University of Glasgow | School of Computing Science

Tracing malicious behaviour on Twitter

George Illingworth

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Level 4 Project — March 28, 2018

Abstract

Social networks have become one of societies primary means of sharing content and belonging to communities. This gives them great power, and the conversations had through them can hold significant value. If a malicious user wishes to control this power they may be able to create a number of puppet accounts, and through them amplify their own voice and give the image of consensus where one does not actually exist. This can influence public opinion in many ways, and has already been used to try and influence important political elections. Detecting these puppet accounts is therefore of great importance. A series of algorithms are constructed that compare the semantics, syntax and grammar of Twitter users. These algorithms were designed to build a *linguistic fingerprint* by which a user could be positively identified. The metrics designed were tested on an appropriate data set, and it can be said with a reasonable degree of confidence that they could provide significant benefit to a wider system that wished to tackle the problem of single users operating multiple accounts.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: _____ Signature: _____

Contents

Chapter 1

Introduction

1.1 Background

Social networks form the backbone of communication in the modern internet. The SimilarWeb records[11] for the most-used websites worldwide place various social networks in 7 of the top 50 places. While search engines account for a further 24 places in this list, the significance of their dominance is lessened by the fact that the overwhelming majority of these (21 of the 24) are various international versions of Google. As the usage of social networks increases across the globe, the cultural value and power they hold grows too.

Social networks represent a fundamental change in the way news, entertainment and other media are processed, produced and shared. Up until within the last decade, only a small subset of the population was responsible for the vast majority of content creation that could be viewed by the majority of the population. Monolithic institutions - newspapers, radio stations, television networks - provided the news of the day in an edited, concise and condensed form.[1] These institutions have always made very deliberate decisions about the sort of content that they choose to provide their audience; every article in every newspaper was, and is, run through multiple layers of editing to ensure that structure, style and content are consistent with the image that they want to present. To become a writer, to be allowed to produce material for the general population to consume, then required years of education, training and work experience.

The explosion of social networks changed all of this. As a user of a social network, every individual, broadly speaking, has exactly the same rights to produce the content they want as everyone else. This freedom applies to every facet of the content creation process. It is possible to imagine a writer for a newspaper having carte blanche to write about any subject they see fit, but they will still be beholden to certain style guidelines. They would not, for example, be allowed to publish full articles using only capital letters, or to misuse and mangle punctuation and spelling for narrative effect.

In addition to this truer linguistic carte blanche, every social media user can decide exactly how much of any other user's content they want to consume at any given moment. If one user likes what another is doing, they are free to copy or share it as they so choose. The potential for such curation and fluidity allows communities to be built and messages, ideas and sentiments to be spread all over the world at unimaginable speeds.

The ideas that are spread so virally may present little cultural value beyond temporary amusement - it is hard to argue that social phenomena such as the 'planking' fad of 2011 have left any lasting mark on society - or they may exert huge influence in enacting major changes in national and international politics.

[1] There were, of course, writers and producers of all sorts of content that catered to local niche audiences but in order to achieve a wider reach they needed the approval of larger organizations.

Also in 2011, Hosni Mubarak was ousted as President of Egypt after holding the position for nearly 30 years.[1] Social networking platforms in this case gave voices to those who would otherwise not have been afforded them, and empowered these voices with a much greater reach than traditional word-of-mouth organisation. The same tools, generously provided by social networks, have been used to similar effect in Ukraine[2], Iran[3] and Tunisia[4]. While it would be overly simplistic to state that none of these events would have happened without social networks, their influence was strongly and incontrovertibly felt.

There is, then, significant value and power in the communities and the discourse which are made possible through social networks; and where there is power there are going to be people who want to control it. In some cases exploitation may be purely selfish: through the creation of several sock puppet accounts, a user could feasibly construct notoriety and fame for themselves. Possibly the best documented of these phenomena occurred within a large community of Harry Potter fans spread across many websites between 2003 and 2006.[2] Fiction Alley[3] user 'msscribe'[5] created and maintained a series of puppet accounts across many similar sites that would both praise and deride her work. Through the intentional creation of controversy, she managed to achieve significant status within that community.

The sort of behaviour exhibited by 'msscribe' has continued in different, and in many ways more sinister, forms as the global influence of social networks has grown. Rather than individuals simply trying to create a name for themselves, political groups and even nation states have been shown to use similar tactics in attempts to exert influence in their own countries and abroad. This is now an expected part of the news cycle for any major political event that has occurred in the last few years. The United States Presidential Election of 2016[6] , Brexit[7] and the French Presidential Election of 2017[8] all featured some form of attempted tampering from outside sources via groups of directed social networking accounts.

There is definite tangible value in being able to detect when this sort of attack is happening in an automated manner, but positively identifying this can be very challenging. The difficulty involved in detecting such groups of accounts depends heavily on how they are created and how they are being run. In simple cases where no effort is made to disguise the location of the individual or group who is orchestrating the attack, they can be traced via their IP address, but it is trivial for these addresses to be faked or hidden. Other factors that can be used to trace such activity include the creation of many similar accounts in a short time period; often rudimentary approaches to grammar; repetitiveness; and formations of unusual posting patterns between accounts.[6] All of these hallmarks can be obscured in some form, and most are negated if each account is being controlled manually.

1.2 Motivation

The democratization of content creation allows every user to form their own style, where they can choose to use - or ignore - any of the customs of any language they want. While in traditional media a writer is encouraged to find their own voice within strict grammatical guidelines, the linguistic shape of the content produced by a social media user is allowed to be considerably more individualized.

In traditional writing there is posited to exist a sort of *linguistic fingerprint*[9], where each writer has their own style that can be positively identified. The applicability of this rule relies heavily on the context: successfully identifying a single writer from a single piece of text is likely to be extremely difficult in all but a few cases; but to be able to compare two pieces of writing and claim that they are likely to have the same author seems much more reasonable. It is relatively well established that a writer's *fingerprint* changes over time[10], but these changes

[2]Before large-scale social networks were popular on the scale that they are today, most interaction online happened on forums. They differed from social networks in that each forum would tend to be focused on a single topic, but in many ways can be considered analogous.

[3]A now archived website for users to post their own writings that exist within and expand upon the Harry Potter universe.

happen gradually and in a manner that would still be trackable if it were possible to access a consistent stream of writing over time.

This paper examines the hypothesis that each social media user leaves behind some kind of *linguistic fingerprint*; and that, through focusing purely on the textual content of an account's posts, it is possible to determine the author. Furthermore, a series of algorithms designed to extract such a fingerprint are designed, tested and analyzed.

If a linguistic fingerprint could be traced, it would be possible to use this as a method of detecting the targeted attacks laid out above. This could then be combined with the more commonly used methods - tracing IP addresses, relative posting times and repetitiveness - in order to provide even greater clarity. It is also likely that a great effort would be required to alter this *fingerprint* to the degree that the true users' identities could be concealed, especially when compared to the methods mentioned earlier.

These algorithms were designed around Twitter users, for a number of reasons. Twitter is the third most popular social media site worldwide[11]; it grants considerably more access through APIs to outside developers; and it sets a strict, small character limit per post. This limit - historically 140 characters but November 2017 moving to 280[12] - both encourages unusual linguistic behaviour as users try to encapsulate as complete an idea as possible within the character limit, and provides relatively small blocks which can be analysed independently. The length constraints on each post allow them to be considered individually in a more performance-efficient and meaningful manner. Two of the three algorithms proposed run in at least linear time with respect to the length of the average post, and so a low upper-bound on that length keeps the run times reasonable.

As discussed, there can be no expectation that a user will punctuate correctly, and so attempting to distinguish sentences automatically can in many cases present an extremely complex task. However, the character limit imposed by Twitter means that users are forced to split long and complex ideas they may wish to write about across many posts. Thus most individual posts contain only a few key ideas, meaning there is a reasonable equivalence to be drawn between a single Twitter post and a single sentence. This is extremely helpful as the deconstruction of a sentence to its constituent grammatical parts is a common linguistic task[13], and so there has been a large amount of research done in this area and complex tools created to perform this task.

Chapter 2

Forming a Linguistic Fingerprint

2.1 Forming a Fingerprint: Theory

2.1.1 Naive Bag-of-Words Approach

The simplest of the methods chosen to determine a fingerprint for a given user starts by forming a set of all the words present in the user's posts. In this method, the term 'word' simply denotes any sequence of characters separated from those around it by spaces. This could include URLs, mentions of other usernames, emoticons, mis-spellings and a variety of other non-dictionary words, but these were not filtered, as they also contribute to the fingerprint of an individual. Since a set is a group of unique elements with no defined order, this set formation turns a list of posts - which can be considered as lists of words - into a disordered group of words. A representation of this process is shown in 2.1, where the lists of words have been replaced with a table of numbers for ease of reading.

These sets can then be compared to each other by examining the ratio between the sizes of the original sets (A and B in 2.2) to the size of the common set (C). If this ratio is unusually high, it is hypothesized that the author of each set is likely to be the same person.

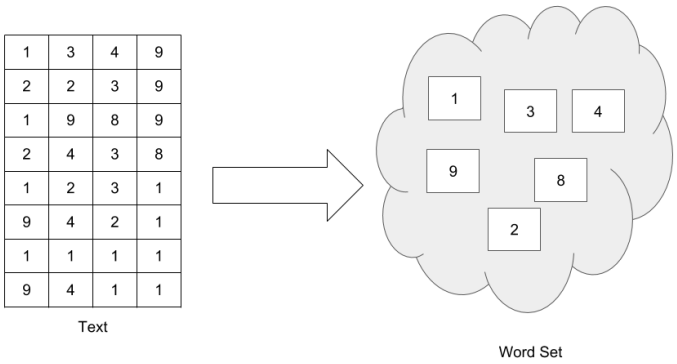

Figure 2.1: Graphic representation of set formation.

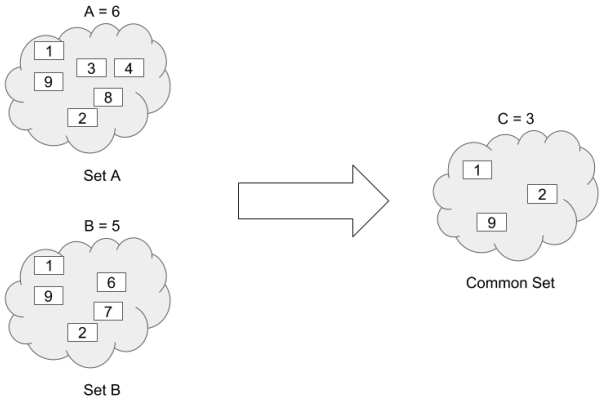

Figure 2.2: Graphic representation of set comparison.

2.1.2 Part-of-Speech Approach

The Bag-Of-Words approach detailed above focuses only on the word choice of a given account; yet the grammatical structures may be just as revealing. Sentences are made up of several constituent parts: they should contain at least a noun and a verb, but almost all contain more words with different, more complex uses. Of course, Twitter users are not constrained to writing in complete sentences or even to attempting to impart meaning in this way. In practice, though, it is found that the vast majority of users are attempting in some way to transmit some form of information - but often in ways that fail to comply with, or even actively disregard, traditional ideas of grammar and structure.

The grammatical informality of Twitter communication could potentially cause issues when attempting to decipher the structures being used, as the vast majority of suitable tools are designed mainly with various formal writings in mind. That being said, when writers are not constrained by the expectation of using strict grammar and convention, they are likely to create or adopt their own. These new conventions are likely to enhance their fingerprint and are detectable with the right tools. The space constraints enforced by Twitter especially exacerbate the need for linguistic improvisation: if a post is slightly too long, the user is likely to look through what they have written for any superfluous words they can remove entirely without the post losing meaning. In doing so, the user is forced to make their own decisions about what they consider to be important in a sentence, and these choices are likely to be relatively individual. When a user is forced to make these sorts of decisions regularly, patterns should begin to emerge and these should be detectable.

There is not a simple 1:1 relationship between a word and its meaning in a syntactic sense. For example, consider the sentence “I refuse to collect the refuse.“ The word “refuse” is used both as a noun and a verb, so the context is important in selecting which meaning is most appropriate at what time. Deciphering meaning in this sense can be computationally complex, but as discussed in 2.2.3, there are freely available tools that can perform most of the heavy lifting for us. This process is called tagging and the tools that do it are known as taggers.

Tagging a post leaves us with a list of the parts of speech used within it, and these lists can be compared to one another in a number of ways. The metrics that were hypothesised to provide greatest insight were the edit distance (ED) and the longest common subsequence (LCS). If posts are broadly similar in makeup, then this is tracked by the ED, and if they reuse certain smaller structures, these are tracked by the LCS. This provides us with effective tools for comparing one post to another, but these measures are statistical and noisy and therefore only of any real use in aggregate.

The chosen method of aggregation when comparing two sets of posts is to compare the means of these measurements within and between the sets. 2.6 illustrates the comparisons required to evaluate two sets with

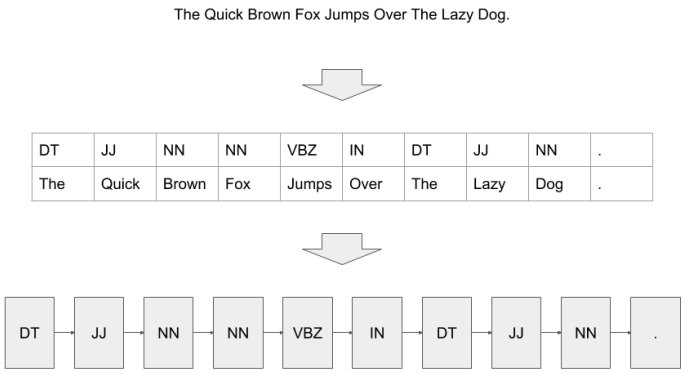

Figure 2.3: Graphic representation of tagging.

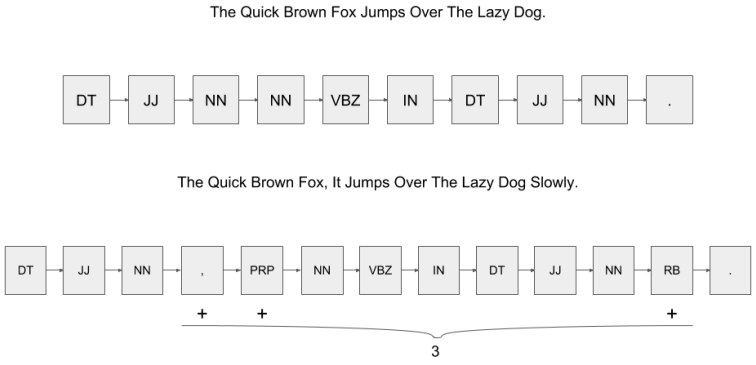

Figure 2.4: Edit Distance Capture.

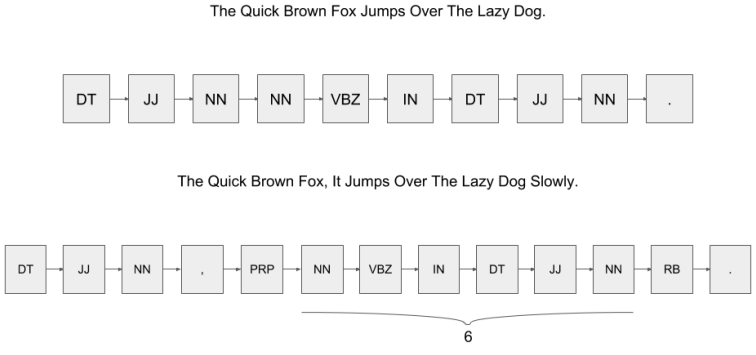

Figure 2.5: Longest Common Subsequence Capture.

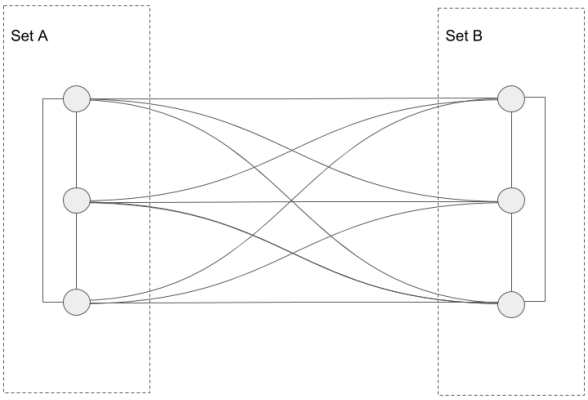

Figure 2.6: Comparisons performed within and across 2 sets.

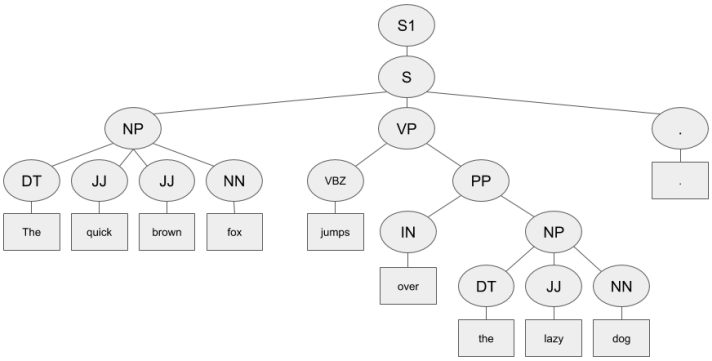

Figure 2.7: Tree formation for “The quick brown fox jumps over the lazy dog“.

three elements each. If we are to compare the sets of posts A and B, the first step is to find the mean ED and mean LCS between one post in A and another also from A. Next the mean ED and mean LCS between posts from B are taken, before finally the same measurements are performed between the sets.

The mean length of each list in each set was also recorded, as longer lists tend to increase both ED and LCS. It is theorized that the mean ED and mean LCS should be lower in the within-set cases than the between-set case when they have different authors, and that if they are very similar then they have the same author.

2.1.3 Tree-building Approach

As a fundamental function of written language, the words in a post are arranged in a linear fashion. Representing the structure of a sentence as a list of parts can be valuable and is in many senses accurate, but it is not a complete picture; writing, as it is used and understood, does not function like a list. There is value in understanding language in this way, but the underlying structures cannot be fully expressed in one dimension.

A word relates to more than just the words before and after it: they form groups within themselves as phrases or clauses, and these groups can themselves form larger structures which eventually build into a complete sentence. This can be represented as a tree (as shown in 2.7 and 2.8), where each of the words is a leaf and every grouping is a parent node.

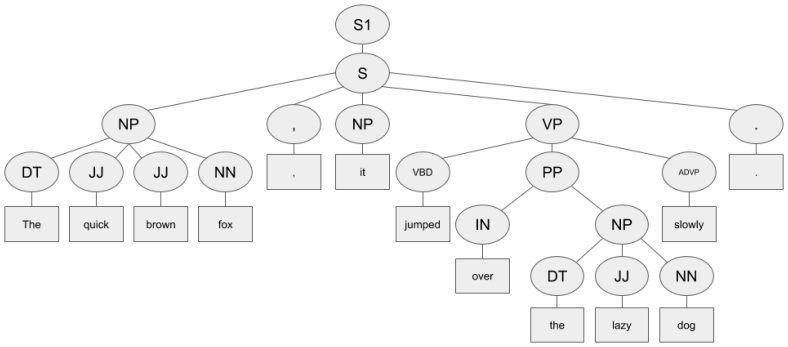

Figure 2.8: Tree formation for "The quick brown fox, it jumped over the lazy dog slowly".

There are freely available tools which can parse the sentence into these tree structures; and although it is more computationally expensive than for lists, trees can compared by edit distance. It is also possible to calculate the largest common subtree - which is analogous to the longest common subsequence used in 2.1.2 - but this as this process is extremely complex[14] and would need to be implemented from scratch it was left out of these experiments. This means that the methods for obtaining similarity can follow the same patterns laid out in 2.1.2. When given two sets of posts, a tree is generated for each post in each set, and the ED between it and every other tree is calculated. The mean ED values within and between sets are recorded. As in 2.1.2, it is hypothesized that if the author of each set is different then there should be a higher mean ED between sets, and if they have the same author there should be no significant difference.

2.2 Forming a Fingerprint: Technical Detail

2.2.1 Building an Appropriate Data Set

In order to build a system which can test the hypotheses, it is firstly necessary to build an appropriate data set. Retrieving the most recent tweets from a given Twitter account is possible through the Tweepy module for Python. This module provides convenient access to Twitter's own APIs, which allows an authenticated user to download up to 3,200 of the most recent posts from any public account. A Python script was created which downloads all the posts from a given account, then finds all the accounts followed by the original account; it then performs the same action on these new accounts. The only information that is kept is the textual content of each account's posts, stored in a text file called either `<username>_1.txt` or `<username>_2.txt`. The reasons for splitting each account into two parts are explained below. This recursive approach allows for an arbitrary number of accounts to be crawled - the only theoretical constraints are the hard disk space and time available.

Beginning with the author's own personal Twitter account, this script ran overnight, by which time just over 1600 accounts' worth of data had been harvested. Using the author's own personal account as a starting point gives a significant skew to the data, but this skew provides several key advantages. Firstly, it significantly increases the probability of the accounts found to be using English as their primary language, which is necessary for the Parts of Speech approach and the Tree-building approach to function as intended. It also places a skew on the kinds of content that the accounts post - there are likely to be social, political and geographical biases across the entire data set - which arguably narrows the applicability of the results, but also means that the algorithms are tested on their ability to differentiate authorship, not just the expected content. For example, if tested, a number of the accounts gathered make extensive use of Scottish slang and dialect. If there were only one such account

in the data set, it could potentially be singled out on that basis alone, but the inclusion of many such accounts means that the algorithm's ability to distinguish individuality within such profiles is also tested.

To test the algorithms in any meaningful way, it is necessary to be able to generate known positive matches, where it can be said ahead of time that the two sets of posts have the same author. This is achieved by splitting each account into two pieces as it is downloaded, with posts being alternately placed into two distinct text files, which creates sets of posts that we can compare to each other in the near-certainty that they have the same author. If the hypotheses are correct then the results generated when each algorithm is given two sets from the same account should show significant difference to those from across accounts.

This approach is unique when compared to other investigations into the same problem: [23] used a combination of resources, including scraping Google Plus profiles to find users with multiple Twitter accounts and comparing users who were known to operate both personal and professional accounts. This represents a significant increase in outlay for a smaller, but theoretically more valuable, data set. These techniques do negate some issues apparent in splitting the same account into two parts, but given that there are likely to be differences between a user operating multiple accounts openly and those operating them in secret, it was decided that the extra outlay was not worth the perceived benefits when compared to the technique detailed above.

2.2.2 Naive Bag-of-Words Implementation

The implementation of this approach relies solely on tools and modules available in the stock Python environment. Each file is initially read into memory, and by using the `.split()` function, is turned into a complete list of all the words used. The default operation of this function is sufficient for this use case as it splits the initial string on spaces and on newline characters. Each list can then be cast to a set using `set()`, which enforces uniqueness as part of its definition and runs in $O(n + m)$ time where *n* and *m* are the lengths of each split list. The core processing algorithm is relatively simple and shown below:

Algorithm 1: Set Comparison for Naive BoW

```
Data: Posts from Set A
Data: Posts from Set B
Data: z
Result: Number of words present in both given sets of posts
for x in Set A do
    for y in Set B do
        if x is equal to y then
            z++
return z
```

This algorithm runs in $O(nm)$ time using the same notation as before.

The algorithm was initially developed to be run from the command line with two file locations as arguments. This allowed for the easiest and fastest testing of the preliminary phases, but once the core logic had been completed it was converted to run as a module using a public function called `run(file1,file2)`. When run from the command line the result is given as a single line with the names of the files used, the length of each set, the number of words found in both sets and the time in seconds taken for the calculation. The same information is returned as a string when it is called through the `run(file1,file2)` function.

That the initial code could be run from a separate Python script allowed the concern of running the algorithm to be separated entirely from the concern of collating many of these runs for data collection. The data collection was performed by another Python script which used the `glob` package to iterate over every set of posts that had the the suffix `_1.txt`. To begin each cycle it would run the algorithm on the other half of the same account, then

run the same algorithm between the inital set and 100 random sets that had the suffix `_2.txt`. All data returned by the `run(file1,file2)` function is written to a CSV file for analysis. This workflow is shown below:

Algorithm 2: Data Collection Workflow

```
Data: A = Data Set 1
Data: B = Data Set 2
for x in A do
    y = Corresponding file from B
    run(x, y)
    for z in range(0,100) do
        y = random file from B
        run(x, y)
```

2.2.3 Part of Speech Implementation

As in all cases, the core logic of this approach runs from a `.run()` function that takes two arguments with the location of two text files that should be full of posts stored as detailed in 2.2.1. To begin to analyse each post in a set, the file is first read into memory and then split in a similar manner to the initial phases of 2.2.2. However, rather than splitting the entire file into a list of single words, it is instead split only on newline characters. This forms a list with a single post in each element. The only notable change from the previous approach is that the character "@" is always replaced with an empty string; this denotes that a user is *tagging*[1] another user in that post, and it was found to cause the part of speech tagger significant difficulty. Removing the "@" left only the username, which gave the best chance that it would be classified as a name. The newly formed list is then iterated over, and each post in the list is processed individually.

As outlined in 2.1.2, this technique for determining authorship relies on forming a one-dimensional image of the syntactic structure of each post. To achieve this a number of features provided in the Natural Language ToolKit (NLTK) were used. The first such feature used was a word *tokenizer*: it splits the given string - in this case a single post - into a list of individual words. While this may seem very similar to the `split()` function called in 2.2.2, the tokenizer's method for splitting out the words is considerably more nuanced. This means that, for example, "didn't" would be split into "did" and "n't", and that punctuation is also split off into its own token when appropriate.

The list of tokens formed is then run through a part of speech tagger, which determines the syntactic function of each word. A list of tuples is returned, where the first element of the tuple is a string code which represents the part of speech - known from here as indicators - and the second is the original token. The tokens are then removed, and this list of indicators is stored for later comparison.

Once both sets of posts have been fully processed, the edit distance and the longest common subsequence between each post in the first set and each other post in the first set is established.[2] The edit distance is calculated each time using the *pip* (Pip installs Packages)[3] library `editdistance`. This library is capable of calculating the edit distance between lists of strings where each addition, removal or replacement of a string in the list counts as a single edit as required. The longest common subsequence was calculated using a short Python function found at [15], and the mean result of each calculation is stored.

A mean edit distance and a mean longest common subsequence are then calculated within the other set of

[1] this definition of tagging is distinct from and unrelated to the previous definition as outlined in 2.1.2

[2] It is of note that in the actual implementation this occurs before the second set is processed, but this change in order has no material effect and enables simpler explanation.

[3] Recursive acronyms are relatively common in open-source software.

posts by following the same process, and then again between posts from the first set and posts from the second set. Alongside the means of the comparisons within and between sets, the mean length of a tweet from each set and the total time taken were recorded. The mean tweet length was thought likely to be a confounding factor in further analysis as a longer list is likely to have more edits.

Another technique considered for comparing the individual posts was to convert the lists of indicators into single strings where each indicator was represented by a single character: there are only 36[16] possible indicators so they could be expressed using only lower case letters and numbers. This would enable use of the much more common string comparison algorithms, but initial testing into this technique showed it to be significantly harder to debug and it was therefore abandoned.

The core logic was then imported to another Python script which handled the aggregation and saving of the data. This script functions in an identical manner to the equivalent script laid out in 2.2.2: it iterates over the files using `glob`, compares each element in set A to its partner in set B and then to 100 randomly selected elements from set B and writes each result to the appropriate CSV file.

2.2.4 Tree-building Implementation

This implementation builds upon the same principal control flow as laid out in 2.2.3, where the initial data sets are split into their constituent posts and then each post is considered individually. Reusing the structure around the algorithms allows them not only be developed faster but also compared in a more even-handed manner. However, each approach brings with it certain challenges and these must be addressed in a more individual manner.

There are two main goals that must be achieved in order to make this implementation functional: a post must be parsed and returned as some form of tree, and these trees must be compared to each other. The first goal was achieved through use of the BLLIP reranking parser. This parser forms probabilistic estimates of the likely shape of the intended tree, and can then interpret these results to form the best possible representation of the given input. [17]

Comparison of these trees was pefrmed using the *pip* library `zss`, which performs tree edit distance comparisons in an efficient manner with regards to both time and space used using the Zhang Shasha algorithm[19]. The biggest challenge while developing this approach was parsing the output of the BLLIP reranking parser in such a way that it could be understood by the `zss` tree edit distance algorithm.

The BLLIP reranking parser returns a string for every input, and this string takes the form of an S-expression[4]. To ease understanding a short example is shown below, and it will be reused as the implementation progresses:

```
(S1 (SINV (ADVP (RB Here)) (VP (VBZ 's)) (NP (DT an) (NN example)) (.  .)))
```

The first part of each bracketed section gives the code associated with that node type, and the rest of each part gives either the original text or the children of that node. This form is readable and in theory tree edit distance could be performed using these representations, but in order to make use of established libraries this data is interpreted by a series of parsers.

The first of these parsers was found at [18], it takes the original string and returns a list where the first element of the list is the first item in the S-expression, and the other elements are either the original text or the children of that node in the same form. This function recursively goes down each branch of the tree and - for the same example as used above - returns the following:

[4]Given the relatively archaic nature of S-expressions and LISP - the language that popularized them - it is unclear whether returning data in this format was intentional or purely coincidence but in either case they can be read in the same way.

```
['S1', ['SINV', ['ADVP', ['RB', 'Here']], ['VP', ['VBZ', "'s"]], ['NP',
            ['DT', 'an'], ['NN', 'example']], ['.', '.']]]
```[5]

The data is now stored in such a way that it can be iterated over or otherwise acted on in the usual way within Python, but this is still not a form that is accepted by `zss`. This library does, however, provide a number of importers that the current list can once again be parsed into. No off-the-shelf parsers exist that can convert the tree from the current form to any that are importable, so one was built from scratch. It was decided that the easiest available form was a dictionary, because this allowed all computation to be performed using standard Python libraries and functionality.

This parser initially checks the type of every item in the list, and if all but the first item of the list are themselves lists then this is an intermediate node[6]. In this case the parser labels the node as part of the dictionary, then moves on to perform the same steps on all but the first element of the current list and inserts those as children of the current node in a recursive manner. The only other possibility is that every item in the list is a string, meaning the current element is a leaf node. These are added to the dictionary with the Part-of-Speech code used as the label and the piece of original text stored as additional information. This is illustrated in the continuing example below:

```
{'children': [{'children': [{'children': [{'b': 'Here', 'label':
'RB'}], 'label': 'ADVP'}, {'children': [{'b': ``'s'', 'label': 'VBZ'}],
    'label': 'VP'}, {'children': [{'b': 'an', 'label': 'DT'}, {'b':
'example', 'label': 'NN'}], 'label': 'NP'}, {'b': '.', 'label': '.'}],
                    'label': 'SINV'}], 'label': 'S1'}
```

While this is significantly harder to read and understand as a human, the `zss` Dictionary Importer can take in data in this form and convert it to a traditional root-node-leaf tree structure. This tree is then added to a list, and all other posts from that set are processed in the same way. The full process is then repeated for every post in the second data set, and their trees are added to their own list.

Once all trees have been formed, they can then compared to each other using the `.simple_distance()` function provided by `zss` and the within-and-between strategy as described in 2.1.2 and Figure 2.6. The `.simple_distance()` function takes the two trees that are to be compared as arguments and another parameter function called `nodeDist()`. This function is used to determine the value added to the edit distance total for the tree when a substitution - replacing the label of one node - is made. The default algorithm used is the string edit distance between the labels, but for these comparisons that metric is inappropriate. It is therefore replaced with a simple function that always returns the integer 1. This means each addition, subtraction and subsititution will carry a value of 1 as in the list edit distance in 2.2.3.

The same techniques used in 2.2.2 and 2.2.3 that allowed the bulk data collection phase to be run from another Python script were employed again in this approach. This mean a seperate function imported and ran the code described through a `run()` function on a collection of files using the `glob` package. However, the tools used in this implementation necessitated some noteworthy design changes. The BLLIP reranking parser is created as an object rather than being a library of functions, and it can only be created once. In order to be able to use it repeatedly, it must be created in the outermost Python script - the script for collecting the data - and passed down through the subsequent function calls as an extra argument.

The substantial increase in time required for each account comparison compared to the previous approaches (detailed in 3.3) meant that considerably fewer total comparisons were possible. Trees were constructed for 200

[5]The actual returned value is another list containing a single element - the list shown - and this layer is removed later on in the implementation, but for ease of reading it was removed here.

[6]i.e. not a node with part of the original sentence directly attached

sets of posts, comprising 100 accounts split into 2 parts. The first part was then compared to the second part of the same account, then the second part of 10 other accounts.

In addition, the trees that were created were stored in a text file so that they could be reused instead of being recreated. The dictionary representation of each tree was written in as a string, and could be recalled by using the standard Python function `eval()`[7]. While this does provide lower execution times, it was designed primarily as a failsafe way to make sure that time spent on execution was not wasted. Total execution time was several days, and if the script were to fail for any reason these trees would be lost and need to be recreated from scratch.[8]

The change in control flow was made possible through the addition of `runSingle()` and `runDouble()` functions in the testing phase. The function `runSingle()` takes a single file of posts and builds a tree for every post, then saves those trees in the dictionary form shown earlier to a different file and calculates the mean edit distance between each of the trees in that set. It returns a tuple with the mean edit distance, the total time taken and a list of all trees generated. The `runDouble()` function takes two lists of trees as generated by `runSingle()` and computes the mean edit distance between each tree across either set, which it then returns.

The final test run was performed using the stored trees, and so the CSV file in which that data is stored follows a slightly different format. The mean edit distance within each set was not recorded as it was not computed at this stage, and so the CSV file has fewer columns and this must be accounted for when it is being read. However, if the trees were stored then that meant this information had to have been calculated in the previous test, so the full picture could be rebuilt relatively easily using a dictionary which stored the relevant information using the file names as keys. By combining these data sets, a complete picture can be built and the results can be analyzed.

[7]The use of this function in an unprotected manner is, of course, extremely dangerous in an open system. However, due to the research nature of this project and as this is used only to read from a file created by the system itself, it was considered an acceptable practice in this case.

[8]This was implemented only after the first attempt at running these large scale tests, in which the exact situation described occurred and so this safeguard was put in place to minimize the impact of it happening again.

Chapter 3

Analysing the Data

3.1 Bag-of-Words Results and Analysis

The output of all comparisons at the set-of-posts level are stored in CSV files, which are a very lightweight way of storing information in a simple manner. Python provides a `csv` library as standard that can read these files, but this library is only capable of returning the string representation of any data stored. It is, however, possible to cast from a string to an integer or a float in - using `int()` or `float()` respectively and so now the original data can be accessed as intended.

The important number to extract for this approach was the ratio of unique words in either set to words in both. This data is stored directly in the CSV file; the only calculations needed at this stage are to form a mean value of the number of unique words in each of the compared sets, then divide that number by the number of unique words that feature in both sets. This value is then added to one of two lists: if the two sets of posts that have been compared are from the same account they are added to a list called `selfRatio`, if they are from different accounts they are added to a list called `otherRatio`. Means and histograms[1] are then created from these lists, it was found that the mean similarity - the percentage of words in both accounts - between two sets from the same account was 26% and the equivalent measurement from across two accounts was only 9.5%. The histograms can be seen combined in Figure 3.1, where the blue graph shows the ratios found when different accounts were examined, and the green when both sets were taken from the same account. The histograms were overlaid because it is the differences in the shapes of the graph that can be seen in this view that give the clearest indication of whether or not the calculation has any predictive power.

It can be seen quite plainly from the histograms that there are differences in these data sets. The peak of the results from different accounts sits around 10% lower than the peak of those sets taken from the same account, and there is a much steeper drop in the likelihood of similarity when comparing two accounts with different authors. The differences in these data sets could be used to make predictions about the authors of new sets of posts: in broad terms, it can be seen from 3.1 that if the similarity returns a score of less than around 10% then it is unlikely that they have the same author, but if the score is anywhere above about 25% it can be said with a reasonable degree of certainty that they have the same author.

The exact likelihood of these predictions can be found by filtering both lists appropriately - so that either only values below 10% or above 25% remain - and then comparing the relative reductions in the length of each

[1] Both data sets saw a significant number of comparisons which found 0 similarity - when comparing across accounts that bar extends well beyond the top of the graph shown - but it is suspected that these are formed primarily of occasions where one or both of the initial sets of posts was very small and therefore contained very few words.

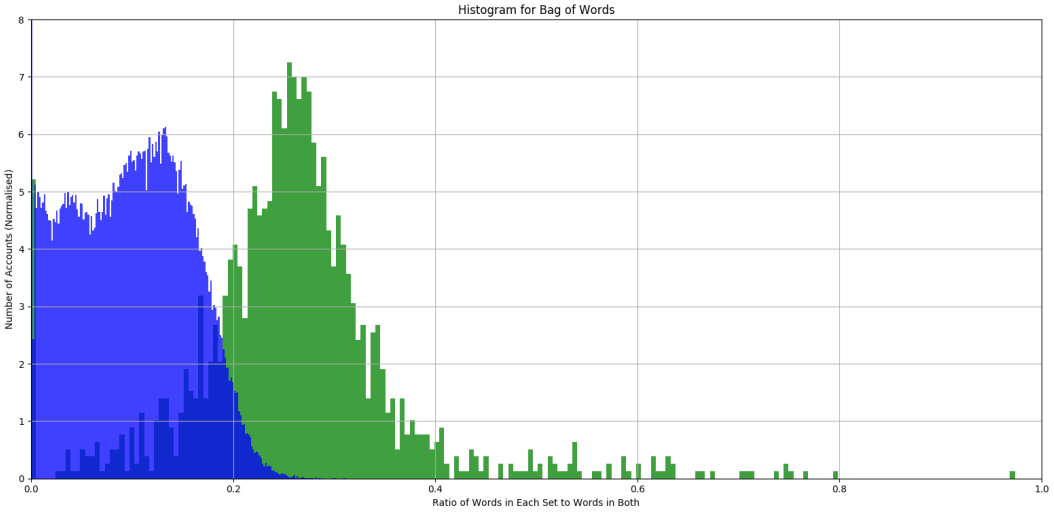

Figure 3.1: Overlaid Histogram of BoW Set Similarities.

list.[2] This technique was applied to the data and it was found that if the ratio found was below 10% then there was an 89.9% chance that the accounts were from different sources, and that if it was above 25% then there was a 99.8%[3] likelihood the accounts had the same source.

3.2 Parts of Speech Results and Analysis

As in 3.1, all the data to be processed exists in a single CSV file and so the same processes were initially performed in order to parse this data into a digestible form. However, the number of data points recorded for each result was increased in order to fully capture the increased complexity of the underlying approach. For every comparison of two sets of posts the mean length, mean edit distance and mean longest common subsequence of each set was recorded, as was the mean edit distance and mean longest common subsequence between each set.

3.2.1 Mean Edit Distance Comparison

There are several possible ways in which the mean ED can be compared, each with its own hypothesis about how authorship may be traced. It is possible that the *fingerprint* takes the form of certain structures being re-used, meaning that a comparison within a single account should deliver a smaller mean ED than one across two accounts It is also possible that the mean ED functions primarily as an indicator of complexity, and that the *fingerprint* can be found by looking at the differences between comparisons made within and comparisons made outwith a single account. In both scenarios, it is also likely that the mean length of each post is either a contributing or a confounding factor.

In order to establish the veracity of either of these hypotheses, the results must be compared in different ways. For the first hypothesis, the relevant factors to be compared are the mean ED in the between phase, both when the two sets are from the same account and when they are from separate accounts. Their histograms are compared

[2]These numbers were chosen in part by an eye test of the histogram and in part because they were close to the respective medians of each data set.

[3]The generated result is almost exactly 699/700, but in this case it has been rounded down so as not to overstate the findings.

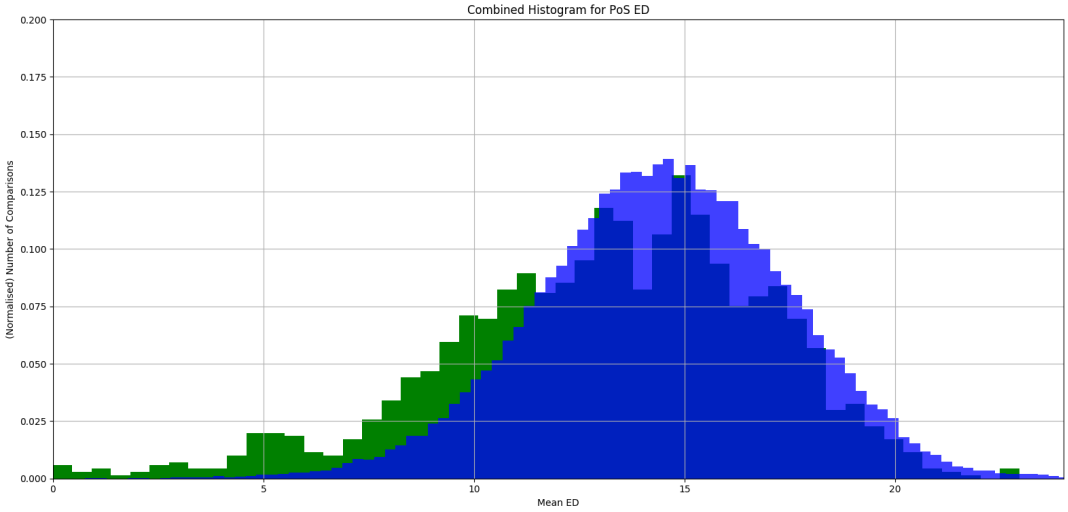

Figure 3.2: Histograms for ED.

using the same colour coding as in 3.1, where the blue graph represents the mean ED between accounts and the green represents the same value within a single account.

The mean value[4] of each set is different - 13.2 for a single account and 14.5 between two - with the within-account comparisons being slightly lower as expected. However, the variance visible in the graph means that it would be impossible to make accurate predictions based on this data as to the relative authorship of any new posts that could be compared. Adjusting for the mean length of posts in each set fails to provide greater insight: the relevant graph is included for completeness, but on this evidence it is impossible to say anything conclusive about the hypothesis that structures are re-used in a way that lowers mean edit distance.

The second hypothesis given above, which postulates that edit distance serves to measure complexity, can be examined by looking at the differences between the within phase and the between stage of each comparison. These values were calculated and from there these results can be split into two groups: those formed by operating on two halves of the same account, and those formed from different accounts. It is hypothesized that there will be a greater difference between the edit distances in the case where the two sets of posts compared are from different accounts.

The resultant histograms are shown in 3.4 and 3.5, and it can be seen that they represent quite different underlying data. As predicted the peak of histogram sits significantly lower when comparing two sets made from the same accounts, but the most useful feature of these graphs is the difference in the length of the tails. When comparing sets from within a single account there is a very tall peak and then the graph drops steeply, but when comparing across different accounts the graph moves down in a much smoother manner.

The predictive value of certain results from these calculations can then be examined using same process as in 3.1. It was found that if the average difference in mean edit distance is below 0.25 then there is a 98.5% chance that they come from the same account and therefore have the same author; and that if it is greater than 0.75 then there is a 74% chance that the accounts have different authors. These results suggest that the second hypothesis is correct: that users tend to make posts that have a certain level of complexity, and that this can be traced in order to form the *linguistic fingerprint*.

[4] As each item being compared at this stage is a mean edit distance, this value is the mean mean edit distance.

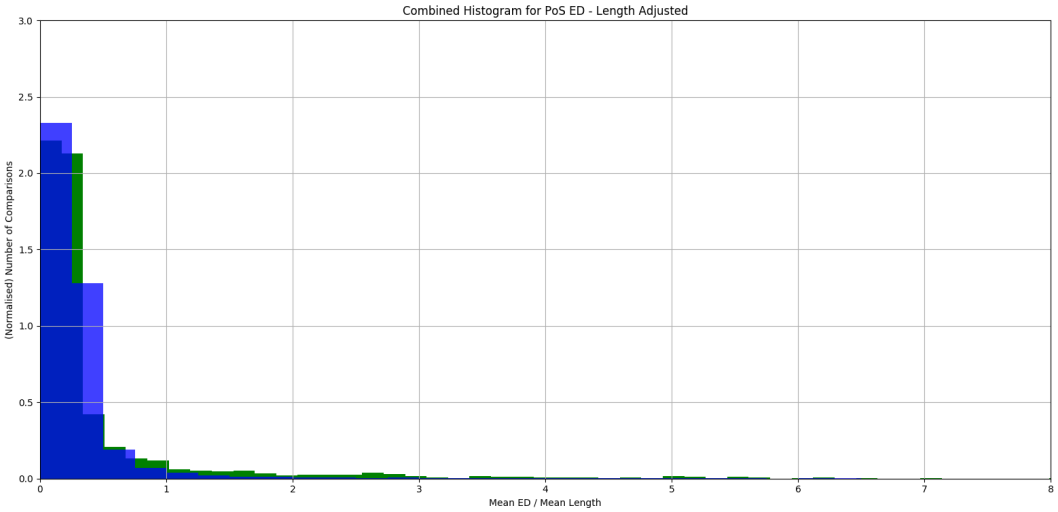

Figure 3.3: Histograms for ED, Length adjusted.

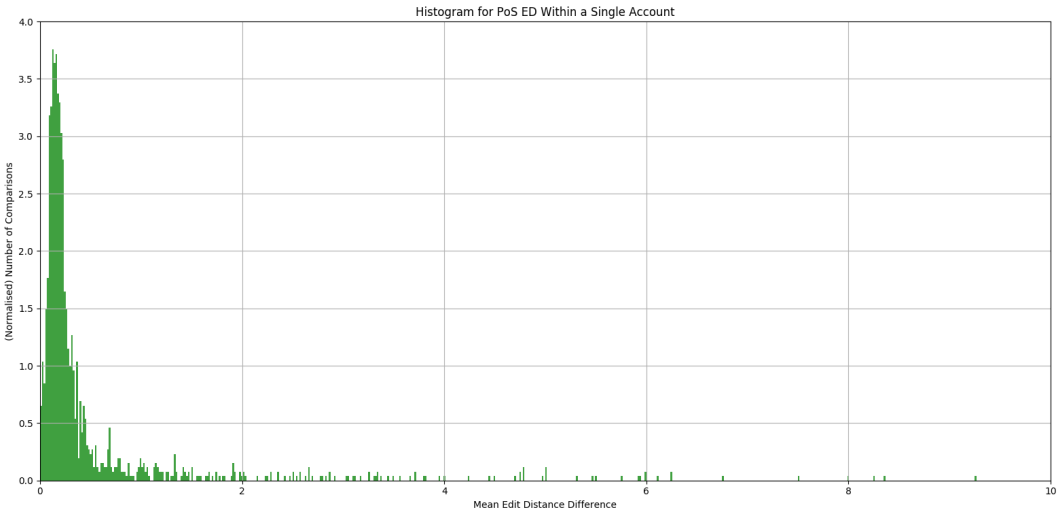

Figure 3.4: Histogram for ED Difference Within a Single Account.

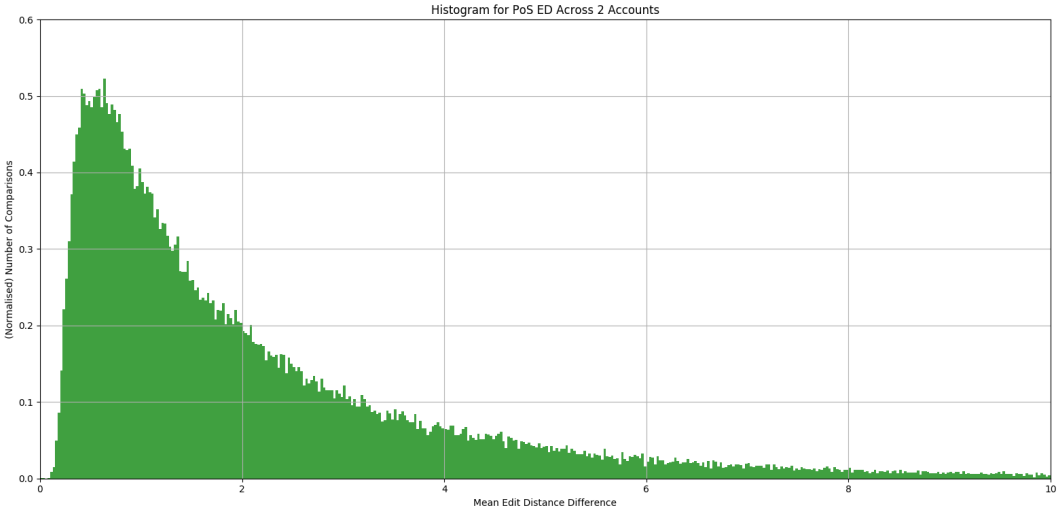

Figure 3.5: Histogram for ED Difference Across Two Accounts.

3.2.2 Longest Common Subsequence Comparison

An attempt was made in 3.2.1 to determine authorship via a measure that was designed around the idea that a user might re-use certain structures in their writing and that these would be traceable. That strategy was unable to give good evidence of the hypothesis, but this could also be traced in a more direct manner by using the longest common subsequence. The mean value of each data set suggests that there is some effect there to be uncovered, with the mean LCS within a single account being 4.6 and the same measurement between different accounts being 3.5. Histograms were also generated in the normal manner, and these are available at Figure 3.6.

It can be seen that although there is quite a large variance in both the data sets, the tail on the single account comparison stretches beyond the tail of the cross-account comparison. Using this it is possible to say that if a new comparison was made between two sets of posts and they had a mean LCS of above 6 then there is an 89% chance that they would have the same author. This is definitely an indication that there is some validity to the algorithm, but unfortunately this test would only be able to identify around 16% of instances where the authors are the same.

3.3 Tree Comparison Results and Analysis

The underlying principles used to compare the data created with tree comparison are the same as in all previous cases, but there are some additional complexities, mainly stemming from the disjointed way in which the data that needs to be accessed is stored. Before any comparisons can be performed, the mean lengths of each post set are loaded from the data stored in 2.2.3 to a dictionary that uses username as a key and the mean length as a value. In addition to this. the data must be moved into the lists for comparison in two stages; this is because the data was stored in two stages and in two slightly different formats. Another dictionary must be also be created, this one to hold the within-account comparisons for each set of posts. These are stored in the same format as in the dictionary of mean lengths discussed earlier.

The first comparison to make involves forming histograms and calculating means with the between-stages of the comparisons. As in each other approach, there is one histogram for two parts of a single account and one for comparisons across two parts of different accounts. In this instance the mean edit distance between one tree from

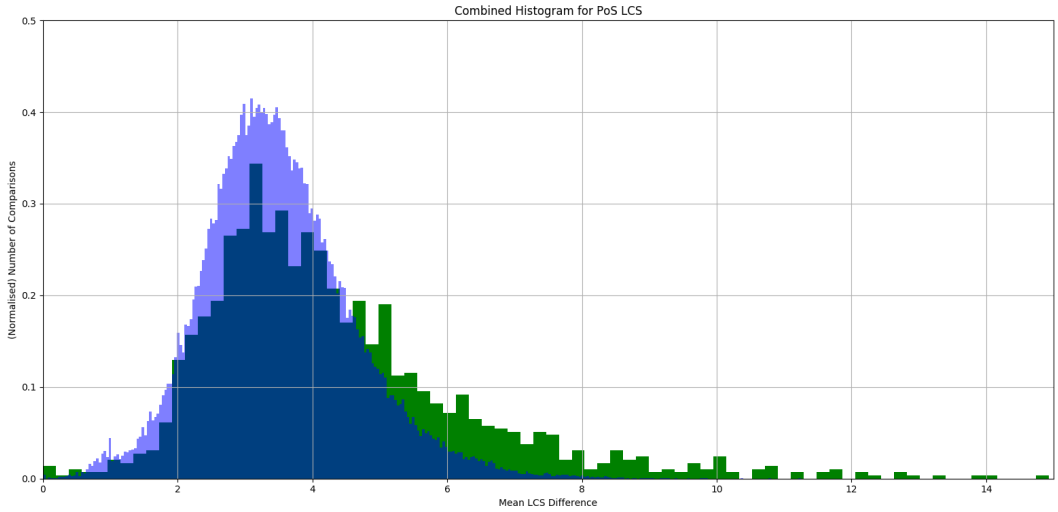

Figure 3.6: Longest Common Subsequence Histogram.

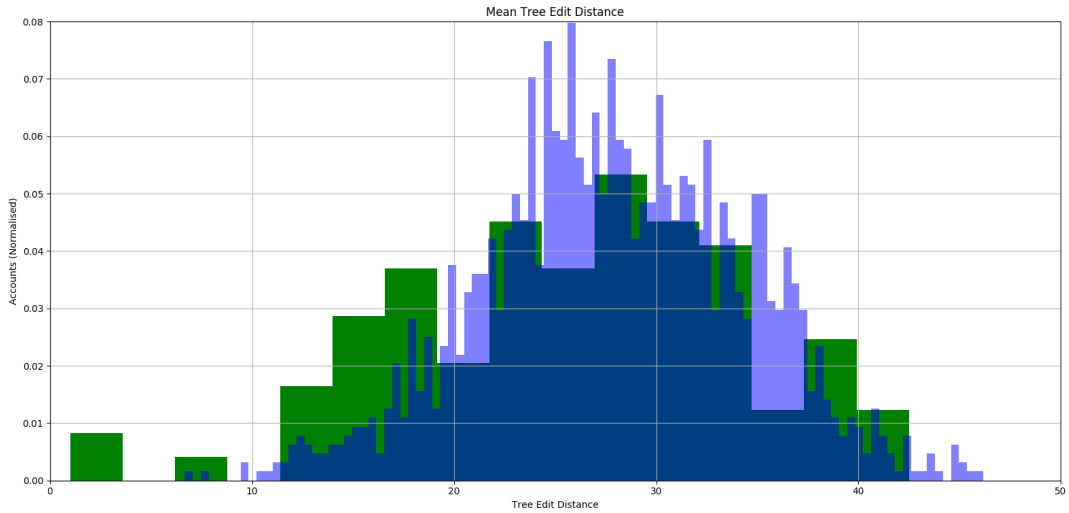

Figure 3.7: Mean Tree Edit Distance Histogram.

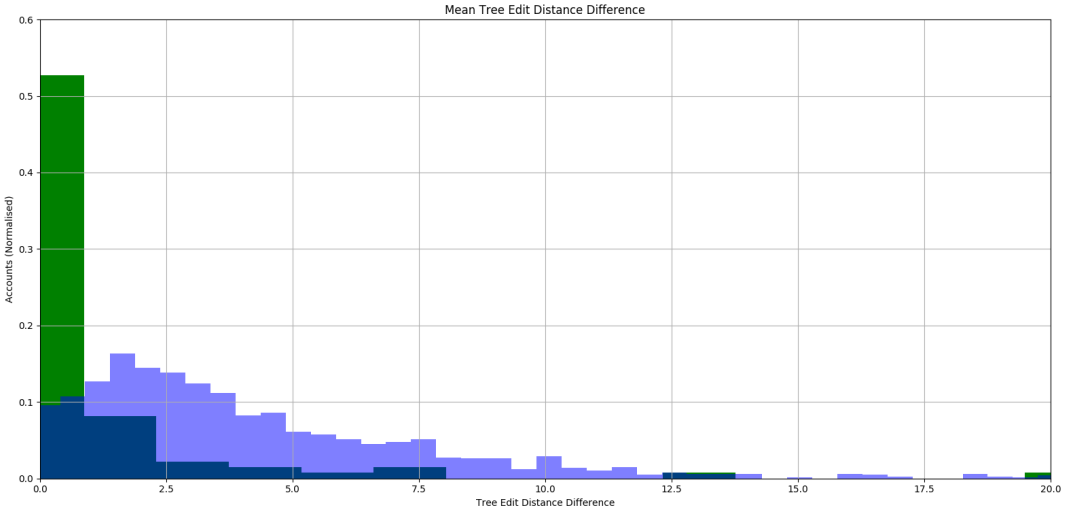

Figure 3.8: Mean Tree Edit Distance Difference Histogram.

one account and another tree from the same account is found to be 25.7, while the same result from one tree in one account to another in a different account is found to be 28.0.

These means match up to the expected theory in that it takes fewer edits on average to get from an arbitrary tree to another tree from the same account than it does to get it to a tree from a different account. However, again the variance in the data is large enough as to make the usage of this data alone for predictive purposes impossible. The histogram can be viewed at 3.7. These values are not clearly different but the Wilcoxon rank-sum[20] as executed by `ranksums()` in the *pip* package `scipy` finds the data sets likely to be from different distributions with a p-value of 0.02. This indicates that there is some validity to the original hypothesis, but that a different analysis is required if there is to be any significant predictive power for this system.

One such change in analysis performed was to divide the mean edit distance by the mean length of each post in the sets. The resultant means are 1.32 edits-per-part-of-speech for the comparisons between two sets from the same account and 0.64 edits-per-part-of-speech for comparisons across two accounts. This is the opposite of the expected result, but further analysis with `ranksums()` reveals that the variance around these values is so high as to make the different categories impossible to distinguish.

The differences in mean edit distances found during the within-and-between stages of analysis were also analysed in a similar manner to in 3.2.1. The results of each calculation were recorded and used to form two histograms, with one for cases where the two sets were made from the same account and one for comparing sets from different accounts. The resulting histograms are shown overlaid in 3.8, with the green histogram representing comparisons made between two parts of the same account and the blue representing comparisons between parts of separate accounts. It can be seen that the data sets form distinct shapes on the graph which can be used in a predictive manner. If the value for this calculation[5] is above 3, it can be said with 79% certainty that the two sets of posts have different authors and if it is below 1 then there is a 68% chance that the author is the same.

[5]The difference in the mean edit distance when moving from one post in a set to another post in the same set, and the mean edit distance when moving from one post in a set to another in a different set.

3.4 Run Time Comparison

The run time of each comparison from one set to another was recorded for all approaches. The mean times taken are shown in the table below:

Approach Used	Time Taken (Seconds)
Bag of Words	0.000 40
Parts of Speech	1.13
Tree (Full)	549.31
Tree (Half)	101.31

Table 3.1: Mean time taken to perform a comparison using various approaches.

Two values are given for Tree edit distance comparison; due to the fragmented nature of the data collection in this approach, it was considered necessary to split these results. The full result includes the time taken to construct all trees for both sets, then to compare within each set and across the two. However, the half result includes only the time taken to read constructed trees for each set in from file, then compare across these two sets. The full result gives a more accurate like-for-like comparison with the other approaches, but it was considered worthwhile including the half result for completeness and as a rough estimate of a possible upper bound on the speed of this approach.[6]

The Bag-of-Words approach runs in a time that is around two orders of magnitude faster than the Parts of Speech approach, and that itself runs around two orders of magnitude faster than the either the full or half Tree comparison. Speed was not considered a major concern at the outset of this project, but it is clearly beneficial in any situation where it is desirable to move through large amounts of data. There are a few possible ways in which each approach could be made more time-efficient: these include implementing the data-saving approach used in 3.3 for both the Bag-of-Words and the Parts of Speech. This would store the data used in each approach before it is compared, so that it could be pulled directly from memory or from a file rather than being re-generated.

It is also possible that some individual post comparisons could be skipped without affecting the accuracy of results. This could be tested by examining the results found when performing every possible comparison to ones where only say 80% or 60% of posts in each account were considered. A similar experiment could be performed by placing limits on the possible number of posts from each account, so that only the first 30 posts would be considered, for example. However, such experiments would require either a major restructuring of the functionality of the whole system, or a significant outlay of time as each possible permutation was run, and so these experiments were not performed.

[6]That is not to say that it is possible to achieve near this time, just that it is likely impossible for it to be faster, even with a highly optimized version of this approach.

Chapter 4

Conclusion

4.1 Effectiveness of Approaches

4.1.1 Data Selection Process

Selection of appropriate data is vital to testing the hypotheses in a way which presents any value to the outside world. The ideal data set for these experiments would include a number of separate accounts that were known to be run by the same user. The accounts that these algorithms are created to detect are impossible to use as positive results because they are, by design, hidden, and so these positive results must be constructed by another method.

Splitting individual accounts in two provides the necessary positive matches, but it is not a perfect analogue to the intended use case. The content that a user discusses influences the words chosen, as do the social circles the user moves in and their general temperament. An ideal data set would be able to control for these variables somehow; in the real world the accounts that are being targeted would need to be detectable even just when looking at other accounts that match their profile in any number of ways. However, automating the profiling of accounts in this manner would likely be complex enough to form a whole project in itself, and even then it would be difficult to quantify how much this would improve analysis.

In addition, it can be assumed that the puppet accounts that are being used will be actively attempting to conceal the identity of the author. The users concerned are being actively deceitful and so it is highly likely that if they were aware that, for example, their word choice was being monitored, they would be able to avoid reusing certain words or phrases. This effect is also not examinable with the available data set, but again it is hard to imagine a realistic way in which this could be implemented.

The data set used is not perfect, and so it cannot be said with certainty that the results found would transfer in an appropriate manner to the intended use case. However, the data set created has proven capable of, at minimum, comparing the relative effectiveness of the various approaches, and it is likely to be extremely hard to create one that improves on the given challenges in any meaningful way.

4.1.2 Bag-of-Words

The design of the Bag-of-Words approach was by a considerable margin the simplest, but it was found to produce some of the most compelling results. This is in part because a simple hypothesis is likely more easy to investigate than a more complex one, but there is also a significant amount of predictive power suggested by these results.

That being said, it is also likely to be the easiest to evade if the user that is operating multiple accounts knows that test is being performed. It is also the least novel of the approaches suggested, having been previously investigated in [21] and [22], but it was included to serve as a yardstick by which the overall system and then the other approaches could be measured.

4.1.3 Parts of Speech

This group of algorithms represents a novel approach to the challenge of determining authorship of a Twitter account. By removing the content and focusing on the linguistic structure employed by the writer, it was thought that the authorship could be established in a way that was more innate and therefore harder to evade. A reasonable amount of success was found in this approach in terms of determining authorship within the data set, but for reasons explained in 4.1.1 was impossible to inspect the algorithms ability to detect those who may be trying to conceal their identity. The validity - and from there the power of predictability - discovered in each algorithm can serve to confirm or deny the original hypotheses, and also give indications as to which of the algorithms would be of most use in the real world.

This approach made use of some simple natural language processing techniques, but these were primarily used in a black-box manner where very little knowledge of the actual underlying processes were required. It is also noteworthy that the Parts of Speech tagger used is not perfect and also was not designed with Twitter users in mind.[1] These two factors combined to make mislabeling of parts of speech within a post an inevitability, but it was found during initial testing that in the vast majority of cases the labels were at least acceptable.

The initial hypothesis, that it would require fewer edits on average to go from one post to another post with the same author than from a post to another post with a different author, was unable to be used in any meaningfully predictive way. This is not to say that the hypothesis was false, an effect could be seen both in the mean which, as expected, was lower when comparing parts from within a single account. There was, however, a large enough variance in the data that it was unfortunately impossible to place any kind of separation between the two distributions.

One aspect of this hypothesis that did seem to provide significant value was the difference between the mean edit distance within each set and then across the two sets that were to be compared. When this result was examined it was found to be able to positively identify - with a high degree of accuracy - a significant number of the occurrences where the two sets had the same author. This algorithm seems to provide one of the most defined *fingerprints*, and has definitely shown itself to be worthy of further investigation.

Alongside the initial hypothesis that the edit distance would be lower when the author was the same, it was posited that the mean longest common subsequence would be greater in the same circumstances. The validity of this effect was confirmed, and it was found that in some cases its predictive power could actually be usable. Unfortunately these cases where the predictive power is significant are in the minority, but this does indicate both that further investigation may be valuable and that this tool would make a good part of a broader approach.

4.1.4 Tree comparison

The theoretical understanding of language that informs the decisions made when comparing Parts of Speech also informs the decisions made when comparing trees. In fact the processes are in some senses almost identical, with one black box providing the analysis of each post and then another black box which can compare these.

[1] Since performing these experiments I have found a tagger that works in broadly the same way and is designed to work with the data generated by Twitter users. This was unfortunately discovered at too late a time to both implement this change and then perform any significant data collection. Implementing this tagger would however definitely be part of any further investigation into the topic as a whole.

In practice, however, the tree comparison took considerably more time to implement, maintain and test. Taking the output of the BLLIP reranking parser and converting it into a form that could be read by the `zss` package represented by some margin the largest technical hurdle encountered in the project. It was also hoped to be able to implement a form of checking a pair of trees for their longest common subtree, but as this would have to be implemented from scratch it was left out of the final project.

The results found when using this approach did suggest that there was some truth to the original hypothesis but given the extremely slow execution it is hard to recommend this approach for use in any further system. This execution time meant there was significantly less data collected on this approach; this is most obvious when looking at the histograms formed between two parts of the same account. The quality of this histogram is significantly worse than any of the other graphs in the rest of the report, and while this could have been fixed in part by performing a higher ratio of these comparisons to those with sets from different accounts, it is indicative of the increased difficulty found working with and understanding this approach.

In short, while there has been some validity seen in these results and a certain amount of predictive power demonstrated these are in no way proportional to the increased amount of work required to interact with this system and the increased processing power required.

4.2 Further Work

There is, as is the case with most research, a significant amount of relevant work still to be done in this area. This could take the form of further investigation into any of the given approaches, whether that was to increase predictive power or increase speed, or working to combine the approaches into one more comprehensive solution. It can be seen that tree-based comparisons are very slow and are an over-engineered solution to the problem at hand but there is possibly significant value in comparing the linguistic structure of users in a single dimension ie. as a list, as in the Parts of Speech approach.

Possible improvements to the Bag of Words approach might involve recording the number of occurrences of each word, so that any major increases in the usage of certain words could be tracked. This would involve an increase in the amount of overhead needed in both space and time. A full implementation of this approach would require a regularly updated dictionary of the baseline occurrence of each word, and the calculations to compare each account would need to be more complex in order to account for this new information.

It may also be beneficial to use the Parts of Speech generated by the tokenizer rather than simply splitting each post on spaces. When this approach is considered individually this would add a significant amount of processing time for an uncertain amount of benefit, but if the Bag of Words were to be combined with the Parts of Speech - as is discussed in more detail shortly - this information would already be available. Tracking and comparing pairs and triples of words may also be of interest, however this metric was tested in [22] and they these to be marginally less useful than individual words. However, this experiment was conducted on text in another language and from a different site so it is possible that this technique would still be useful for English-speaking Twitter users.

This technique of tracking short runs of words may also be applicable in the Parts of Speech approach, it is possible that this could track the re-use of certain structures and that these could lead to a positive identification of authorship. This idea was explored through the measuring of the mean Longest Common Subsequence but it is possible that would provide improved, or even just different, insight into the problem. Further work on the internal processing of this approach could include forming a dynamic table that stored each list of parts as it was created, saving the time currently required to recreate this information on each run.

However, it is in the combination of the approaches that have been evaluated so far that the most useful further work could be done. No investigation has been made into how these approaches intersect, but if it were

possible to view them in aggregate it seems reasonably likely that this could provide significantly greater insight. It is possible to imagine a final program into which the usernames of two Twitter accounts could be entered, and that could then return the probability that these two accounts had the same author. The output of this program could then compared to the results generated through the use of tools developed in [21] and [23], and/or their approach could be incorporated to hopefully form an even clearer picture.

Considered as a whole, this project gives novel insight into an area of research that is only going to become more important as social media becomes increasingly central not only in the lives of some individuals but in the socio-political fabric of humanity worldwide. It presents a novel approach to solving the problem of detecting authorship in social media accounts that, as far as can be ascertained[2], has not been tested and provides a reasonable level of positive results. While the validity of a final product was not able to tested as a whole, there are a number of useful possibilities and tools presented for those who wish to continue this work moving forward.

4.3 Acknowledgements

I would like to thank Inah Omoronyia for his excellent help and guidance as a supervisor, and Erin Ross for her proofreading skill and general support.

[2]It is possible that experiments similar to these have been performed behind closed doors at any number of social media sites, but if this is true then as of the time of writing they have not been released for public viewing.

Bibliography

[1] Maeve Shearlaw.
Egypt five years on: was it ever a 'social media revolution'?.
The Guardian, 25th February 2011.

[2] Jennifer Dickinson.
Prosymo Maksymal'nyi Perepost! Tactical and Discursive Uses of Social Media in Ukraine's Euromaidan.
Ab Imperio, 3/2014:75–93, 2014.

[3] Lev Grossman.
Iran Protests: Twitter, the Medium of the Movement.
Time, 17th June 2009.

[4] Peter Beaumont.
The truth about Twitter, Facebook and the uprisings in the Arab world.
The Guardian, 25th February 2011.

[5] charlottelennox (pseudonym)
The Ms.Scribe Story: An Unauthorized Fandom Biography.
Journalfen (archived), February 2007.

[6] Scott Shane.
The Fake Americans Russia Created to Influence the Election.
New York Times, 7th September 2017.

[7] Robert Booth, Matthew Weaver, Alex Hern , Stacee Smith and Shaun Walker.
Russia used hundreds of fake accounts to tweet about Brexit, data shows.
The Guardian, 14th November 2017.

[8] Emilio Ferrara.
Disinformation and Social Bot Operations in the Run Up to the 2017 French Presidential Election.
First Monday, 22(8), 2017

[9] Rebecca Crankshaw.
The validity of the Linguistic Fingerprint in forensic investigation
Diffusion: the UCLan Journal of Undergraduate Research, Volume 5 Issue 2, December 2012

[10] Malcolm Coulthard.
Author Identification, Idiolect, and Linguistic Uniqueness
Applied Linguistics, Volume 25 Issue 4, December 2004

[11] Unknown.
Top Websites in the world
SimilarWeb, 1st February 2018 `https://www.similarweb.com/top-websites`

[12] Aliza Rosen.
Tweeting Made Easier.
Twitter Blog, 7th November 2017
https://blog.twitter.com/official/en_us/topics/product/2017/tweetingmadeeasier.html

[13] Amy Reynolds.
Drawing Sentence Syntax Trees.
Ling 101 Online,
http://amyrey.web.unc.edu/classes/ling-101-online/tutorials/how-to-draw-syntax-trees/

[14] Sanjeev Khanna.
Approximation Algorithms for the Largest Common Subtree Problem.
Stanford University Technical Report, 1995

[15] Martijn Pieters.
Python: Length of longest common subsequence of lists
Stack Overflow, 3rd July 2014
https://stackoverflow.com/questions/24547641/python-length-of-longest-common-subsequence-of-lists/24547864

[16] Mark Liberman.
Alphabetical list of part-of-speech tags used in the Penn Treebank Project:
Linguistics 001: Introduction to Linguistics, Fall 2003
https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[17] Eugene Charniak and Mark Johnson.
Coarse-to-fine n-best parsing and MaxEnt discriminative reranking
Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, 25th June 2005

[18] Unknown
S-expression
Wikipedia, 2nd May 2015
https://en.wikipedia.org/wiki/S-expression#Parsing

[19] Kaizhon Zhang and Dennis Shasha.
Coarse-to-fine n-best parsing and MaxEnt discriminative reranking
Siam J. Comput., 12451262, December 1989

[20] Clay Ford.
The Wilcoxon Rank Sum Test
University of Virginia Library: Research Data Services + Sciences, 5th Jan 2017
http://data.library.virginia.edu/the-wilcoxon-rank-sum-test/

[21] Rachel M. Green and John W. Sheppard. Comparing Frequency- and Style-Based Features for Twitter Author Identification
Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference, January 2013

[22] Marcia Fissette. Author identification in short texts
Bachelors Thesis, Department of Artificial Intelligence, Radboud University, 2010

[23] Antonio Castro and Brian Lindauer.
Author Identification on Twitter
Stanford University 2012
http://cs229.stanford.edu/proj2012/CastroLindauer-AuthorIdentification
OnTwitter.pdf