Software Using Stylometry to Detect Differences in Authorship

Michael J. Bleakley, Michael J. Wise

Word Count = 7763 Words

1 Abstract

With contract cheating (more colloquially known as ghostwriting) on the rise in academic institutions, this paper describes an open-source program to aid academics and institutions in combatting and potentially resolving this issue. Specifically, this program should be able to identify if the attributed author did not write an academic paper.

Methodology: A program was developed that couples a neural network (using tensor flow 2.0) with the linguistic science of stylometry. Stylometry is the attribution of values to aspects of literary style and, in this case, was used to develop a set of metrics. These metrics were then passed to a neural network, to train it to recognise the literary style of an author. A new document can then be parsed to this model to determine if its authorship is legitimate or not. To achieve such a result, two or more documents were parsed into the program and split into segments based on a distribution. Then the metrics are developed for each segment and used to train the network. This model was run with each document as the test data to output the median prediction value.

Results: The initial results for this program were positive. An interesting finding is that the accuracy value the network produces is not necessarily the deciding factor, rather the difference between prediction values is essential. This difference is normally distributed; therefore, a threshold is defined, which is used to ratify the authorship of a document. A vital factor to note is that authorship of documents can only be accurately determined if the documents are of the same type. Saying this, when the documents are of a similar type, the model exhibits workable accuracy.

Conclusions: This paper and associated program show that it is viable to develop a program that can attribute authorship; however, there are some shortcomings and potential for further work. Prominently, the need for an *intelligent* user to aid in the running and production of viable results is not ideal. Further work could include the negation of this need or creating document type recognition based on similar metrics such that the program can run continuously and independently. Finally, using this program as the 'be-all and end-all' is not possible, and should preferably be used to aid in identifying suspected cases of academic misconduct.

1.1 Acknowledgements

The present research has been provided with extensive help from Associate professor Michael J. Wise, and my thanks are extended. Along with this, thank you to all the students who gave me documents to develop this program, none of whom wished for their names disclosed.

1.2 Code Repository

https://github.com/bleaksmb/Michael_Stylo

2 Introduction

Cheating, or rather academic integrity, has and always will be a prevalent issue in academic institutions. From plagiarism to collusion, the ways in which students can cheat are ever-increasing and coupled with the rise of the internet it is now easier than ever. Most institutions were quick to adapt to these changes, with services like Turnitin[1] being used to combat the most prevalent issues. One such area that is on the rise is the act of ghostwriting. Ghost-writing, which is the

act of hiring another person (or persons) to complete a piece of work for you, such as an essay for an assignment, and then attributing said to work fraudulently to yourself [2]. Much like how issues of plagiarism and collusion have risen with the internet, the availability of such ghostwriting services has also begun to rise. This rise is not insignificant, with a study from Swansea University (2014) in the U.K estimating that the overall growth of students and academics using such services increasing over 12% above the historical average of 3.5% [3].

Such a problem is not an easy one to solve either, as unlike the other instances of cheating, ghostwriting is much harder to prove empirically due to the more nuanced nature of style. This system seeks to provide academics and academic institutions with a program to combat this issue. This paper will seek to apply stylometry, which will be coupled with a neural network to distinguish between authors. Stylometry is a study of linguistics that describes an author's style by quantifying certain factors, rather than looking at general stylistic features. Using this, numerical values will be attributed to works submitted by current university students, to test the feasibility of developing such a solution. The code will be open source such that Institutions can apply the solution to help combat the issue and bring to light cases of academic misconduct.

This approach to the issue seems promising, with adequate prior researching showing that it is possible to determine authorship, using stylometry and a neural network. The issue that this program will face, when compared to previous solutions is the overall lack of data. Most previous papers, use large data sets such as the works of Shakespeare [4] or the Federalist papers [5], this paper will have to seek to provide a program that is robust to a small set of data. Initially, it seems promising to break works down into similar chunks, such that more data is available for both training and testing. This provides the ability not only to increase author attribution accuracy, but also homogeneity of data as elements such as document size and type have the potential to play a role in the data produced. The paper will present the methodology used to solve this issue, along with the relevant results.

3 Background and Literature Review

3.1 History

The history of stylometry as a measure of an author's style dates back over 200 years, initially using rudimentary methods such as average word length but progressing quickly to more mathematically significant measurements. In present times, access to large data sets and computational analysis has led to a more focused and accurate approach than ever before, whilst giving rise to more complex issues[6]. The paper *Surveying Stylometry Techniques and Applications* discusses and gives an overview of the way in which style attribution has changed, as well as the current problems in the field. The specific technique employed in this paper is that of authorship verification, which is one of the three main areas of focus (with the other two being authorship attribution and authorship profiling) [6].

Another key aspect covered by this paper is the fundamental nature of approaching a stylometry problem, or rather the necessary steps which are similar to those in which a traditional network is created. As the data is loaded in, processed, features are extracted, and then training occurs to develop a classification [6]. A notable key area of feature extraction revolves around the complexity of an author's writing. A notable issue that these types of approaches face is that no matter the

mathematical representation, the language of the text is essential. Therefore, even if the same author wrote texts if the language is different, the difference in calculations will be mathematically significant.

Finally, the analysis of recent research papers shows that machine learning and distance calculations are the two more prominent ways to analyse style computationally. More notably, when using machine learning the extraction of features (lexical, syntactical and character-based) is the primary method, whereas distance calculations tend to use character n-grams [6].

3.2 Important Metrics

When talking about stylometry and metrics, one of the main areas are metrics that seek to attribute meaning to the usage of specific words. George Kinsley Zipf and George Yule develop two of the most prominent metrics, these being Zipf's law, and Yule's Characteristic (K)[6]. Zipf's law follows the notion that human language, and thus word usage follow a relationship that is independent on the context of the language. This relationship is reciprocal as seen in Equation (1) where r is rank of the word based on its usage (rank one is the most, two is the 2nd most and so forth) and f(r) is the frequency of that word [7]. It is also important to note that this relationship follows a linear log-log relationship [7].

$$f(r) \propto \frac{1}{r^a} \tag{1}$$

The other key metric is Yule's K, which is a measure of the repetition in a text. It a measure that is used to represent the richness of the vocabulary of the author and has been historically used to attribute authorship in various studies [8]. The larger the value of K, the less complex the vocabulary of the text (author's vocabulary) is, and the inverse is true. Equation (2) is the representation of this characteristic. It is essential to note the usage of C in the equation, which is simply a 10^4 multiplier to avoid working with decimals.

$$K = C[-\frac{1}{N} + \sum_{i=1}^{V} V(i,N)(\frac{i}{N})^2] \tag{2}$$

3.3 Famous Problems: Merriam and Matthews

The use of stylometry in combination with a neural network is in itself, not a new concept and used previously to aid in distinguishing of works between Shakespeare and various other authors. Thomas Merriam and Robert Matthews authored the first two papers employing neural networks in the field of stylometry. Written in 1993[4] and 1994[9] respectively, the two papers *Neural Computation in Stylometry I: An Application to the Works of Shakespeare and Fletcher* and *Neural and Computation in Stylometry II: An Application to the Works of Shakespeare and Marlowe* set most of the foundations for studies that are released today.

Both papers define a standard procedure for applying a multi-layer perceptron (MLP) to work with irregular sized data sets and the ability to distinguish between two authors. In both papers, the authors train and develop their MLP using works from the author's core cannons, and trained the network setting the outputs of either author and then back-propagating to increase the prediction accuracy[4][9]. For example, in the first paper, they used works where authorship is known and not disputed, such as Romeo and Juliet for Shakespeare and Valentinian for Fletcher [4]. They

followed a similar process to as described in [6] for a primary classification which is suitable when distinguishing between two authors.

Imperative for a project such as the current study (in which data is limited) is the noted performance of the authors. They note that the performance of an MLP is consistent even when in the presence of statistical noise [4]. This is key as such noise will be present in the application of stylometry and neural networks to works of students. Another important aspect that both papers follow is the notion that the prediction accuracy is purely relative to the amount of data that has been input the network. For example, in the first paper, it is mentioned that the MLP performed more favourably on whole plays, due to there being significantly more data. However, they do note that the predictions on individual acts are influential and can be used to show the suspected collaboration of authors[4].

The main thing that will have to be altered for the research presented in this paper is how the network is trained to distinguish between authors. What this means is that, in the discussed papers, results that are around 50% are not insignificant and could indicate cooperation between the two authors for said pieces [9]. This will not be possible in this project as the detection is based on the null hypothesis such that the same author writes all documents. Therefore a new way of defining authorship will have to be developed over just the prediction range.

3.4 Similar Work

As the primary goal of this paper is to seek to address the issue of ghostwriting in academic institutions, it is worthwhile to look into other works in which like goals are assessed. The paper *De-anonymising Programmers Using Stylometry* seeks to address similar goals and looks at determining or verifying the authorship of pieces of code. The paper brings to light many similar motivations to this work, such as those of a professor seeking to verify that a student wrote their work (not a previous student), or seeking to disprove the claimed authorship without knowing the actual author. [10]

As stated above, when discussing vocabulary based metrics, the language is imperative, and such features are not independent of the language that they are written. This is interestingly a similar case as the language in which the code is written is important (e.g. C++ vs Java), but similar to other solutions, the same model can be used generally as long as it is trained using information from such languages [10]. Using a corpora from the Google Code Jam, the authors had access to thousands of codes of like solutions. This was the key issue was something as rigid as code tends to change drastically depending on the application of the piece of code, even if the authorship remains consistent.

How classifications were achieved was through the use of feature extraction primarily focussing on lexical and syntactic features. This is a similar approach to those used in [4]; however, the exact calculations used are different. For example, instead of looking at common word usage, the usage of keywords (if, else, do, while) was analysed. For the syntax-based approach, instead of looking at sentence length or looking at punctuation, the usage of tabs or spaces was used as this is a crucial aspect of code syntax [10]. It is important to note, much like the applications in this project and others pertaining to stylometry, the authors normalised the values over the entire data set such that the size of the data set does not play a factor in incorrectly identifying authors due to coincidence rather actual similarity.

The product is a program and methodology that can identify code authorship in as little as 550 lines of code. This is due to many factors, but the highly structured and *rule-based* nature of code allows the classification of more accurate metrics. This resulted in the production of a *Code Stylometry Feature Set*, which has the potential to develop thousands of features for a relatively small number (low hundreds) of authors [10]. A similar approach will be taken in this research paper, as several metrics will be chosen based on their qualities to be developed and attributed to authors.

4 Design and Development Methodology

4.1 Data Preparation

The first key issue to address is the nature and state of the documents that will be passed into the program. Initially, documents from university students have a lot of uninformitive data that is not required and has the potential to cloud the results from the network. Specifically, they can include but are not limited to.

1. Quotations from External Sources
2. Special Characters and Symbols
    (a) This is highly prevalent in papers involving mathematical formulae
3. Numerical Values
4. Cover Pages
    (a) For example, the university cover pages describing instances of academic misconduct
5. Images and Diagrams.

Therefore, data cleaning is a big part of this program. Initially, the data is parsed through the Python package textract[11]. This package enables the opening of more than 20 different file types and specifically the opening of Microsoft Word documents and PDFs . This is important because the raw file opening of python does not correctly read and encode such data. The other key factor of textract is the fact that it produces a byte encoded string. This is important due to the requirement of removing quotes and images.

As quotations are not written by the same author they have the potential to influence the stylistic features of the specific piece, especially if the quote is long. The use of a byte string representation is vital as it allows start and end quotes to be accurately defined, as well as the type of quote (double versus single). For this project, double quotes (") are the only ones defined as quotations due to the more widespread usage of double quotes for quotations, whereas single quotes are more commonly used for emphasis. The second key feature of the byte string is the ability to identify images only documents. An easy way to *skirt* the recognition software is to provide it with a PDF that is images of text rather than a pdf of text. An image only document includes a byte string which printers use to force the next sheet of paper, similar to a page break. The steps for this decoding process are found in the method *opening*. It performs the following.

1. reads the words in the file unit it finds a quote

   (a) if no quotes are found, decode string and add to the new file list

2. initialises a counter
3. count words until it finds either an opening or ending quote
   (a) if an opening quote found, new counter initialised for quote in quote
4. when it finds an ending quote it either
   (a) removes one from the quote in quote counter
   (b) signifies that a quote has ended
5. Once a quote has ended
   (a) it adds quote (quoteLen) to the new file as a label to be used in metric calculations
6. continue until the end of file detected

The final process is the program then saves the cleaned data as .txt files after their original name. This is performed as it not only improves memory performance but also means that if a .txt file is empty or invalid, it is identifiable before the following steps. Along with this, it allows the use of base python functions to open and read the file. There are two prominent cases in which an error will be thrown because of an unacceptable file. These are when the file cannot be read, which is usually because of a unique character which cannot be decoded (usually in a PDF), or when a file is blank (either empty or all images). Conveniently, in the first case, Microsoft Word can open PDFs and save them as a .docx. In the second case, the author has to be contacted for a new variant of the document as there is no simple way to obtain accurate information from an image.

4.2 Feature Extraction and Development of Metrics

One of the key steps in all processes pertaining to neural networks is the process of feature extraction. In this case, the features refer to numerical values that are created using principles found in stylometry. As previously discussed, stylometry is the numerical representation of a writer's style. A key thing to note about this representation is that the style has the potential to vary significantly based on document size. To avoid this, and to produce more data points for training, the files are distributed into same size chunks. These chunks are defined by a distribution and are uniform across all documents and based on the size of the smallest document.

The features extracted from the document are Yule's characteristic, Yule's I (the inverse of K), two that relate to Zipf's Law, quotation percentage, punctuation usage, sentence length, average word length and use of contractions. The most signification metrics are those that relate to Zipf's law and the values from Yule's Characteristic. In regards to Zipf's law, there is a significant reason why there are two metrics attributed to it, not just one. Even though human language is very complex, there is a high likelihood that there will be a close to one to one relationship when applied to a log-log scale [7].

Because of this, there is a chance that two writers could have a very similar value. To combat this, a second calculation is included for robustness. The difference between the two calculations it the initial one cuts out extraneous single-use words, and the other does not. The difference that this makes is relatively large as can be seen in Figure 1 and Figure 2 with there being a 5% difference between the two numbers. Although not huge, this decreased similarity between the calculations produced by the network and has the potential to be a defining factor in the differences between authors.

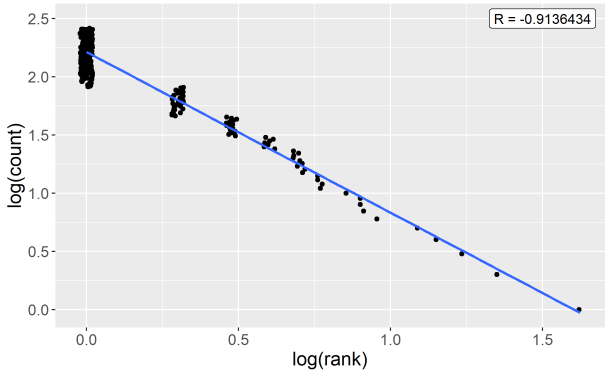

Figure 1: Zipf's without extraneous single words

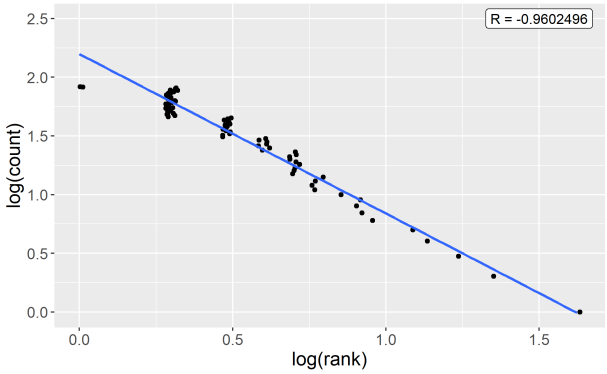

Figure 2: Zipf's with extraneous single words

Yule's Characteristic is calculated simply by using the formula described in Equation 2. Yules I, which is simply calculated using the inverse of Yules K, but can be developed using Equation (3).[12]

$$\begin{aligned} I &= \frac{V^2}{M_2 - V} \\ M_2 &= \sum_{i=1}^{V} i^2 x f_v(i, N) \end{aligned} \tag{3}$$

The last five metrics, as described above, are mathematically simple when compared to the two other metrics but cover crucial aspects. Although the number of metrics (9), may seem arbitrary at first, the premise behind their choice is due to their interrelation with other metrics. For example, both formulae from Yule and Zipf relate to the richness of language or the use of vocabulary; therefore, none of the other metrics should cover this. The percentage of quotes shows the preference of an author to quote rather than paraphrase. Punctuation count covers commas, question, exclamation marks and semicolons. Commas are essential here as they can extend sentence length but highlight the use of certain words such as However and But. This couples nicely with sentence length and word length, which are more intuitive stylistic features. Finally, there is the contraction percentage which develops a value based on the usage of apostrophes and hyphens to shorten or conjoin words. This is a useful statistic as it shows whether or not the style of the writer is more academic or conversational.

In total, these statistics cover vocabulary, the regularity of writing, sentence length, sentence structure, word usage, nature of writing and propensity to cite. It is important to note that more metrics could be added, however through testing in network development, there was a depreciating return on the number of statistics, which resulted after the current choices. The usage of the metrics is to create a fingerprint of an author. The application of the fingerprints is that they are

randomly chosen and passed into the network as alternative training data, such that the files being authenticated are only used as positives. Providing more data to make the network less prone to returning false positives, as well as making the predictions more accurate overall. Table 1 shows a sample of one of the resultant fingerprints and shows five chunks worth of data.

Yules K	*Yules I*	*Zipfs 1*	*Zipfs 2*	*Quote %*	*Punc*	*Sentence*	*Wordlen*	*Contract*
121.32	76.14986	0.97487	0.904113	0.024	0.038	18.74	4.806	0.009
99.68	91.17433	0.96235	0.888941	0.049	0.061	18.76	5.159	0.01
140.19	66.58233	0.952198	0.880467	0.201	0.059025	8.352	6.018	0.00940975
71.56	122.6091	0.957061	0.888872	0.238	0.078409	20.5641	4.705	0.00340909
63.66	135.7589	0.963921	0.898838	0.014	0.036	20.25532	6.324	0.011

Table 1: Fingerprint Sample

4.3 Network Development

This is both the most simple yet most important facet of this project. Developing using Tensorflow 2 and Keras [13], a fully connected, five-layer neural network (3 hidden layers) was developed using scaled activation exponential units (hereafter SELU) on all layers except for the last layer, which uses a sigmoid to scale the value to either 1 or 0. The network was created in an iterative process in which combinations of different layer numbers, sizes and activations were used to find the best possible result. To develop this network, the fingerprints, as mentioned in the last section, were used.

The fingerprints were prepared using chunk sizes of 250, 500 and 1000 such that the network could be tested on varying chunk sizes to ensure a high accuracy no matter what the chunk size. 500 words was the best performer across the testing sets even though logic would suggest that it should be that 1000. The main reason why 500 performed the best was that it was consistent over all the authors, with 1000 losing some information in smaller documents from certain authors. The main issue that this kind of problem or application has in regards to neural networks or machine learning is the overall small sample set of data. As the network has to be trained every time to develop a model that can accurately discern between two people, then the amount of data will always be relatively small. Initial testing was completed with only six of the nine metrics and achieved a median accuracy of approximately 65%. The addition of the extra three metrics brought the median score up a minimum of 10% across all test sets.

The critical thing to note is that the considered value for this project should not be the accuracy of the network as a whole (across all chunks).Not only does this not vary greatly (due to small amounts of data), but it also does not tell the whole story. As tensor flow calculates the accuracy based on the value either being 0 or 1, 0.49 and 0.51 provide different return results even though their predicted value is 0.02 apart. The value the network computes in the final iteration of the program is using is the median prediction value, not accuracy. The prediction value is a value between zero and one relating to how similar the test data is to one or the other. The decision between zero and one to signify the null hypothesis was made after testing each of the people's fingerprints, with 500-word chunks, five times. This resulted in approximately 15,000 data points. These data pointed were plotted and can be seen in Figure 3 and Figure 4. Although the highs were relatively similar between the two, zero was much more consistent when set as the assumed author

resulting in a median prediction of 0.00807 whereas when one was set as the assumed author, it resulted in a median prediction of 0.83625.

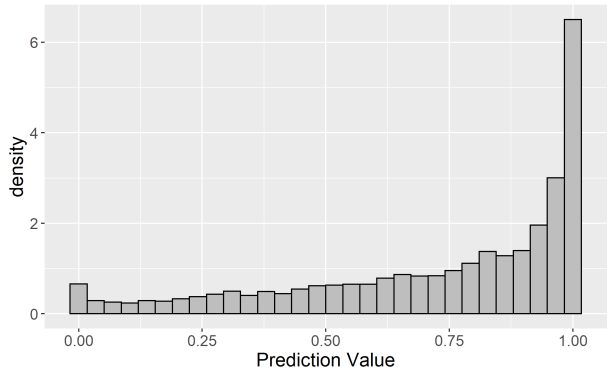

Figure 3: One as null hypothesis

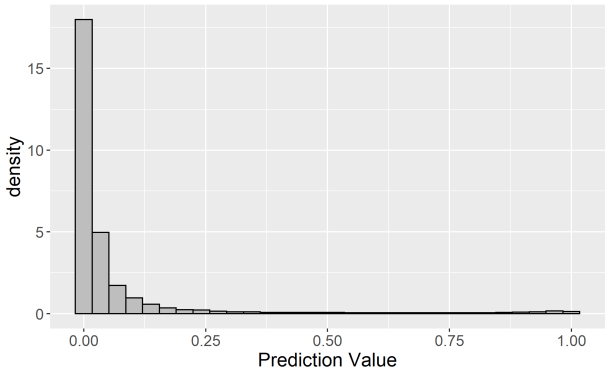

Figure 4: Zero as null hypothesis

4.4 Program Development

The final step in the designing of the system was to move all the developed functionality as described in the last three sections and combine it into the final program. The first thing that had to be altered was how the neural network performed its tests. Initially, in the proposal, the runtime for the network was as follows.

1. Load in 5x 500-word documents
2. Train the model using the first 4
3. Test the 5th document against the model and get the result
4. Repeat the process four times, swapping the 5th document
5. Map all the results on a Kaplan-Meir Curve
6. Output the Expected Author of the Papers

The initial issues that arose with this case were not only was that the amount of data insufficient to achieve meaningful predictions, but rather there was no good way to accurately determine whether or not a different author wrote a document. The amount of influence that a document written by another author had (in this version) was not significant enough to change the resultant outcome. Along with this, documents would either test very well or very poorly without being predictable. The workaround for this was using the fingerprints that were developed from the original authors as the counterexample to the papers being processed. The purpose of this is to set the documents being loaded in as zero, and the fingerprint data as one. This means that there is effectively twice the amount of data being loaded in to train the network. Therefore, the new process is as follows

1. Load in Documents (minimum two at 500 words length)
2. Train the model using n-1 documents(0) and a random fingerprint(1)
   (a) Len(n-1) = Len(fingerprint)

3. Record Predictions based on the final document(0) and save
4. Repeat for all documents 20x
5. Solve for median prediction value for each document

Another aspect that that had to be altered was the overall performance of the program. The program can run very well when a device has a CPU with high clock speed and good single-core performance. With the rise of smaller, multicore processors (especially on laptops) a better performing program will be required if it is to be widely adopted as an aid into catching ghostwriting. Therefore, to improve this, the code has been parallelised to enable faster performance. On average, this provided a 3x speedup with an iteration which would typically take 15 minutes to run taking approximately 5. Finally, Figure 5 shows the runtime of the program from loading in the documents to the output of a prediction value.

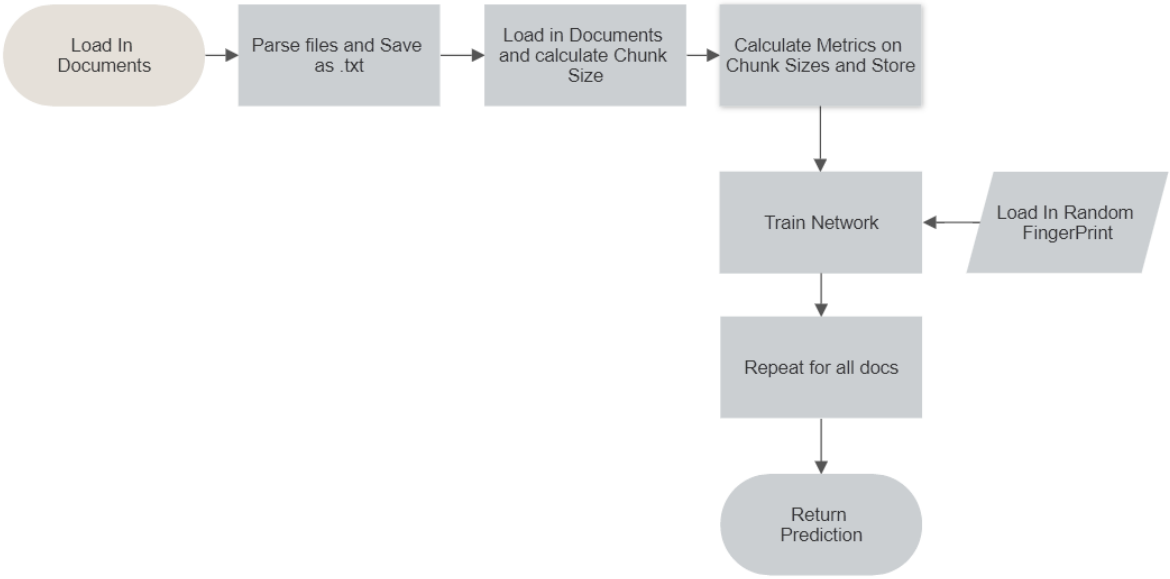

Figure 5: Program Runtime

Finally, the most important new calculation to be included in the final program was the chunk size of the documents. There was initially two-ways in which this could work. Either, compute a distribution that works on all documents or have a pre-run process that determines the choice with the highest delta between documents. The pre-run process worked well but had times where it was inconsistent with chunk sizes given. Therefore, the decision was made to make a distribution that defined chunk size.

The tests were run on 3 document sets of varying contexts. Initially, tests were run to find a suitable candidate fingerprint (to keep consistent) at a previously found, accurate chunk size. Then, tests were run varying the chunk size from a minimum of 100 to a maximum of 500 words per chunk. The data was then empirically analysed to find the best word size. These values were

mapped against the smallest document size in the set, and the word chunk size to give the linear distribution in Figure 6 and thus, chunk size can be solved as per Equation (4).

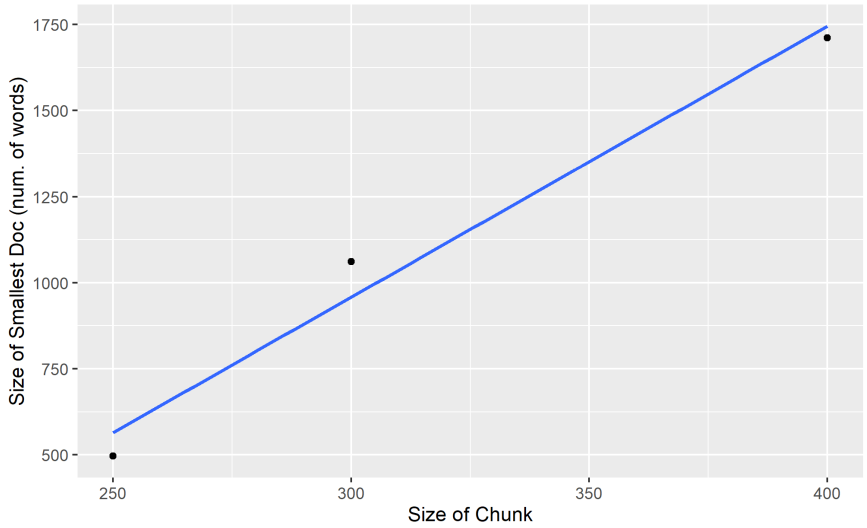

Figure 6: Chunk Size Calculation

$$\begin{aligned} n_{raw} &= 0.124214 min_{doclength} + 181.356 \\ n_{chunk} &= round(\frac{n_{raw}}{50} \times 50) \end{aligned} \tag{4}$$

5 Results and Discussion

5.1 Prediction Mechanism

Through the various round of testing, it was proven that the program was able to produce a significantly different value between documents that were written by different authors versus those that were written by the same person. However, the biggest issue is that these were not consistent. It would make sense that the documents that were statistically different would predict at a value closer to one, but what if the randomly chosen fingerprint has a similar style to the author of the papers in question? After more analysis, the important value is not necessarily how closely it predicts to one or the other as if it is different; the prediction should be somewhere in the middle.

This is because, if either author did not write the document, it will merely tend to the value that it is more similar too. The critical value is the numerical difference between the two values of the top two (closest to one) predictions. Therefore, it would be logical that the different authors will present a statistically different prediction value such that authorship can be reliably defined, independent of the fingerprint that is randomly chosen for training.

This value was plotted using a boxplot (Figure 7) to determine if it was viable to use a threshold to offer a prediction of document author. How the threshold was chosen was that it was the value of the lower quartile. Therefore 75% of all differences between documents was in this range (Figure 8). The upper quartile value was not crucial as if the difference gets higher, then the likelihood of the

same author writing both papers becomes less and less likely. More to the point, the values of these differences distribute somewhat normally with minimal outliers.

To perform the prediction, the values of differences between the two predictions closest to one (the max) were taken. If this difference was greater than the threshold, then it can be deemed that the same author did not write the document and thus, the program outputs the document name and the fact that it was unlikely to be written by the same author. Should the value be close to the threshold, the user is instructed to rerun the test to make sure that the result is accurate? This is a very rare occurrence and only occurs when the document size is small. An indeterminate result occurs because when the document size is small, the chunk size is also small (or there may be one chunk). This results in an inconsistent prediction thus documents of small sizes (less than 500 words) should be avoided

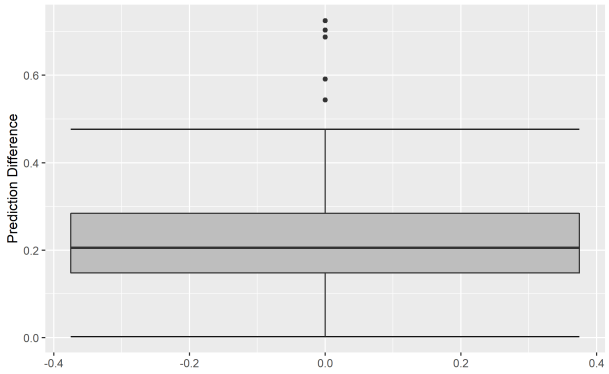

Figure 7: Boxplot for Threshold

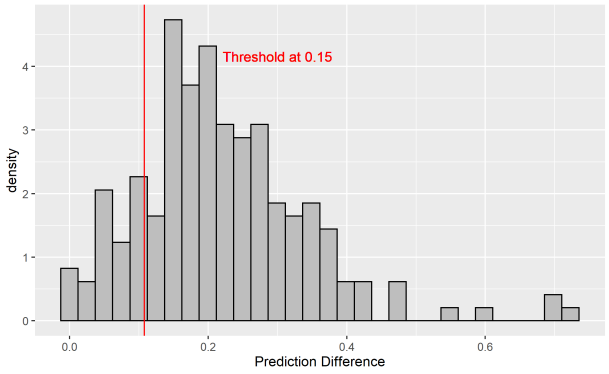

Figure 8: Threshold Distribution

5.2 Universality of Model

One of the primary purposes of this project was to develop a system of metrics that, when coupled with a neural network, can identify if the author of a piece of work is different. Because of this premise, the model has to perform uniformly across multiple sets of data whilst still forming a valid result. To prove the universality of the model, the weights of each network node should be relatively similar across all iterations. If this can be determined, then the model is 'similar' in the sense that each metric holds the same influence on the results for one set of data as it would for another. The tests were run on four different data sets 400 times each. This produced 1600 data points with there being 400 data-points for each of the four layers. To analyse the similarity, each of the four tests was loaded in taking their mean weights and plotting them with the other four iterations of the test. Each of these was then plotted on a box plot.

Analysing the results, the maximum differentials occur in the final layer, with a difference of 0.02 between which is mathematically insignificant. Over the other three layers, the changes were in the thousandths, which makes little to no difference. Over the rest of the layers, with 15 nodes, two had outliers, when there is 10, there are also two, and when there are five, there is one node that has an outlier. This can be seen in Figure 9, Figure 10, Figure 11 and Figure 12, where the weight of the node is mapped on the y-axis, and the nodes are the values along the x-axis.

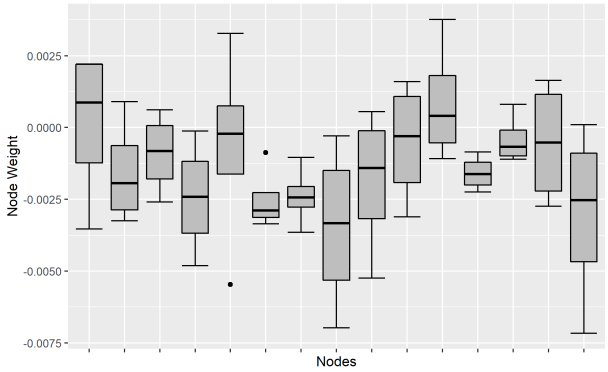

Figure 9: First Layer

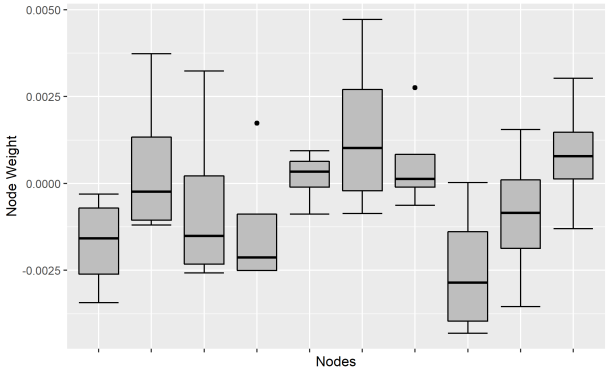

Figure 10: Second Layer

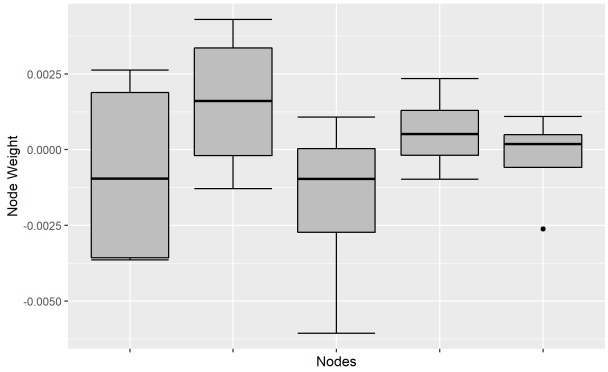

Figure 11: Third Layer

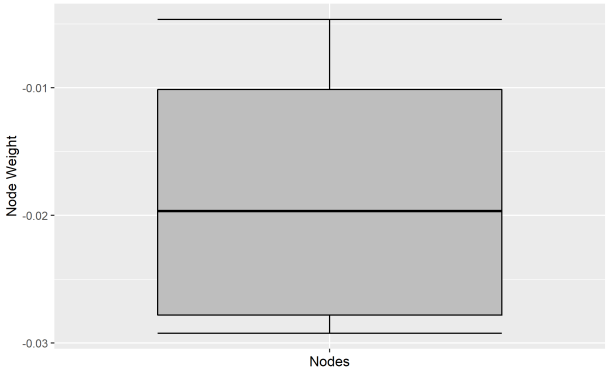

Figure 12: Final Layer

5.3 Metric Usage

As previously, discussed, the justification for the choice of metrics was centred on the lack of overlap of the metrics between each other such that there was little to no metrics referencing the same aspect of the text, providing the most amount of relevant data for the smallest number of metrics. Because of this, the ordering of the metric was done intelligently and is not inconsequential. The metrics are ordered in the order of most *mathematical* to least.

Therefore, the first four metrics are those developed from Zipf's and Yule's laws, followed by the quotes, and then the last four metrics, focussing around the structure of text and usages of certain kinds of words/punctuation are included. To test the relative impact of each metrics, the network was trained and tested on different data sets with an increasing number of metrics. On the first two test sets, the power/influence of Yules and Zipf's laws was shown such that the prediction of the document that was written by a different author was consistently high.

The more exciting occurrence is in the third test. For the initial metrics, the values of the differences were too close, such that the network was not able to predict that a different author did not write one document. However, when the later, more structural based metrics were including in the training, the prediction of the dissimilar document roughly doubled such that the author distinction was evident, and determined by the program.

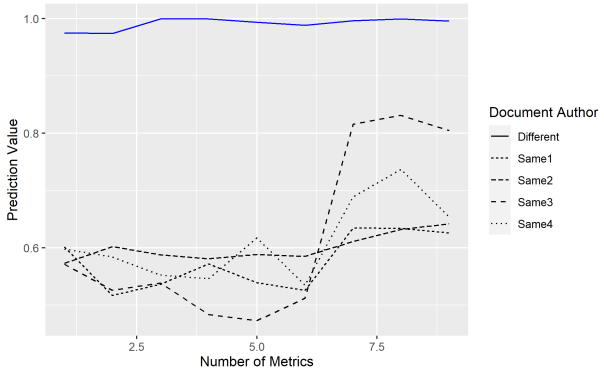

Figure 13: Metric Test 1

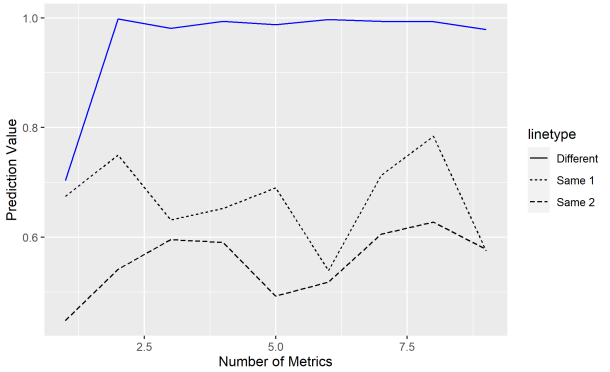

Figure 14: Metric Test 2

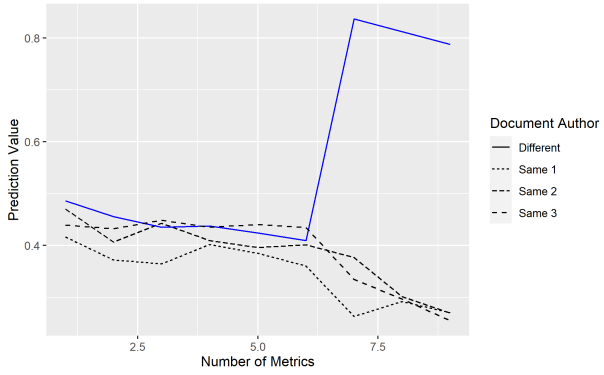

Figure 15: Metric Test 3

5.4 Prediction Consistency

The final consistency test of the program was performed on the fingerprints data sets, as described above. As these fingerprints are the style signatures of the authors who submitted documents for the development of this thesis, it is a fair assumption that the author's style will be relatively consistent over the data and that the order of that data is not essential. A new fingerprint was developed, which encapsulates the author of this papers writing style to use as the non-author in the program. These fingerprints were developed in a similar form to those previously, with there being fingerprints for the chunk sizes of 250, 500 and 1000 words.

To perform the test, two random fingerprints are chosen from the pool of similar chunk sizes (for example, John250 and Jane250). Then a predefined number of randomly assigned chunks are chosen from the data to be loaded into the network. The tests occurred following the same steps for 200 random iterations

1. Load in 2 random fingerprints, check authors similarity
   (a) choose x chunks randomly
2. Load into a NumPy array as two separate fake documents
3. Run prediction as defined in Figure 5 as if the documents created in step one are typical documents

4. Return the prediction
    (a) If a positive result (same same, different different), add 1 to total
5. If Incorrect, figure out what kind of error and add on to either of the values

The best prediction values were chosen from Table 2 to develop the three confusion matrices. It is important to note that the best prediction values follow the distribution as described earlier in the paper. According to the distribution, a document of 500-words will be broken into two 250 word chunks, and a 2500-word one will be broken into five 500 word chunks. The only one that does not follow the distribution is 1000 words. This is more likely to be affected by the data inputted rather than the distribution being incorrect, as to create 1000 word chunks, a document size of roughly 7000 is required, and no such documents were submitted.

Chunk Size	**2 Chunks**	**5 Chunks**	**10 Chunks**
250 Words	73%	66.5%	65.5%
500 Words	66%	81.5%	63%
1000 Words	76%	56.5%	50%

Table 2: Prediction Accuracy Based on Number of Chunks

250 Word Chunks		*Same*	*Different*	*Indeterminate*
Document Similarity	*Same*	31	5	3
	Different	34	116	11
F1 Score:	0.883			

500 Word Chunks		*Same*	*Different*	*Indeterminate*
Document Similarity	*Same*	25	3	3
	Different	30	128	11
F1 Score:	0.903			

1000 Word Chunks		*Same*	*Different*	*Indeterminate*
Document Similarity	*Same*	27	3	1
	Different	37	125	7
F1 Score:	0.884			

Table 3: Prediction Confusion Matrices

Looking at the confusion matrices, not only is the accuracy high, but the F1 score is also high, and the number of false positives is low. In a data space where the result of the program can be used to enforce disciplinary action against a person, the program should err on the side of caution when outputting positive results. In this case, if the program returns that a different author writes documents, the user can assume beyond a reasonable doubt (95% sure) that they are. In actuality, if such a result is returned, then there is a 1% chance that the program is incorrect. In regards to

the F1 score, having a high value for this is desirable is not only is it a useful metric for consistency but also it shows that the network is predicting the author, rather than using the data already submitted to assume the author.

5.5 Shortcomings

In short, there are two significant shortcomings of the program, these being the requirement of an intelligent user to ensure that the program performs as expected and that it cannot be used as the only proof of ghostwriting. The first shortcoming is the most major as when acting without an intelligent author; the program has the potential to return a false positive. This is that it says that a document is not written by the same author, even though it was. The primary reason that something like this has the potential of happening is that an author will most likely change their style depending on the context for which the piece of work is developed.

In many cases, this does not necessarily matter as the change is negligible and will not affect the performance of the program. An excellent example of this is an essay versus a reflection. Although one is generally written to discuss a problem and the other is to reflect on an event or set of events, how the writing occurs is not drastically different. Along with this, because the metrics are largely independent of language, what is being said in each piece does not matter. The primary reason for this is that more likely than not the form of both pieces of writing are long-form, fully formatted sentences. However, looking at the difference between a reflection and a laboratory report, the structure, and thus, the style is drastically different. Using documents written by me, it is demonstrable that it will return an incorrect prediction in such a case. An interesting effect that this can have is that, even though the documents are contextually different, if a document written by a different author is included in the test set, then that difference is large enough that it will be picked out. This will even occur if the styles (writing piece) of one of the two initial documents is similar to the odd document out.

```
Michael Bleakley Self Reflection Sprint 2.txt 0.2486099
Project 1 Report - 21959589.txt 0.07226178
All Results
The Documents are Written By Different Authors
```

Figure 16: Same author, Differing Context

```
Michael Bleakley Self Reflection Sprint 2.txt 0.0014962541
Project 1 Report - 21959589.txt 0.2776669
Proposal - John.txt 0.7848799
All Results
The Document Proposal - John.txt Is not written by the Same Author
```

Figure 17: Different Author Included

6 Conclusions

6.1 Future Work

Although this research produced a workable, reliably accurate product, some other elements could be improved or potentially set up for future research. Overall, four main areas can encapsulate

future work. These are the recognition of document types, improvement of the prediction values, different data processing, a graphical user interface (GUI) and the potential deployment of the program.

Starting simply, as the target audience for program are Academics, there is a fair assumption that those outside of STEM fields may not have to ability/ knowledge to run the program from a command line. Therefore, developing a GUI has vast benefits for adoption. Along with this, it can include the ability to modify how the files are read such that minimal data preparation has to be completed to test a wide range of data samples. The potential of deployment couples with this but can be applied in a slightly different way. Simply, it could be employed as the software Turnitin is applied at institutions currently wherein students have to parse their documents through it in the process of submission. This could take out the process of needing to parse documents only if they are suspicious, as well as allowing the development of a fingerprint throughout a student's studies, solving the propensity for there to be minimal available data.

The biggest issue currently facing the program is the potential that it can return false positives giving the applications of writing. The primary reason that this has not yet been solved is because the metrics were developed to be independent of language. Because of this, there is not the right way to determine what the context of a document is. For example, looking at the two documents as described in the tests above, the following metrics are returned.

Looking at the Samples of Data in Table 4, There are some more apparent differences between authors, with author two having lower values for the first four mathematic metrics, as well as having shorter sentences but on average, higher word length. The more interesting fact is that author matrics are not drastically different from the perspective of a human viewer between documents one and two, the program outputs them as different authors which can be seen in Figure 16. The most noticeable difference between the two (numerically) is in the areas of Yules K/I (being lower and higher in document one respectively), and the sentences being longer on average in document one. This also makes sense in the context of the document type, as a reflection is less formal, therefore will be more conversational in style, as well as being written with a more simple vocabulary. A relatively simple workaround to this issue could occur in deployment, where the unit coordinator defines the document type.

Yules K	Yules I	Zipfs 1	Zipfs 2	Quote %	Punc	Sentence	Wordlen	Contractions
Document 1: Reflection Author 1								
132.222	60.4026	0.93100	0.88577	0	0.02667	34.625	4.856667	0
245.111	35.9138	0.94095	0.87880	0	0.02	23.3333	4.603333	0
165.1111	50.3919	0.93040	0.91519	0	0.02	33.125	4.503333	0
Document 2: Report Author 1								
223.555	38.9273	0.93921	0.87724	0	0.03	21.4615	4.49	0.016667
190.666	44.6428	0.95386	0.91204	0	0.03333	23.7272	4.653333	0.01
244.222	36.0288	0.92551	0.90938	0	0.03	20.1428	4.53	0
Document 3: Proposal Author 2								
91.3333	80.2139	0.89794	0.84345	0	0.02333	18.7142	5.056667	0.013333
95.1111	77.8546	0.91125	0.88612	0	0.01333	20.1428	5.486667	0.003333
122.888	64.0113	0.91252	0.88717	0	0.01666	22.3333	5.41	0.006667

Table 4: Author Data From Context Failure

Finally, there is the improvement of the prediction values as a whole. Realistically, this is the most challenging area to improve for this type of program due to there not being enough data to improve the solution reliably. There are potential ways, for example, either developing more metrics or figuring out a better way to break down documents such that more data is produced. The main area in which the prediction could be improved is to produce less false negatives. Although false negatives are not a big issue in this context, it is the primary required improvement.

6.2 Conclusion

Throughout this study, it has been proven that it is possible to make a lightweight, simple and effective program for the identification of authors and ghostwriting. This was achieved through using stylometry and its fundamental theorems, choosing syntactic and Lexi-graphical factors such that the system is language independent. The key findings of the paper are three-fold. The first is that vocabulary based metrics create a good baseline for predictions of authorship. Secondly that the essential factor of discerning authorship is not the prediction itself, but rather the difference between prediction. This is a factor of the use case as the prediction is one too many instead of being one to one. Finally, the model itself is universal enough, even with the different document types and contexts of the documents. This shows the metrics are not only consistent but also weigh similarly with documents changing context. An interesting use case of this project is that it can be used to detect plagiarism as well, however services like Turnitin are more suited to this case. The future potential of this project is quite extensive, with the potential to not only improve and implement the program but also to extend its use case.

Ideally, this program would be implemented in academic institutions to catch ghostwriting and dispense disciplinary action to the authors in question. In terms of extension to other domains (such as journal articles), it can be extended and employed into other aspects of written work, with little to no change of metrics and prediction required. Given the results and accuracy, this program can be an influential tool in the combating of ghostwriting and fulfils the original aim of this paper.

7 Appendices

References

[1] "Turnitin." [Online]. Available: ttps://www.turnitin.com/regions/apa

[2] S. Singh and D. Remenyi, "Plagiarism and ghostwriting: The rise in academic misconduct," *South African Journal of Science*, vol. 112, no. 5-6, pp. 1–7, Jun. 2016. [Online]. Available: http://www.scielo.org.za/scielo.php?script=sci_abstract&pid=S0038-23532016000300012&lng=en&nrm=iso&tlng=en

[3] P. M. Newton, "How Common Is Commercial Contract Cheating in Higher Education and Is It Increasing? A Systematic Review," *Frontiers in Education*, vol. 3, 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/feduc.2018.00067/full

[4] R. Matthews and T. Merriam, "Neural Computation in Stylometry I: An Application to the Works of Shakespeare and Fletcher," 1993.

[5] F. J. Tweedie, S. Singh, and D. I. Holmes, "Neural network applications in stylometry: The Federalist Papers," *Computers and the Humanities*, vol. 30, no. 1, pp. 1–10, Jan. 1996. [Online]. Available: https://doi.org/10.1007/BF00054024

[6] T. Neal, K. Sundararajan, A. Fatima, Y. Yan, Y. Xiang, and D. Woodard, "Surveying Stylometry Techniques and Applications," Nov. 2017. [Online]. Available: https://doi.org/10.1145/3132039

[7] S. T. Piantadosi, "Zipf's word frequency law in natural language: A critical review and future directions," *Psychonomic bulletin & review*, vol. 21, no. 5, pp. 1112–1130, Oct. 2014. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4176592/

[8] A. Miranda-García and J. Calle-MartÍn, "Yule's Characteristic K Revisited," *Language Resources and Evaluation*, vol. 39, no. 4, pp. 287–294, Dec. 2005. [Online]. Available: https://doi.org/10.1007/s10579-005-8622-8

[9] T. V. N. Merriam, "Neural Computation in Stylometry II: An Application to the Works of Shakespeare and Marlowe," *Literary and Linguistic Computing*, vol. 9, no. 1, pp. 1–6, Jan. 1994. [Online]. Available: https://academic.oup.com/dsh/article-lookup/doi/10.1093/llc/9.1.1

[10] A. Caliskan-Islam, R. Harang, A. Liu, A. Narayanan, C. Voss, F. Yamaguchi, and R. Greenstadt, "De-anonymizing programmers via code stylometry," in *Proceedings of the 24th USENIX Conference on Security Symposium*, ser. SEC'15. Washington, D.C.: USENIX Association, Aug. 2015, pp. 255–270.

[11] "textract — textract 1.6.1 documentation." [Online]. Available: https://textract.readthedocs.io/en/stable/

[12] "Calculate lexical diversity — textstat_lexdiv." [Online]. Available: https://quanteda.io/reference/textstat_lexdiv.html

[13] "TensorFlow." [Online]. Available: https://www.tensorflow.org/