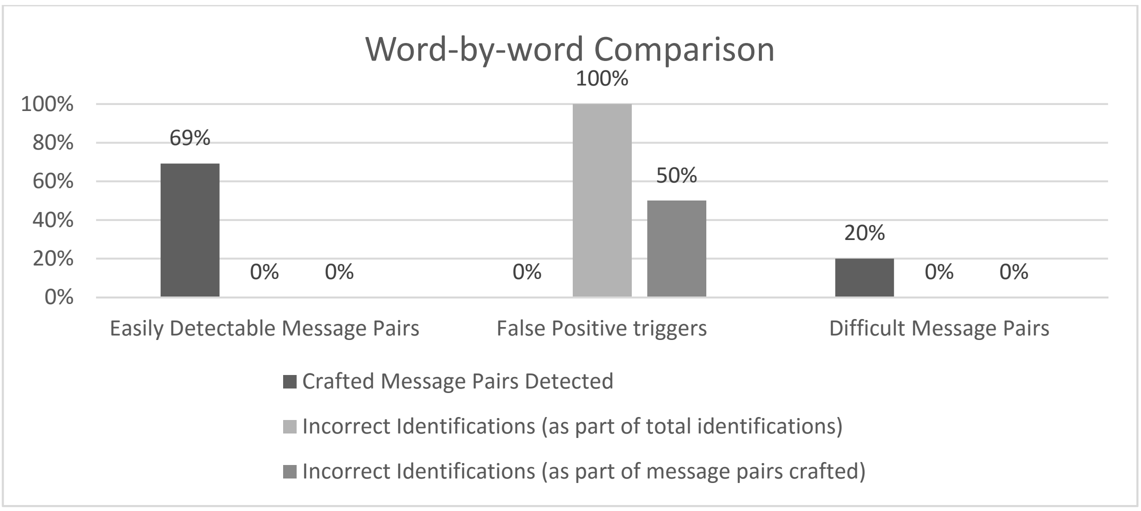

Fig. 2: Accuracy results of the word-by-word comparison algorithm. For each dataset we show the number of intentionally similar message pairs correctly identified (left bar), number of false positives generated relative to the number of message pairs flagged as similar (middle bar) and the number of false positives generated relative to the number of actually similar messages.

4.3 Fingerprinting

Compared to other methods it appeared that fingerprinting caused far more false positives to be detected. When testing for easily identifiable messages with high similarity it correctly identified 85% of messages, but also incorrectly identified a third of all messages in the dataset as being suspiciously similar. Against message pairs created to elicit false positives it incorrectly identified 74% of messages as being similar. Of these incorrectly identified message pairs approximately half included messages from the control group data scraped from Twitter. When testing against those message pairs we created so they would be difficult to detect it correctly identified 18% of messages, and incorrectly identified 1% of messages.

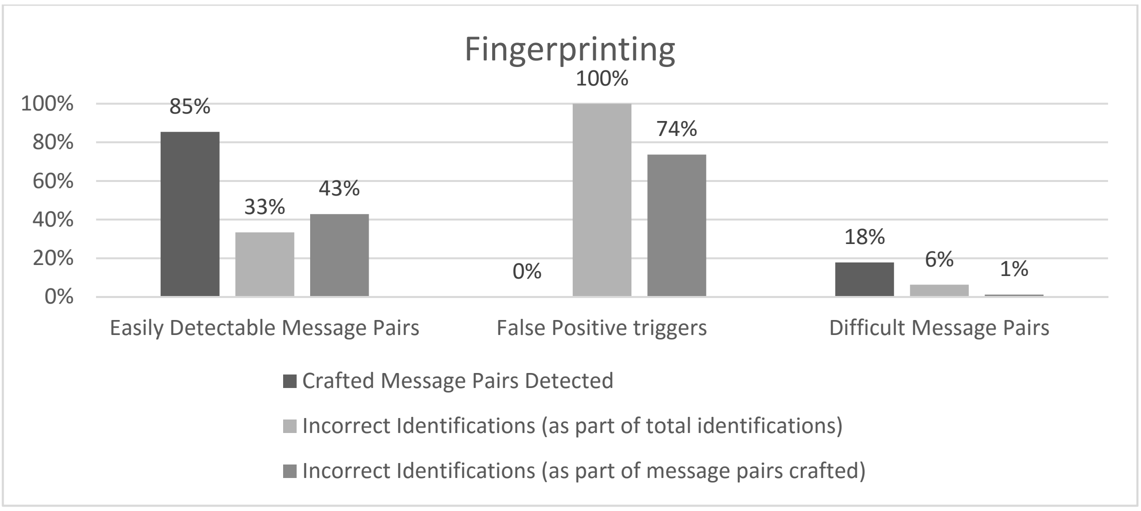

Fig. 3: Accuracy results of the fingerprinting algorithm. For each dataset we show the number of intentionally similar message pairs correctly identified (left bar), number of false positives generated relative to the number of message pairs flagged as similar (middle bar) and the number of false positives generated relative to the number of actually similar messages.

4.4 Vector Space Model

In tests against more easily detectable messages this method correctly identified 100% of messages with no false positives. Against our intentional false positives it incorrectly identified 33% of messages. When tested against those messages that attempted to evade detection it correctly identified 50% of messages, with no false positives generated.

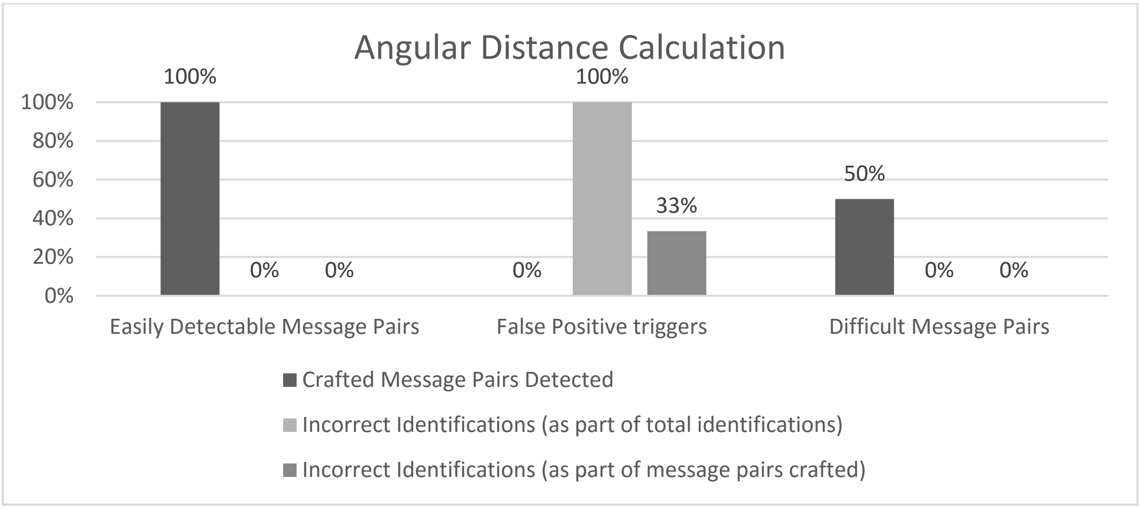

Fig. 4: Accuracy results of the algorithm using a vector space model to calculate the angular distance between messages. For each dataset we show the number of intentionally similar message pairs correctly identified (left bar), number of false positives generated relative to the number of message pairs flagged as similar (middle bar) and the number of false positives generated relative to the number of actually similar messages.

4.5 Summary

For a better overview of how the various methods we tested performed, see figures 5 & 6 below. These compare how well each algorithm performed against our easy and difficult datasets respectively. Note that only fingerprinting generated a significant amount of false positives.

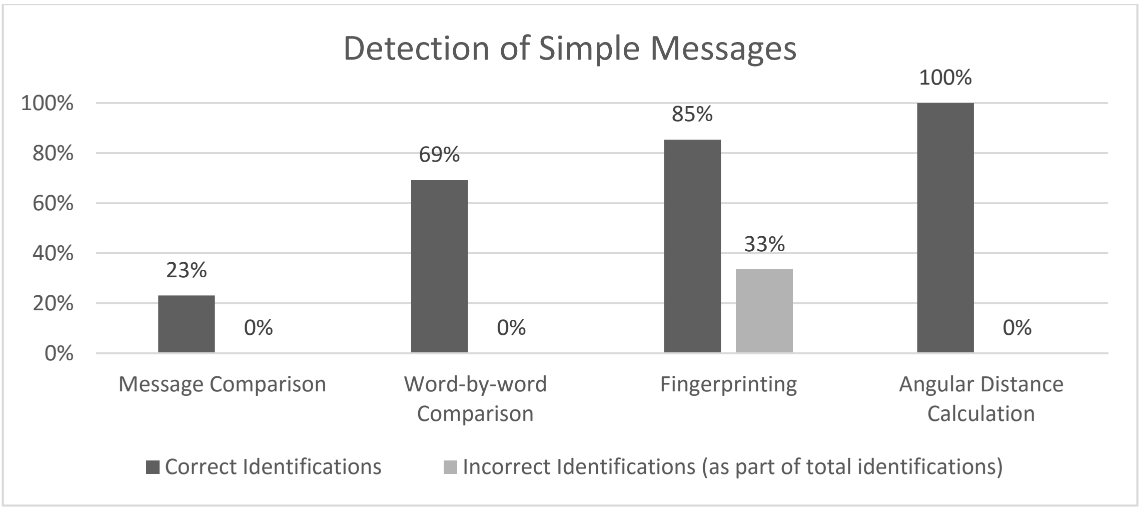

Fig. 5: Overview of the accuracy results of the various algorithms tested when using the easier dataset. For each algorithm we show the number of intentionally similar message pairs correctly identified (left bar) and the number of false positives generated (right bar).

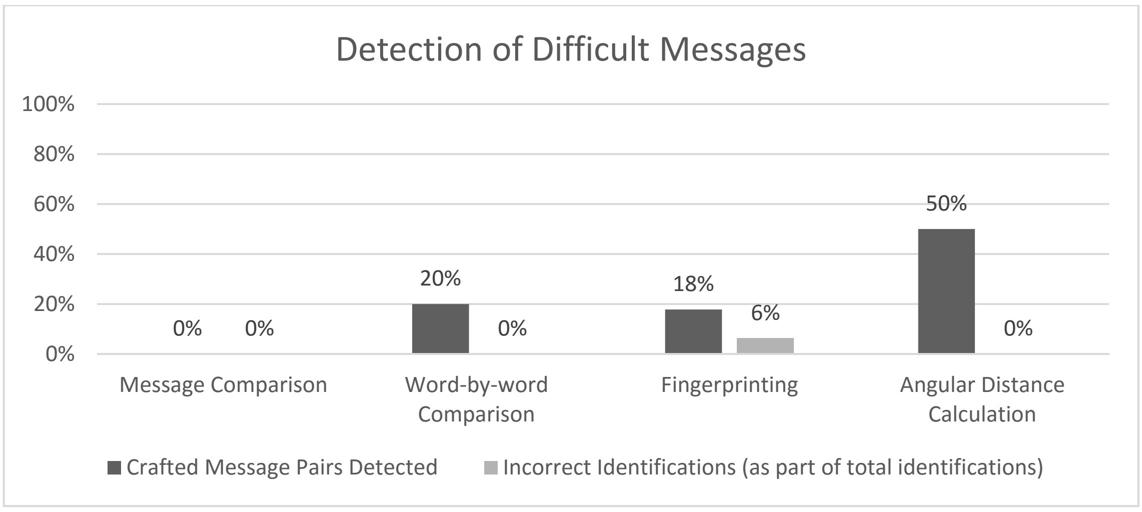

Fig. 6: Overview of the accuracy results of the various algorithms tested when using the more difficult dataset. For each algorithm we show the number of intentionally similar message pairs correctly identified (left bar) and the number of false positives generated (right bar).

5 Discussion

The problem statement of this paper asked if it was possible to detect collusion between user accounts in spreading messages on Twitter by collecting messages (in bulk) and applying plagiarism detection algorithms. Our results show that this is indeed possible, though different algorithms will have varying rates of success in detecting similar messages (see table 1).

Of the various plagiarism detection algorithms tested, calculating the angular distance between texts was by far the most successful. Using a simple word-by-word comparison did reasonably well against trivial cases, and handled texts passed through paraphrasing tools well when they were of the kind to simply replace a few words with synonyms.

The algorithm used for fingerprinting also identified most of the trivial messages, but at the cost of a generating a very high rate of false positives. This was likely due to the nature of the short texts found in Twitter messages, along with the specific implementation we used. While it is a technique that likely works well on a larger corpus of text, it can not be recommended for any kind of microblog use where messages tend to be minimal in size – even with an implementation that tried to better adapt the fingerprint size to these short texts it would likely generate too many false positives to really be useful.

Method	**Attributes**		
	Sufficient accuracy	Few false positives	Successful overall
Message Comparison		✓	
Word-by-word Comparison		✓	
Fingerprinting	✓		
Angular Distance Calculation	✓	✓	✓

Table 1: A summary of how the methods tested met our demands on accuracy and false positives, as described in the problem statement.

It should be noted that those messages in the difficult-to-detect group where paraphrasing was heavily used to change texts, while more difficult to detect with these algorithms, were generally trivial to detect manually when they were generated by automated tools. The way these were worded would also make them useless for sockpuppets striving to appear as real users, as these tools tend to create texts that are simply not idiomatic.

Paraphrasing performed manually was the most difficult to detect overall. It is a technique that requires far more work that others however, and it is thus questionable if it would be used for spreading a message from a large number of sockpuppet accounts given the time this would take. In cases where it is used it does not appear that the algorithms tested would be able to detect it.

5.1 Limitations

The group of messages crafted to elicit false positive identifications likely yielded some unreliable results; these were difficult to design and none of the included algorithms are capable of the deeper understanding of a text necessary to test for changes in the meaning that can arise from changing only the subject or verb in a sentence, for example.

The other types of messages crafted for our dataset could also have been more numerous, both to better test for various special cases and to give a more robust testing overall. The data gleaned from the tests performed did however seem sufficient to determine a general degree of effectiveness for the various algorithms used and how they compare to one another.

Implementing one of the more advanced forms of vector space model discussed to by Bao et al. [10] may have given more insight into whether that degree of complexity is necessary for these types of texts, but as this paper was chiefly concerned with whether this type of algorithm was at all applicable to the problem at hand we felt this degree of complexity fell outside the scope of it.

5.2 Future Research

While our work shows that detection of information control by way of sockpuppets is possible, it also leaves several new avenues of research to explore. Other forms of microblogs, and social media in general, have their own unique settings to take into account when applying the algorithms we have suggested, and may provide further lessons.

There is also the possibility of further testing of which algorithms provide the greatest accuracy and with the best performance – variations on vector space models such as those discussed by Bao et al. (one of which we highlighted in the background section) may provide even greater capacity for identifying mass-posted messages.

Solutions that make it possible to analyze larger datasets from a greater number of users at once, or even in real time, would of course be the ultimate tool against these forms of information control. If the attempts to identify them could be made proactive, rather than reactive, it would likely mean the end for this particular breed of digital age propaganda.

6 References

[1] Urban Dictionary. [Online] [Cited: Feb 21, 2017.] https://www.urbandictionary.com/define.php?term=signal+boost&defid=4480596

[2] Urban Dictionary. [Online] [Cited: Feb 21, 2017.] https://www.urbandictionary.com/define.php?term=sock+puppet&defid=437318

[3] Chen, Adrian. The Agency. The New York Times. [Online] June 2, 2015. [Cited: Feb 27, 2017.] https://www.nytimes.com/2015/06/07/magazine/the-agency.html.

[4] Sydell, Laura. We Tracked Down A Fake-News Creator In The Suburbs. Here's What We Learned. National Public Radio. [Online] November 23, 2016. [Cited: 02 28, 2017.] http://www.npr.org/sections/alltechconsidered/2016/11/23/503146770/npr-finds-the-head-of-a-covert-fake-news-operation-in-the-suburbs.

[5] Maurer, Hermann A., Frank Kappe, and Bilal Zaka. "Plagiarism-a survey." J. UCS 12.8 (2006): 1050-1084.

[6] Manber, Udi. "Finding Similar Files in a Large File System." Usenix Winter. Vol. 94. 1994.

[7] Niezgoda, Sebastian, and Thomas P. Way. "SNITCH: a software tool for detecting cut and paste plagiarism." ACM SIGCSE Bulletin. Vol. 38. No. 1. ACM, 2006.

[8] Zu Eissen, Sven Meyer, and Benno Stein. "Intrinsic plagiarism detection." European Conference on Information Retrieval. Springer Berlin Heidelberg, 2006.

[9] Stein, Benno, Nedim Lipka, and Peter Prettenhofer. "Intrinsic plagiarism analysis." Language Resources and Evaluation 45.1 (2011): 63-82.

[10] Bao, Jun-Peng, et al. "A fast document copy detection model." Soft Computing 10.1 (2006): 41-46.

[11] Ahmed, Bilal. "Lexical Normalisation of Twitter Data." Science and Information Conference (SAI), 2015. IEEE, 2015.

[12] Twitter REST API. [Online] [Cited: March 31, 2017] https://dev.twitter.com/rest/public

Appendix A: Dataset

A1. Similar Messages

Account A	Account B
These messages are identical.	These messages are identical.
Once again we see texts that are very much the same.	Once again we see texts that are very much the same.
I will leave out all discussion on republics, inasmuch as in another place I have written of them at length, and will address myself...	I will leave out all discussion on republics, inasmuch as in another place I have written of them at length, and will address myself...
We must not let terrorism win! Praying for peace and unity now.	We must not let #terrorism win! Praying for #peace and #unity now.
.@johndoe did you see the last episode of Television Show? Things are really heating up.	Did you see the last episode of Television Show? Things are really heating up.
changing case shouldn't matter.	CHANGING CASE SHOULDN'T MATTER.
I think we can all learn something from the elderly. Their wisdom is inspiring to us all.	Certainly I think we can all learn something from the elderly. Their wisdom is inspiring to us all.
The experience of survival is the key to the gravity of love.	The experience of survival, they say, is the key to the gravity of love.
If I stop drinking all at once, I'm afraid the cumulative hangover will kill me.	If I stop drinking all at once, I'm afraid the cumulative hangover will kill me, honestly.
Literature is a luxury.	Literature is a luxury, fiction is a necessity, said Chesterton on the subject.
I fear for our future.	I worry for our future.
For him, there's really no such thing as deciding on a profession to grow up into.	For him, there's really no such thing as choosing a job to grow up into.
The UK government has said that it wants talks on trade relations to run in parallel with the separation negotiations.	The UK government has said that it wants discussions on trade deals to run in parallel with the separation negotiations.

A2. False Positive-generating Message Pairs

Account A	Account B
This message is blue.	This message is red.
I love my cats, so cute! I like to take them out for walkies.	I love my dogs, so cute! I like to take them out for walkies.
I strongly believe we must stand together and unite against all those who would threaten our access to free bagels.	I strongly believe we must stand together and unite against all those who would threaten our way of life.
I once owned a horse.	horse horse horse horse bird
My old washing machine broke down recently. It was really bad.	washing washing washing washing washing washing things.
I really like the idea of living out near nature, it's just so tranquil out where the animals roam.	tranquil tranquil tranquil tranquil tranquil tranquil tranquil tranquil tranquil home.

A3. Difficult Message Pairs

Account A	Account B
Possible shooting in NYC, scared.	Someone shooting in new york, terrible.
All options for responding to future provocations must remain on the table.	Any and all possible solutions to handling future provocations have to be considered.
Healthy young child goes to doctor, gets pumped with massive shot of many vaccines, doesn't feel good and changes - AUTISM. Many such cases!	I've heard of ordinary children being treated with vaccines at the doctor's ending up with autism. Apparently common, and very disturbing.
Large fire reported in #boston, firefighters seen on the way there. Did anyone hear the explosion a minute ago?	Big fire in #boston, firetrucks seen on the way there. Did anyone hear the loud noise a while ago?
178 Savin St. evacuated due to strong chemical smell. Police and Fire evacuated the building, secured the area and called hazmat.	Savin street evacuated, chem. smell in the air. Emergency services clearing out occupants, calling in specialists.
But there are stipulations that must be met first.	But there are conditions that must be met first.
Our notion of similarity throughout this paper is completely syntactic.	Our notion of kindred attribute throughout this paper is consummately syntactic.
His administration could play as comedy, if not for its vast capacity to harm.	His management could play as comedy, if not for its huge ability to harm.
Statistics indicate that restaurant visits in the United States declined in 2016 for the first time in years.	Measurements show that eatery visits in the Assembled States declined in 2016 without precedent for years.
All the progress that we’ve made these last eight years goes out the window if we don’t win this election!.	Every last one of advancement that we’ve aggravated these final one eight a considerable length of time dives crazy the window In we don’t win this election!.