A Framework Supporting Human-AI Adversarial Authorship: The Analysis of User Frustration to Improve System Efficiency

by

Sadaira Packer

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 5, 2023

Keywords: adversarial authorship, natural language processing, feature extraction

Approved by

Gerry Dozier, Chair, Charles D. McCrary Eminent Chair Professor of Computer Science and Software Engineering
Cheryl Seals, Co-Chair, Charles W. Barkley Professor of Computer Science and Software Engineering
Jakita Thomas, Philpott-WestPoint Stevens Associate Professor of Computer Science and Software Engineering
Sanjeev Baskiyar, Professor of Computer Science and Software Engineering
Michael King, Associate Professor of Computer Engineering and Sciences

Abstract

Writing style can be traced back to a specific author with the use of authorship attribution techniques. These techniques use machine learning algorithms to classify the authors. This document discusses research focused on creating adversarial text to conceal an author's identity. A tool, AuthorCAAT, that performs adversarial authorship to assist in the anonymization of text using feature sets, language translations, and other transformation methods is utilized throughout this work. This tool is compared with other anonymization techniques while attempting to circumvent detection by high performing authorship attribution algorithms. We also explore combining anonymization methods to further improve the performance against the authorship attribution algorithms. This work is extended by using the anonymization methods in a partially observable environment where the authorship attribution algorithms are incorporated into the process of creating the adversarial text.

Another focus of this work is developing a more general framework for adversarial authorship. This is done by first examining the problem points of our tool then designing a user interface that mitigates the issues that contribute to a poor user experience. We then work to improve on the components of the framework by utilizing document clustering, deep learning and model interpretability.

Acknowledgments

I would like to thank my advisors Dr. Gerry Dozier and Dr. Cheryl Seals for their guidance, patience, and encouragement throughout this process. I am forever grateful for the opportunity they have provided me. I also would like to thank Drs. Marcellus Smith, Alexicia Richardson, and Brandon Brown for being incredibly helpful throughout this process. Finally, I would like to thank my parents for always believing in me and supporting my academic efforts.

Table of Contents

List of Figures

List of Abbreviations

AASs Authorship Attribution Systems

AIM-IT Automated Intelligent Masking Information Tool

AMT Author Masking Technique

AuthorCAAT Author Cyber Analysis Advisement Tool

EBEC Entropy-Based Evolutionary Clustering

IEC Interactive Evolutionary Computation

JohariMAA Johari Model for Adversarial Authorship

LDA Latent Dirichlet Allocation

LIWC Linguistic Inquiry and Word Count

Chapter 1

Introduction

Sharing information through the use of social media and blogs has become a part of everyday life. We give up a lot more about ourselves than we may realize. Every post that is written, liked, or upvoted divulges our opinions and interests. This information can be collected and used for targeted advertising and the spread of disinformation. We have seen something similar to this being done by Cambridge Analytica where they obtained data from 87 million Facebook user profiles and used it for targeted political advertising [1][2].

Regardless of if a user has their actual name attached to their internet profiles or not, they can be identified by their writing style. Stylometric analysis can be done on these online posts to reveal the author. There are some cases where the ability to reveal the identity of an author who wishes to remain anonymous can be detrimental such as in the realms of journalism and activism. Threats to privacy and anonymity can deter individuals from exposing information regarding the wrongdoings of an organization out of fear of retaliation.

Adversarial stylometry has been used as a method for authors to evade attribution. Adversarial stylometry [3] is eluding authorship attribution by altering writing style. Adversarial stylometry appears in the forms of imitation, translation, and obfuscation. Anonymization efforts have been made by[5] by way of a framework called Anonymouth that gives users suggestions on how to alter their documents for anonymization. They found that when using a complex feature set, users had a difficult time making the suggested changes to the document which led to them not being as successful at anonymizing their documents as they were with a simpler feature set.

Another method used for anonymization is adversarial authorship. Adversarial authorship was introduced by [6] in the form of AuthorCAAT-I (Author Cyber Analysis Advisement

Tool). AuthorCAAT-I utilized iterative paraphrasing and iterative language translation before adding in AuthorWebs in the second version of AuthorCAAT. An AuthorWeb shows the relationship between authors and their writing samples. AuthorCAAT-III [7] introduced interactive evolutionary hill-climbing and author target clusters.

AuthorCAAT-III was able to evade attribution of the Writeprints (Limited) feature set 50% of the time. While AuthorCAAT has shown to be an effective tool for anonymization, there are issues pertaining to its usability. Using AuthorCAAT can be a very tedious process. There are many steps and long waiting periods. The purpose of this research is to design, develop, and evaluate a human-AI collaborative system for adversarial authorship, JohariMAA (Johari Model for Adversarial Authorship). JohariMAA aims to maintain the effectiveness of AuthorCAAT while offering a more pleasant user experience. This document presents research conducted on methods to evolve adversarial text to conceal the identity of an author. This document also presents a generic framework for adversarial authorship that not only makes it easier to evaluate adversarial authorship methods, but also allows for more flexibility in the composition of these methods.

The remainder of this document is arranged as follows. Chapter 2 includes a review of literature on genetic algorithms, interactive evolutionary computation, genetic & evolutionary feature selection, authorship attribution, adversarial stylometry, adversarial authorship, and interaction design. Chapter 3 discusses the datasets used in our work. Chapter 4 presents a study of AuthorCAAT-V and the potential for a human-AI hybrid for adversarial authorship. Chapter 5 builds on the work from the previous chapter by performing adversarial authorship in a partially observable environment. Chapter 6 discusses our reasoning for moving away from AuthorCAAT and towards JohariMAA. Chapter 7 discusses the design and implementation of JohariMAA. Chapter 8 explores utilizing the interpretability of deep learning models for adversarial text generation and document clustering as a potential way to further improve the obfuscation of a document. Chapter 9 presents future directions that this research could take.

Chapter 2

Literature Review

2.1 Genetic Algorithms

Genetic algorithms [19] are search algorithms that are used to solve optimization problems. In a genetic algorithm, a population of individual chromosomes (or candidate solutions) is continuously evolved to find the best individual [45]. The main components of a genetic algorithm are selection, crossover, mutation and the fitness function. Each individual is usually represented by an array of either binary or floating point numbers and given a fitness value based on some fitness function. Individuals, referred to as parents, are selected from the population to create offspring. Parents are often selected based on how fit of an individual they are. They can also be randomly selected. Some selection methods include tournament selection and fitness proportionate selection [45]. For tournament selection a group of individuals is randomly selected from the population and the best individual from the group is chosen. For fitness proportionate selection the probability of an individual becoming a parent is proportional to its fitness. Once the parents are selected, crossover is performed to create the offspring. Some crossover operators include uniform crossover and multi point crossover [47]. Uniform crossover is done by inheriting each gene from a parent randomly. Multi point crossover is done by breaking the chromosome into multiple segments and alternating which segment is inherited from which parent. Mutation maintains the diversity of the population by randomly making changes to the genes of an offspring. Some mutation operators are swap mutation and scramble mutation [47]. Swap mutation is done by randomly selecting to genes and then swapping their alleles. Scramble mutation is done by selecting a subset or the entire chromosome and scrambling the

positions of the alleles. After crossover and mutation is performed on the offspring, the offspring replaces the least fit individual. The process is repeated until a termination condition is met. The termination condition could simply be a finite number of fitness function evaluations or the population has not improved after a certain number of fitness function evaluations.

2.2 Interactive Evolutionary Computation

Interactive Evolutionary Computation (IEC) [12] involves a human's subjective evaluation in the optimization of a system. IEC is useful for problems that may not be easily evaluated with a fitness function such as design and music generation problems. In [46] they used IEC in a genetic algorithm that generates jazz solos called the GenJam. GenJam has a population of 48 phrases population and 64 measures. The user evaluates each measure of the generated solo by rating it as good or bad. The user rating increments or decrements the fitness score of the solo. The main issue that arises with the addition of a human element into the evolutionary process is user fatigue. A user is only able to perform a finite number of evaluations. Some problems may not reach an optimal solution before the user is fatigued. The GenJam is configured to receive user feedback per an entire measure rather than per note, which helps to alleviate user fatigue.

2.3 Genetic Evolutionary Feature Selection

Genetic Evolutionary Feature Selection (GEFeS) is a feature selection method that uses a steady-state genetic algorithm [19][20][21]. It is used to evolve a population of feature masks. These feature masks are subsets of the features of a specific dataset. GEFeS is used to find the subset of features that are the most important. The feature masks are represented by chromosomes consisting of binary values that indicate which features are turned on (1's) and off (0's). The fitness for each feature mask is determined by accuracy and the percentage of features that are turned on. GEFeS was used in [48][49][50] with feature sets that covered sentiment analysis and topic modeling to evolve more useful feature masks. A multilayer perceptron (MLP), a radial basis function support vector machine (RBFSVM), and a linear support vector machine

(LSVM) were the classifiers used for testing the feature masks. In [48], the use of GEFeS with the sentiment analysis feature set improved the accuracy of all of the classifiers by 31-39% while using only 21-25% of the features. The greatest improvement occurred with the LSVM. In [49], they tested GEFeS with two feature sets, stic Inquiry and Word Count (LIWC) and sentiment analysis (SA). The feature sets were evaluated separately and combined as one larger feature set. The use of GEFeS improved the accuracy for all three feature sets. It increased the accuracy for SA and LIWC by 38% and 17% respectively. For the combined feature set, the accuracy was increased by 30%. There was a 1% difference between the accuracy of LIWC and the combination of LIWC and SA. They also tested how effective these feature masks are on a set of adversarial texts that were generated using character unigrams. They saw that they could reduce the adversarial effectiveness when performing authorship attribution with a feature set that differs from the feature set used to create the adversarial text. In [50], GEFeS was tested with LIWC, sentiment analysis (SA), and topic modelling (TM). They tested the four possible combinations of the three feature sets. When combining all three feature sets and classifying with the RBFSVM, they were able to achieve 100% attribution accuracy with as few as 17.8-56.4% of features being used. These results show that using GEFeS improves performance in authorship attribution systems.

2.4 Authorship Attribution

Authorship attribution refers to the act of identifying the author of an unknown document[3]. Authorship attribution could be used to aid in forensic investigations and plagiarism detection. In [8], The Brennan-Greenstadt dataset (all English text) was used along with samples from two French and four Dutch authors. The feature set used was built from a combination of features from the Basic 9 and Writeprints. They called this the "Translation Feature Set." JStylo was used for authorship attribution. Bing and Google translators were used to translate the samples from the Brennan-Greenstadt dataset. It was done in three different sequences: from the original text to German and then back to English; from the original text to Japanese and then back to English; from the original text to Japanese, then German, and then back to English. Google Translate, Language Weaver, Systran were used to translate the samples from

the French Dutch authors to English. The results showed that the more translations a text went through, the easier it was to attribute the correct translator to the text. Japanese translations were also attributed more accurately than German translations when attributing translators. Most of the author attributions for the Google and Bing translations had accuracies above 90%. The texts that went through Japanese and German before going back to English had accuracies below 90%. The French samples were attributed to the correct author 100% of the time for all translators. The Dutch samples performed the worst. This is possibly because all the authors' samples are on the same topics. The Translation Feature Set had higher accuracy than all other feature sets in Jstylo for both author and translator attribution.

Herz and Bellaachia [31] take 37 of Obama's speeches and attributes them to four speechwriters. The speeches are preprocessed in four different ways: stemmed with augmented term weighting, un-stemmed with augmented term weighting, stemmed with normalized term weighting, and un-stemmed with normalized term weighting. Function words is the only feature that is used. Analysis of variance was done on the frequencies of the function words of each set to select distinguishable feature vectors. Each set was classified using Naïve Bayes, k-nearest neighbors on the projections of Principal Component Analysis, Linear Discriminant Analysis, and feed-forward neural networks. Leave one out cross validation was used to test the accuracy of each classifier. Linear discriminant analysis performed the worst, being around the accuracy of random chance. The most successful classifier was the k-nearest neighbor with principal component analysis. It had accuracies around 68-78% for all four sets. There was some trouble identifying the speeches of one writer. He had the fewest speeches (five) and they seemed to be more of a collaborative effort than the speeches of the other three speechwriters. Removing his speeches from the set significantly improved the accuracy of all the classifiers. The results for the k nearest neighbors classifier was still the highest at 81-91%. The classifiers performed best on the un-stemmed sets.

In[3] they assessed the performance of several authorship attribution algorithms. Two of the best performing algorithms were referred to as teahan03 and koppel11. Teahan03 [27] uses Prediction by Partial Matching text compression for text categorization. An author is attributed by obtaining the cross-entropy for the categories and then selecting the author with the lowest

cross-entropy. Koppel11 [28] works by iteratively selecting random subsets of a feature set and using cosine similarity to find the top match. The text is attributed to the author that is selected as the top match the most.

2.5 Adversarial Stylometry

[3] defined five subtasks of stylometry (authorship attribution, authorship verification, authorship profiling, stylochronometry, and adversarial stylometry) and gives an overview of feature categories and common classification methods. Authorship verification deals with deciding if two documents were written by the same author. This problem is approached with either the many-candidates method or with one-class classification. Authorship profiling deals with analyzing a document to determine the demographics of its author (i.e. gender) without actually identifying the author. Stylochronometry is the study and detection of changes in authorial style over time. Stylochronometry isn't as reliable as authorship attribution and verification because it hasn't been studied as much. Adversarial stylometry is the altering of one's writing style in order to evade authorship attribution. The three forms of adversarial stylometry are imitation, obfuscation, and translation. An overview of the feature categories was given along with two feature extraction techniques was given. Lexical, syntactic, semantic, structural, and domain specific features are described and examples of each was given. Lexical features are the most basic category and can be character-based or word-based. Next, various methods of classification were discussed. Machine learning algorithms are a common approach to classification. Some algorithms used are decision trees, support vector machines, and logistic regression. Support vector machines are popular because of their ability to handle large amounts of data. Nearest-neighbor is a method where the similarity between the unknown text and known texts is measured. Distance can be measured in several ways including Delta, Chi-Square, and Kullback-Leibler Divergence. Probabilistic models are also used for classification. Performance of the subtasks can be analyzed using a few different metrics. Some of the most common metrics include accuracy, recall, and precision. The CASIS dataset was used for testing. The dataset has 4,000 blog samples from 1,000 authors. Each sample is a short blog post. The dataset was then broken down into subsets with different amounts of authors, samples and

sentences (i.e. CASIS-5 has 378 authors and 6,691 samples with 5 sentences per post). Bag-of-words features were extracted. They tested authorship verification on the CASIS dataset and its subsets using five-fold cross validation and observed that there was an increase in the equal error rate as there was a decrease in the number of sentences per post. Fourteen algorithms for authorship attribution were tested against the datasets. Lower-level features (i.e. byte and character-level) performed better than higher-level features (i.e. syntactical and word-level). The performance of all algorithms decreased as the text lengths decreased from 20 sentences per post to sentences per post across the sets. They then move on to address challenges faced within the research of stylometry. There is a need for a large dataset consisting of documents of varying types and topics. Authorship attribution becomes challenging when working with posts from blogs and social networks because of anonymity and the use of slang or improper grammar. There is an issue with not being able to scale a solution from a small, balanced dataset to a larger, less stable dataset. Machine translation can cause problems because of the possible loss of meaning from the original text.

In [32] they used data from underground forums to test an algorithm that detects the identity of one user across multiple accounts. Some of the features used includes frequency of language-specific parts-of-speech and function words, percentage of leetspeak per document. The feature set excludes word n-grams but includes all other n-grams. Principal Component Analysis was used to weight and select features with high variance. The algorithm that they used to identify multiple identities of an author calculated the probability that one author's document would be attributed to a second author and that the document of the second author would be attributed to the first author. They then combine the two probabilities and if they are higher than a certain threshold the two authors are the same. They tested the algorithm on the two German forums Carders L33tCrew. They were able to use email addresses to find common members between the forums. 28 pairs of authors were tested and a result of .85 precision with .82 recall was produced with 4 false positives. The algorithm was also tested on the private messages of users from just Carders where they manually analyzed the top 21 pairs that had the highest probability score to validate their results. 13 of the 21 were decided to be either true or probably true.

Adversarial stylometry is the act of altering one's writing style to avoid authorship attribution. In [9], the strength of select stylometry methods against obfuscation and imitation are tested. Each of the 15 participants submitted pre-existing sample writings of 5,000 words for their dataset. They were asked to create an obfuscation sample on a specific topic and a sample imitating another author (all imitating the same author) on a specific topic in the third person. The training text was the sample taken from the author that was being imitated. Three authorship attribution methods were used. For the statistical method, the features used were word lengths, letter usage, and punctuation. The sample text was compared with each author's dataset the chi-square of the two was summed. The lowest chi-square value of the pairs was selected as the author. For the neural networks method, nine features were extracted using Textalyser (lexical density, Gunning-Fog readability, sentence count, etc.). For the synonym-based classifier method, a match value was calculated by summing the number of synonyms a word has multiplied by the shared frequency of the word between sample text training text of a known author for all shared words between the two texts. Authorship was attributed to the author with the highest match value. All methods were tested with groups of 2, 3, 4, and 5 authors. For obfuscation, the methods performed at random chance for most of the tests. There were only 3 instances where the accuracy was slightly better than random chance. All performed significantly worse against imitation than they did for obfuscation. Authorship was attributed to the author being imitated most of the time. All three methods performed far below random chance in every instance.

In [10] the three techniques of adversarial stylometry (imitation, obfuscation, and translation) are tested against stylometry techniques. Human subjects were used to test imitation and obfuscation while a machine was used for translation. Manual adversarial stylometry was shown to have performed better than machine adversarial stylometry. The Brennan-Greenstadt Adversarial Stylometry Corpus consists of a large collection of texts from 12 unique authors and adversarial texts written by the same authors. Each author submitted at least 5,000 words from professional or academic preexisting texts, an obfuscation sample and a sample imitating the writing style of Cormac McCarthy. All authors wrote about two specific topics for both the obfuscation and imitation samples. The Extended Brennan-Greenstadt Corpus was created

with 45 authors, all submitting at least 6,500 words of scholarly preexisting texts. Three stylometry techniques were used to test the adversarial techniques. First was a neural network with the Basic-9 feature set. This was considered a basic technique. Second, they tested a synonym-based method where the only feature used was the vocabulary frequency. Then they tested a support vector machine with Writeprints-Static. Writeprints-Static is a hybrid of the Writeprints baseline and extended feature sets created to reduce computation costs. It consists of 557 features. The support vector machine used was a sequential minimization optimization. 1,000 unique test sets of authors from the Extended Brennan-Greenstadt Corpus was used to train the classifiers. The test sets had varying amounts of authors in them (i.e. 5, 10, 15, etc.). For obfuscation, the only method that was able to attribute authorship correctly above that of random chance was the support vector machine. For imitation, the attribution accuracy was below random chance for all methods. Two translation methods were tested: one-step and two-step. For one-step, they translated English to German and then back to English and from English to Japanese and back to English. For two-step, they translated from English to German then to Japanese and back to English. Google Translate and Bing Translator were the tools used to do this. The one-step translations for both German and Japanese were not effective at hiding the author's style. The two-step translation was generally no more effective than two-step. Synonym-based classification performed the best on all 3 experiments. Machine translation is not a very effective adversarial technique.

Anonymouth is a program that helps one to make their writings anonymous by offering suggestions to alter the document [5]. The user submits the document they want to anonymize, other samples of their writing, and samples from three other authors. Features are then extracted from all of these sample groups and then groups the features by these three groups plus a fourth group consisting of the combination of features extracted from the user and other authors. The top features are calculated from the fourth group. The features from the other authors' samples are clustered and are used to create target values for the user's final document configuration. A user is then presented with a list of suggestions for changes that they can make to reach these target values. When changes are made the document is reclassified to see how close it is to its target values. This process continues, making changes and reclassifying, until the

document's values reach the target values. They had ten participants test the program and eight of them were able to make their document anonymous. Of the two whose documents weren't anonymous, one didn't make any changes and the other's original document wasn't attributed to him as an author.

JStylo was used to check the anonymity of the participants' final documents. The Brennan-Greenstadt dataset was used as a background dataset. They tested the documents with six authors and then all 13 from the dataset. The change of the background dataset changed the anonymization of the documents. The participants were only able to make the changes suggested by the Basic 9 feature set, as the Writeprints (Limited) feature set's suggestions were too complex. Larger feature sets are more difficult to use because the changes they suggest aren't easy for a user to apply. With using the Basic 9 feature set to make changes to their document and for attribution, 80% of participants were able to evade attribution. When attempting to attribute the same documents using Writeprints (Limited), all of the participants were correctly attributed.

The changes made to Anonymouth are explained in [11]. They added two-way translations that are applied to each sentence in 15 different languages. The translated sentences are sorted by which one is the most likely to provide anonymity based on feature frequencies, information gain, and target values. The user can then select a sentence and make any edits needed. An "Anonymity Bar" was added for the user to gauge their progress. The target values that are possibly the best are tested against the classifier before being chosen as the target values.

2.6 Adversarial Authorship

AuthorCAAT (Author Cyber Analysis Advisement Tool) is an instance of a (1 + 1) IEC [**?**][13][14][15][16] that allows a human to interactively evolve adversarial text. Adversarial text helps to preserve privacy and anonymity. The goal of AuthorCAAT is not only to conceal the identity of the author through the creation of adversarial text, but to also preserve the context of the original text. The first version of AuthorCAAT utilized iterative language translation (ILT) and iterative paraphrasing as mutation methods [6][17]. The languages used for translations were English, Spanish, and Chinese.

AuthorCAAT-II included AuthorWebs [18]. AuthorWebs are developed through Entropy-Based Evolutionary Clustering (EBEC). An AuthorWeb is made up of authors, their writing samples, and vectors of the features extracted from the writing samples. They are used to visualize nodes that have arcs directed to and from them. The arcs represent writing samples where each node represents an author. Each arc is directed towards the node that it is closest to with respect to the extracted feature vector. The arcs directed to a node are referred to as an author cluster. In AuthorCAAT-II a user can select which author cluster that they would like to right towards or away from.

AuthorCAAT-III introduced ILT hill-climbing to the IEC process as an additional mutation method [52]. The languages used for translation are Spanish, Chinese, Japanese, Korean, Russian, Arabic, French, and German. In ILT hill-climbing, ILT is done with all eight languages for a number of iterations. After these iterations, the resulting translation that is closest to the selected author target is presented to the user. The user can select this translation to replace the current text or modify the current text and resubmit it to the ILT hill-climbing process.

2.7 Interaction Design

Ceaparu et al. [24] conducted a study where they had 111 students perform their regular tasks on a computer for a couple of hours and log their frustrating experiences in a time diary. The frustrating experiences were broken down into five categories: internet, application, operating system, hardware, and other. Some of the most common frustrating experiences involved error messages, long download time, connection issues, and missing or hard to find features. One-third to one half of the time spent on the computer was wasted due to the frustrating experiences. This time was calculated by adding the time spent solving the problem and the time spent recovering the work that was lost because of the problem. They also found that the majority of frustrating experiences encountered by the participants happen on a regular basis.

A similar study was done in [25] with 50 participants in the workplace. The most common frustrating experiences were system crashes caused by specific applications and the operating system. The time wasted solving and recovering from the problem was around 40%. Some of the frustrating experiences caused by applications include uncontrollable pop-up windows,

unpredictable behavior of the application, unclear error messages, and missing or hard to find features. While some of these issues may be harder to rectify, the issues relating to the user interface can be remedied through better usability testing.

When designing an application it is important to take into account its usability. Usability refers to the ease of use of the application. It is defined by [26] as having five components: Learnability, Efficiency, Memorability, Errors, and Satisfaction. These five components are important to keep in mind when designing an application because they contribute to the user having an overall good experience, thus increasing the chance that they will continue use of the application. It is imperative that the usability of the application is evaluated by performing user testing with a group of users that represent the target audience of the application. When adding the element of artificial intelligence to a program, it is important to take into account how it affects the user experience. In [29] they examined the effect of having artificial intelligence involved at varying amounts in a program called DuetDraw where a user collaborates on drawings with artificial intelligence. There are five functions that can be carried out by the artificial intelligence of DuetDraw: the AI can automatically complete a user's drawing; draw an object that is similar to an object that was drawn by the user; draw an object that makes sense with what the user has already drawn; locate an empty space on the canvas; and add color to the sketches based on the colors chosen by the user. Four setups were tested where two different communication styles (detailed or basic instructions) were paired up with two different initiative styles (lead or assist). The detailed instructions walk the user through each step whereas the basic instructions only show an icon (e.g. a pen). For the lead initiative style the user does the majority of the drawing while the AI handles the smaller tasks. For the assist initiative style the user takes on the smaller tasks while the AI does the majority of the drawing. 30 participants were asked to create five drawings each, one drawing for each of the four combinations of initiative and communication styles as well as a fifth setup where there was no AI. User feedback was gathered through the think aloud method, post survey, and semi-structured interviews. The participants preferred detailed instructions over the basic instruction when paired with both initiative styles. They also found that most users wanted to take the lead. The users felt that the more repetitive tasks should be left to the AI. The setup that did not include AI had

the best scores for predictable, comprehensible, and controllable. The detailed instruction style performed better than the basic instruction style in these areas.

In [30] the relationship between controllability and accuracy is explored by varying the accuracy of the AI and the controllability of the crane movement in a crane simulator. The objective of the crane simulator is to use the crane to stack boxes. AI is used to locate the target segment where the box should be placed, called the Box Detection System. How accurately the box detection system locates the target is dependent on the accuracy factor. They tested five accuracy values ranging from 0.1 to 0.9. The controllability of the crane depends on how much the crane moves with each manual command, called the step. They tested five step values ranging from 0.8 to 32. Each step value was paired with each accuracy value to make 25 conditions. The participants had five practice trials and 50 measured trials. Afterwards they were given a survey to fill out. The participants had three possible rectification approaches: manual controls only, automation only, or both manual controls and automation. Participants preferred manual rectification for 21 out of the 25 conditions. They found that users preferred to move the crane manually even when the controllability was low and less time efficient than the autonomous option.

Chapter 3

Datasets

The CASIS-1000 (Center for Advanced Studies in Identity Science) dataset [3] is made up of 1,000 authors, each having four blog samples for a total of 4,000 blog samples. The blog samples are written in English by non-native English speakers. There is an average of 13 sentences per author. The CASIS-25 dataset is a subset of the CASIS-1000 dataset. It consists of the first 25 authors of the CASIS-1000 dataset, therefore it is 100 samples. The Bot Detection dataset comes from the PAN author profiling task for 2019 [51]. The dataset is made up of tweets from bots and humans. It consists of an English set and a Spanish set, we only focus on the English set. There are 3,380 bots and an equal number of humans, so a total of 6,760 authors. The humans are evenly split at 1,690 each for males and females. There are 100 tweets per author. Some of the bots they used were pulled from other datasets and others they found on their own by searching on Twitter with phrases like "I'm a bot." They were able to break the bots down into four categories: template, feed, quote, and advanced. The bots in the template category respond to tweets that are of a specific topic while the bots in the feed category retweets or shares news on a specific topic. The bots in the quote category tweets quotes from famous people or works (e.g. music, books, etc.). The tweets from the bots in the advanced category are generated in a more complex manner (e.g. Markov chains). The humans in the dataset come from the datasets used in the author profiling task in 2017 and 2018. Bots-25 is a subset of this dataset that is made up of 25 authors. Nine of the authors are bots and the other 16 are human (8 male, 8 female). There are five samples per author, giving a total of 125 sample. There are 20 tweets per sample.

The Enron dataset consists of emails from employees of Enron **??**. The full dataset has 158 authors and 200,399 samples. On average, there are 757 samples per author. The Extended Brennan-Greenstadt Adversarial Stylometry Corpus includes 45 authors and a minimum of 6,500 words per author [10]. Twenty-five authors were taken from each dataset and used to create models for each feature set in JohariMAA.

Chapter 4

Towards a Human-AI Hybrid for Adversarial Authorship

4.1 Introduction

According to Shou [55], one of the imminent threats to the anonymity of our cyber identities is that it is difficult for current AI systems to 'forget' our digital exhaust – the data that has been collected from us. Even if one could develop methods that would allow AI systems to forget, there may still be some who would refuse to eradicate the digital exhaust of others. This is an ever-growing problem with respect to Internet users and their privacy. AI systems for Authorship Attribution [3][4] are now becoming ever more sophisticated and efficient in identifying individuals based on their writing style. One method that can be used to preserve the privacy of internet user, with respect to their writing style, is known as of Adversarial Authorship. Some forms of Adversarial Authorship include Adversarial Stylometry and Author Obfuscation/Masking. For each of these methods, the objective is to mask the true identity of an author. In this paper, we compare a number of methods for Adversarial Authorship in an effort to determine their effectiveness in developing adversarial texts in an effort to conceal the identity of an author against a number of well-known authorship attribution systems (AASs). In this chapter, we also present a human-AI hybrid for Adversarial Authorship that outperforms a number of state-of-the-art author masking techniques (AMTs). Hybridizing human intelligence with artificial intelligence can potential to provide better perform than just relying on human intelligence or artificial intelligence alone [56]. Such collaborative systems allow for the strengths of both humans and AI to be utilized in a complementary way. AI can provide a

quick analysis of large amounts of data, while a human's intuition is invaluable when it comes to decision making.

4.2 AuthorCAAT-V

The process for creating adversarial text with AuthorCAAT-V begins with the user entering their parent text that they wish to anonymize. The user picks an author target from a list of 25 that they would like to move towards or away from and they pick a feature set that they would like to focus on. The feature sets available are character unigrams, sentiment analysis, Linguistic Inquiry and Word Count (LIWC), topic model, bag of words, and stylometry. AuthorCAAT-V uses multiple feature sets because changing a document based on one feature set may not provide anonymity when an authorship attribution system classifies using a different feature set than the one that was used to make changes to the document [33].

LIWC is a program that extracts information from text that provides an understanding of the psychological state of an author [22]. The words from the text are taken and compared to LIWC's dictionary of 6400 words. These 6400 words belong to different hierarchical categories. Each word from the sample text that is found in the dictionary counts towards the category associated with the word from the dictionary. For the topic model feature set we use the MALLET program to extract a set number of topics that are specific to the group of documents being analyzed [23]. Topics are groupings of words that have been determined to be related because they commonly appear together in a set of documents. MALLET gathers the topics by analyzing the document using the topic modeling algorithm latent Dirichlet allocation (LDA) and Gibbs sampling. The program returns the words with their associated topics and the distribution of the topics in each document. OpinionFinder is the program used for the sentiment analysis feature set [34][35][36][37]. It allows users to observe the polarity and subjectivity of the text. OpinionFinder examines the words of the documents and compares them to its dictionary. The words in the dictionary have an assigned polarity and subjectivity.

AuthorCAAT-V uses GEFeS (Genetic Evolutionary Feature Selection) for off-line feature selection. GEFeS allows for the evaluation of various subsets of a feature set in order to determine the best features for a specific dataset [19][20][21]. AuthorCAAT-V also utilizes a linear

support vector machine because it has shown to perform well in [33] and [38] when used with multiple feature sets.

Once an author target or feature set has been selected, a visual representation of the fitness scores for all 25 authors is displayed. The numerical value for the selected author target is displayed under the parent text. The higher the score is the closer the text is to the author target. These values change depending on the selected feature set. The user then mutates their text using two-way translations or iterative language translation hill-climbing (steepest ascent or descent). The languages used for translating are Spanish, Chinese, Arabic, Japanese, Russian, Korean, German, and French. After running one of these methods a mutated text is produced along with a fitness score for the mutation. Like the score for the parent text, this score is relative to the selected author target. The user can then decide if the mutated text is good enough to become the parent text or if they would like to mutate the parent text again. Once a satisfactory mutated text has been created, the user can copy this text to the main text box where they can make modifications and restart the process. Figure 4.1 depicts the process of using AuthorCAAT-V.

4.3 Experiment

AuthorCAAT-V was used to create 25 adversarial texts. Three other author masking techniques (AMTs), Castro [40], Mihaylova [41], and Rahgouy [42], were also used individually and together in a system called AIM-IT to create sets of adversarial texts. The technique referred to as Castro uses a simple method for masking the original text in an attempt to shorten it. This is primarily done through contraction replacement, synonym substitution, and sentence simplification. The technique referred to as Mihaylova targets many different style indicators typically used in author identification. The authors present three main categories: text transformations, noise, and general transformations. Text transformation consists of methods such as adding or removing punctuation, splitting or merging sentences, etc. Noise consists of replacing American English words with their British English counterparts and vice versa as well as adding or removing function words at beginnings of sentences. The general transformation consists

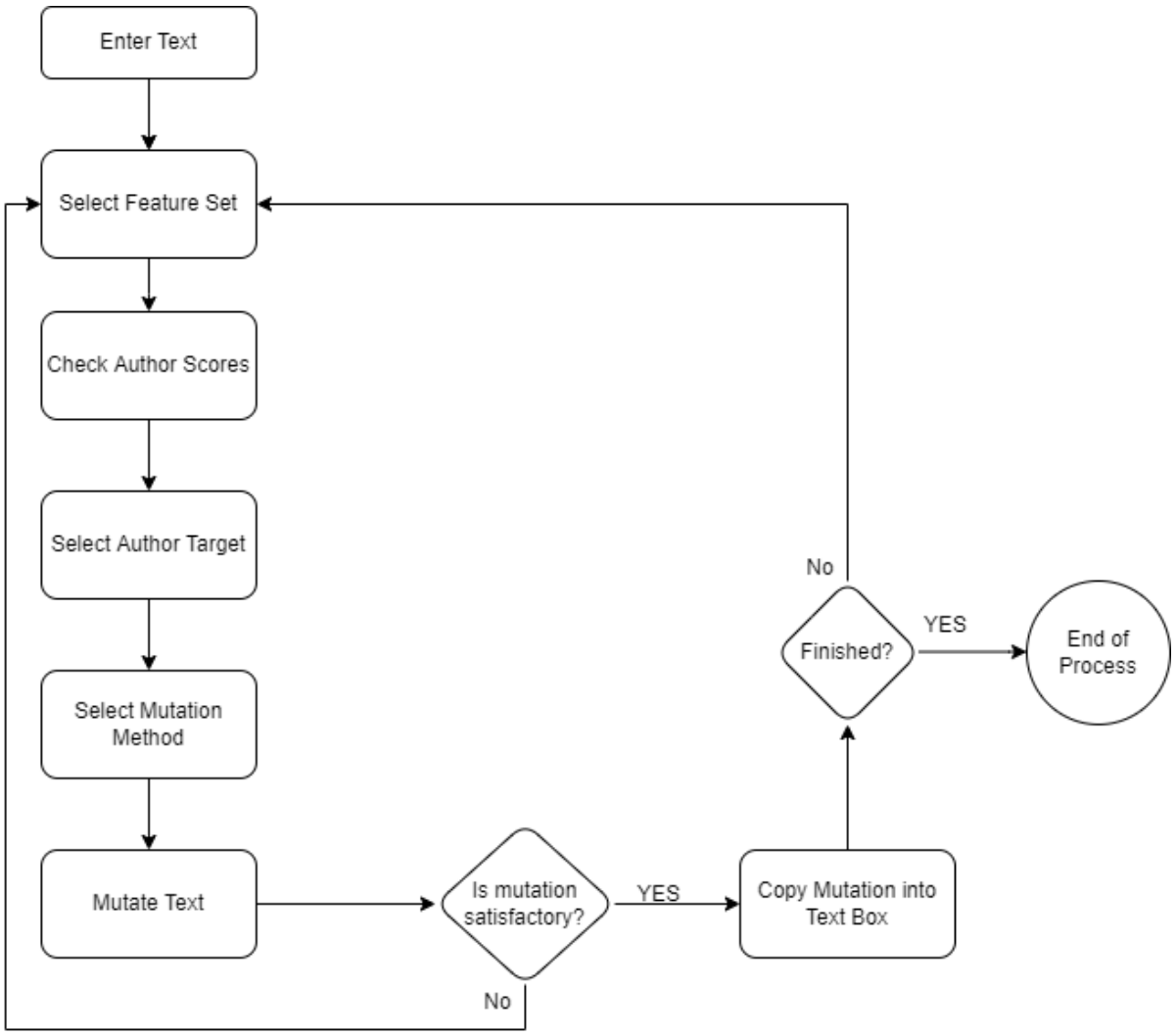

Figure 4.1: Flowchart of AuthorCAAT-V Process

	Keselj2003	Teahan2003	Koppel2011	CNN
Original	0.60	0.92	0.76	0.84

Table 4.1: Comparison of the AASs on the CASIS-25 Dataset

of techniques like contraction replacement, replacing possessive phrases using regular expressions, etc. All of these methods are used to push the features of a given text toward the average of a specified training corpus. The technique referred to as Rahgouy is a method similar to Mihaylova. Rahgouy used word replacement, phrase replacement, contraction replacement, and either sentence splitting or merging, in order to transform their text samples.

In our experiment, we compare the masking performances of the five AMTs presented earlier on their ability to mask the first 25 instances of the CASIS-1000 dataset. The masking performance of a method is simply the change in accuracy, [54], given by the original text when classified by the following authorship attribution systems:

1. Keselj-2003,
2. Teahan-2003,
3. Koppel-2011,
4. CNN [44].

Table 4.1 shows the baseline performance of the four AASs on the CASIS-25 dataset.

4.4 Results

The results presented were generated by applying the adversarial authorship techniques to the fourth writing instances of the first 25 authors of the CASIS-1000 dataset. The adversarial authorship techniques were 'blind' in that they had no prior interaction with author attribution or verification systems. After the AMTs 'evaded' classification of their internal author identification systems, the resulting adversarial texts were submitted to the set of author AASs systems.

Table 4.2 provides the results of our experiment. The first column contains the names associated with the authorship attribution systems and the change in accuracy, Δ_{Acc} using only

the writing samples of the first 25 authors of the CASIS-1000 dataset (referred to as CASIS-25). The entries within the first column of Table 4.2 corresponding to the authorship attribution systems are: Keselj2003, Teahan2003, and Koppel2011. The next five columns of Table 4.2 correspond to the five following AMTs presented earlier: Castro, Mihaylova, Rahgouy, AIM-IT, and AuthorCAAT-V. Castro, Mihaylova, and Rahgouy were selected because of the top ten AMTs presented in [54], they were ranked 1st, 2nd, and 3rd respectively. The last column corresponds to the AuthorCAAT-V + AIM-IT hybrid. The hybrid adversarial text were formed by taking the adversarial texts developed by AuthorCAAT-V and running them through AIM-IT. In Table 4.2, with respect to the first five AMTs, one can see that AuthorCAAT-V had the best performance against three AASs while Castro, Mihaylova, Rahgouy, and AIM-IT had the best performance against 0, 0, 1, and 2 AASs respectively. In Table 4.2, one can see that the hybrid has the best performance against three of four the AASs. Notice also, that on the three for which the hybrid has the best performance, that the hybrid dramatically reduces the identification accuracy. In terms of the average reduction against the four well-known AASs, the hybrid has the greatest average reduction (-32%), followed by AuthorCAAT-V (-14%), AIM-IT (-11%), Rahgouy (-10%), Mihaylova (-8%), and Castro (4%). Tables 4.3 and 4.4 show the detailed results of the classification of the adversarial texts when submitted to the three AASs (Table 4.3) and the CNN (Table 4.4). The leftmost column of each table shows the name of each of the 25 adversarial texts followed by the change in accuracy for each AAS, the average change in accuracy of the three systems for each author masking technique, the accuracy for each attribution system with respect to the AMT, and the average accuracy for each attribution system. In the tables, '✗' means neither the original nor the adversarial texts were correctly classified, '✓' means both the original and the adversarial texts were correctly classified, 'TF' means the original text was correctly classified and the adversarial text was incorrectly classified, and 'FT' means the original text was incorrectly classified and the adversarial text was correctly classified. In Tables 4.3 and 4.4, the last four rows represent the change in accuracy given an AAS or the CNN (denoted as Δ Acc.), the average change in accuracy (denoted as Δ Acc. Avg.), the reduced adversarial accuracy (denoted as RAA), and the average RAA (denoted as RAA Avg.). In Table 4.3, one can see that had a total of 10 TFs, 47 ✓s, and 1 FT. This can be

	Castro	Mihaylova	Rahgouy	AIM-IT	AuthorCAAT-V	AuthorCAAT-V + AIM-IT
Keselj2003	-0.12	-0.16	-0.16	-0.20	-0.20	-0.16
Teahan2003	0.00	-0.04	-0.12	0.84	-0.08	-0.32
Koppel2011	0.00	-0.04	0.04	0.84	-0.08	-0.52
CNN	0.00	-0.08	-0.08	-0.08	-0.20	-0.28

Table 4.2: Results of the Reduction in Accuracy Based on the Adversarial Texts

	AuthorCAAT-V			AIM-IT			Mihaylova			Rahgouy			Castro			AuthorCAAT-V + AIM-IT		
	Keselj03	Teahan03	Koppel11	Keselj03	Teahan03	Koppel11	Keselj03	Teahan03	Koppel11	Keselj03	Teahan03	Koppel11	Keselj03	Teahan03	Koppel11	Keselj03	Teahan03	Koppel11
1000_4	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	FT	✓	✓
1001_4	✗	TF	✗	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	TF	✗
1002_4	TF	✓	TF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
1003_4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	TF	TF	TF
1004_4	✓	✓	✓	TF	✓	✓	✓	✓	✓	TF	TF	✓	✓	✓	TF	TF	TF	TF
1005_4	TF	TF	TF	TF	✓	TF	TF	✓	✓	✓	✓	✓	TF	✓	✓	✓	✓	✓
1006_4	TF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	TF
1007_4	TF	✓	✓	✓	✓	✓	TF	✓	✓	✓	✓	✓	✓	✓	✓	TF	✓	TF
1008_4	FT	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗		✗	✗	✓	✗	FT	TF
1009_4	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	FT	✓	✓
1010_4	✗	✓	✗	✗	TF	FT	✗	✓	✗	✗	✓	✓	✗	✓	✓	FT	✓	✓
1011_4	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓
1012_4	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	FT	✓	✓	✗	✓	✓	✗
1013_4	TF	✓	✓	TF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	TF	✓	TF
1014_4	✗	✓	✗	✗	TF	✗	✗	✓	✗	✗	TF	✗	✗	✓	✗	FT	✓	✗
1015_4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	TF
1016_4	✓	✓	✓	✓	✓	✓	TF	✓	TF	TF	✓	✓	✓	✓	✓	TF	TF	TF
1017_4	✗	✓	✓	✗	TF	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	TF	TF
1018_4	✓	✓	✓	✓	✓	✓	TF	✓	✓	TF	✓	✓	TF	✓✓		TF	TF	TF
1019_4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
1020_4	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	FT	✓	✓
1021_4	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
1022_4	✓	✓	✓	✓	✓	✓	✓	TF	✓	✓	✓	✓	✓	✓	✓	TF	TF	TF
1023_4	✓	✓	✓	TF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	TF	TF	TF
1024_4	TF	✓	✗	TF	✓	✗	✓	✓	✗	TF	✓	✗	✓	✓	✗	✓	✓	✗
Δ Acc	-20%	-8%	-8%	-20%	-12%	0%	-16$	-4%	-4%	-16%	-12%	4%	-12%	0%	0%	-16%	-32%	-52%
Δ Acc. Avg		-12%			-11%			-8%			-8%			-4%			-33%	
RAA	40%	84%	68%	40%	80%	76%	44%	88%	72%	44%	80%	80%	48%	92%	76%	44%	60%	24%
RAA Avg.		64%			65%			68%			68%			72%			43%	

Table 4.3: Detailed Authorship Attribution Results

represented using the following notation, <10, 47, 1>. Similarly, the performances of AIM-IT, Mihaylova, Rahgouy, Castro, and the hybrid can be represented as <8, 47, 1>, <6, 51, 0>, <7, 49, 1>, <4, 52, 1>, and <30, 27, 5>. Given these results one can see that the hybrid dramatically outperforms the other AMTs in terms of evading detection by the AASs (with 30 TFs) while having only 27 ✓s. However, this improvement in performance of the hybrid comes at the cost of an increased number of FTs. In Table 4.4, one can see the performances of the five AMTs are as follows <5, 16, 0> for AuthorCAAT-V, <3, 19, 0> for AIM-IT, <2, 19, 0> for Mihaylova, <2, 19, 0> for Rahgouy, <1, 20, 1> for Castro, and <7, 14, 0> for the hybrid.

Overall, the performance of AIM-IT and AuthorCAAT-V was equal to or better than the performances of three state-of-the-art AMTs. Furthermore, our results show that the hybridization of AuthorCAAT-V and AIM-IT provides a greater reduction in the identification rate against three of the four well-known AASs. See Tables 4.5 and 4.6 for examples of adversarial text generated using AuthorCAAT-V and the hybrid configuration of AuthorCAAT-V and AIM-IT.

	AuthorCAAT-V	AIM-IT	Mihaylova	Rahgouy	Castro	AuthorCAAT-V + AIM-IT
	CNN					
1000_4	✓	✓	✓	✓	✓	✓
1001_4	TF	✓	TF	✓	✓	TF
1002_4	TF	✓	✓	✗	✓	TF
1003_4	✓	✓	✓	✓	✓	✓
1004_4	✓	✓	✓	✓	✓	✓
1005_4	TF	TF	✓	✓	✓	TF
1006_4	✓	✓	✓	✓	✓	✓
1007_4	✓	✓	✓	✓	✓	✓
1008_4	✗	✗	✗	✗	✗	✗
1009_4	✓	✓	✓	✓	✓	✓
1010_4	✓	✓	✓	✓	✓	✓
1011_4	✓	✓	✓	✓	TF	✓
1012_4	✗	✗	✗	✗	✗	✗
1013_4	✗	✗	✗	✗	FT	✗
1014_4	TF	TF	✓	TF	✓	TF
1015_4	✓	✓	✓	✓	✓	✓
1016_4	✓	✓	✓	✓	✓	✓
1017_4	✓	✓	✓	✗	✓	✓
1018_4	✓	TF	✓	TF	✓	✓
1019_4	✓	✓	✓	✓	✓	✓
1020_4	✓	✓	TF	✓	✓	TF
1021_4	✗	✗	✗	✗	✗	✗
1022_4	✓	✓	✓	✓	✓	✓
1023_4	TF	✓	✓	✓	✓	TF
1024_4	✓	✓	✓	✓	✓	TF
Δ Acc	-20%	-12%	-8%	-8%	0%	-28%
Δ Acc. Avg	-20%	-12%	-8%	-8%	0%	-28%
RAA	64%	72%	76%	76%	84%	56%
RAA Avg.	64%	72%	76%	76%	84%	56%

Table 4.4: Detailed Convolutional Neural Network Results

	Text
Original	India affirms to be a country which is elusive and mysterious because of the innumerable features that it houses within its terrain. Nobody can pen down all of the alluring facets that India beholds as it is a nation which is blessed with profound beauty that words can't describe.
Adversarial	Because of its myriad benefits, India declares itself as a distant and mysterious state. Everyone can not penetrate all the attractive aspects of the attractive country India sees and this country has beauty that can not be expressed in words.

Table 4.5: Adversarial Text Created with AuthorCAAT-V

	Text
Original	In an analysis which is difficult to understand in a presidential race in Ghana, Asiedu Nketia stated that Nana Akufo – Addo won four regions in the 2008 election against Prof John Evans Atta Mills but got only two regions in 2012 elections against President John Dramani Mahama. One is advised to refrain from arguments that tend to link us to regionalism, ethnicity and tribalism.
Adversarial	In an incomprehensible analysis of the Ghanaian presidential campaign, Asidou Nkia said Nana Akufo-Addo won four election in the 2008 general election against Professor John Evans Atta Mills, but there were only two district won in the 2012 election. Arguments concerning regionalism, ethnicity and tribalism are not recommended.

Table 4.6: Adversarial Text Created with AuthorCAAT-V + AIM-IT

4.5 Summary

In this chapter, our goal was to evolve adversarial text that preserved context and concealed the identity of the author. We compared five AMTs for adversarial authorship. All of the performances with respect to Δ_{Acc} were fairly close. Overall, the performance of AIM-IT and AuthorCAAT-V was equal to or better than the performances of three state-of-the-art AMTs. Furthermore, our results show that the hybridization of AuthorCAAT-V and AIM-IT provides a greater reduction in the identification rate against three of the four well-known AASs.

Chapter 5

Towards a Human-AI Hybrid for Adversarial Authorship in a Partially-Observable Environment

5.1 Introduction

We have previously seen how these systems respond to adversarial authorship attempts when these author masking systems are effectively in the dark. Now we are going to explore exactly how these systems respond to adversarial authorship when someone finally turns on the lights. Creating adversarial text without knowing what features an AAS is focusing on when attributing text can be difficult. The features changed when creating the adversarial text may not correspond with the features that are being analyzed by the AAS. In this chapter, we attempt to improve the effectiveness of adversarial text by using AASs in conjunction with our masking techniques to create a partially observable environment in which we can traverse. Additionally, we seek to explore the interaction between some AASs and what relationships can be discovered to help future efforts in adversarial authorship.

In the previous chapter we saw that teahan03 and koppel11 performed significantly better than keselj03. We noticed that whenever keselj03 correctly attributed a text at least teahan03 or koppel11 were also able to correctly attribute it. Using this information, we can conclude that teahan03 and koppel11 form a minimal set of AASs that are needed for accurate attribution.

When examining these AASs, it is important to understand how each of the systems work and how this could affect the minimal set relationship. Koppel11 is an n-gram method using 4-grams and a limit of the 20,000 most common features. An author profile is created of the candidate authors. Then, a random subset of the unknown text is compared to the author profiles using cosine similarity. This process is done 100 times, and the candidate author with

the most votes is chosen as the author of the unknown text. Teahan03 also uses n-grams, but is using unigrams to create author profiles for the candidate authors. The unknown text is then compared against the author profiles using cross-entropy, and the author of the unknown text is predicted. Due to the random nature of the pieces of text chosen in koppel11, the results are not deterministic. To counter this, we run the system five times and use majority voting to determine an author. This value was determined to be the lowest value to give a deterministic author identification through our evaluation.

5.2 Experiments

Here we examine a partially observable approach to creating adversarial text with two masking approaches; AuthorCAAT-VI, AIM-IT. It is partially observable because of the AASs being used within the process of creating the adversarial text. In [39] only support vector machines built in to each masking system were used for classification. We compare the effects of using AASs that are in the minimal set within the process of creating adversarial text with the masking systems. We also try a combination of the AASs that are both in and not in the minimal set.

In AuthorCAAT-V the 25 authors are the first 25 authors of the CASIS-1000 dataset, but in AuthorCAAT-VI a dataset of any 25 authors can be uploaded. AuthorCAAT-VI also adds the option of mutating the text by way of paraphrasing.

In these experiments, each of the two masking systems were used in collaboration with the two AASs that form the minimal set to create adversarial texts. We used three of the texts that were used for creating adversarial texts in [39]. These texts come from three different authors in CASIS-25, a subset of the CASIS-1000 dataset. For further testing of AIM-IT we created adversarial text using all of the same texts that were used in the experiment from the previous chapter. All three of the combinations of the two aforementioned AASs were used with the two masking methods. The configurations include:

1. Teahan03,
2. Koppel11,
3. Teahan03 and Koppel11

The adversarial text was then tested using the two AASs. It should be noted that when using the Koppel11 attribution algorithm, the process was completed a number of times and the results were averaged. This is because of the random seed generated by the system that is used in part of the attribution. For our purposes, the number of runs used to average the results was 5, as it has proven to be the lowest number of runs to give consistent and repeatable results.

When creating adversarial text with AuthorCAAT-VI we would go through the regular process of using the system, as described previously, but we would periodically run the AAS(s) after changes were made to the text. If the AAS(s) correctly attribute the text then we would continue making changes to the text. Once the text was classified wrong, it was saved as the adversarial text. If it is classified wrong the majority of those times it is then saved. The CASIS-25 dataset was also used for the set of the 25 author targets in AuthorCAAT-VI. While AuthorCAAT-VI doesn't use GEFeS for the datasets that are uploaded (all features are used), we were able to utilize the feature masks created with GEFeS for the CASIS-25 dataset.

We hoped that by using the various AASs for training and evaluating its effects on testing, we could find a connection between the AASs that could be exploited to advance the research for a minimal set spanning all AASs. Previously, we stated that keselj03 was shown to not be in the minimal set, and similar data might be obtainable when the AASs are available during training.

5.3 Results

The results were generated by applying the adversarial authorship techniques to the fourth writing instances of three authors from the CASIS-1000 dataset. Tables I and III provide the results of our experiment. The first column contains the names of the three samples that were used to create the adversarial text followed by the change in accuracy for each of the AASs. The top row represents which AAS or set of AASs were used in the process of creating the adversarial text. The next row shows the name of the AASs used for testing. The values in these tables are formatted in the same manner as in the previous chapter. In the tables, '' means neither the original nor the adversarial texts were correctly classified, '' means both the original and the adversarial texts were correctly classified, 'TF' means the original text was correctly

classified and the adversarial text was incorrectly classified, and 'FT' means the original text was incorrectly classified and the adversarial text was correctly classified.

Tables 5.1 through 5.3 provide the results of our experiment. Tables 5.1 and 5.3 show the classification of the five texts when being masked with AuthorCAAT-VI and AIM-IT respectively. The first column contains the names of the five samples that were used to create the adversarial text. The top row represents which AAS or set of AASs were used in the process of creating the adversarial text. The next row shows the name of the AASs used for testing. The bottom row shows the change in accuracy from the original to the adversarial texts. Table 5.2 mirrors the previously mentioned tables, but it contains the results for AIM-IT being run on the entirety of the test set.

One important relationship we determine is the conditional probability of incorrect attribution. We examine the probability of incorrect attribution using koppel11 given that it was incorrectly attributed using teahan03, and vice versa. We found that, when using AuthorCAAT-VI, the probability of incorrect attribution using koppel11 given that it was incorrectly attributed using teahan03 is 93%. As it so happens, the probability of incorrect attribution using tehan03 given that it was incorrectly attributed using koppel11 is also 93%. We found that, when using AIM-IT, the probability of incorrect attribution using koppel11 given that it was incorrectly attributed using teahan03 is 100%. As it so happens, the probability of incorrect attribution using tehan03 given that it was incorrectly attributed using koppel11 is also 100%. When run on the entirety of the test set, AIM-IT yields more useful results. We found that using AIM-IT on the full test set the probability of incorrect attribution using koppel11 given that it was incorrectly attributed using teahan03 is 64%. As it so happens, the probability of incorrect attribution using teahan03 given that it was incorrectly attributed using koppel11 is also 78%.

In each of the tables, there is a column titled "LSVM" that represents the respective results from [39] where the training process was aided by a linear support vector machine as opposed to the work of this paper where the training process was aided by AASs. As you can see, the performance of AuthorCAAT-VI and AIM-IT are both superior when operating in a partially observable environment. Table 5.1 shows that AIM-IT performs at least as well if not better

Train	LSVM		teahan03		koppel11		teahan03, koppel11	
Test	teahan03	koppel11	teahan03	koppel11	teahan03	koppel11	teahan03	koppel11
1002_4	✓	✓	✓	✓	✓	✓	✓	✓
1004_4	✓	✓	TF	TF	TF	TF	TF	TF
1005_4	✓	TF	✓	✓	✓	✓	✓	✓
1007_4	✓	✓	✓	✓	✓	✓	✓	✓
1009_4	✓	✓	✓	✓	✓	✓	✓	✓
Δ Acc.	0%	-20%	-20%	-20%	-20%	-20%	-20%	-20%

Table 5.1: AIM-IT RESULTS

than the blind environment. Table 5.2 shows that AIM-IT performs better in all cases when operating in a partially-observable environment. Table 5.3 shows that AuthorCAAT-VI performs much better in a partially observable environment and was able to cause incorrect attribution in all instances of the AAS it was using to train. The current state-of-the-art in author masking is proposed by PAN @ CLEF to be 10-12% [54] and both AuthorCAAT-VI and AIM-IT exceed these performance benchmarks.

As previously mentioned, the attribution performance of keselj03 can be achieved between a combination of koppel11 and teahan03. The results above demonstrate that koppel11 and teahan03 are closely connected when attempting to classify adversarial texts. The ultimate goal of was to discover the relationships between AASs and the potential to exploit these relationships to the advantage of anonymity moving forward. The vast number of AASs currently known to the public is so large that it would be nearly impossible to learn to evade them all individually. Also, it would be considerably difficult just to run all of these systems in a reasonable amount of time. However, there may exist a subset of AASs whose performances subsume that of all other AASs. In this case, it would not be necessary to use all of the AASs for training in order to evade attribution. Rather, we could just use those dominant systems to achieve the same result in a fraction of the time.

Note that AIM-IT is capable of generating a much larger volume of adversarial texts than its counterpart AuthorCAAT-VI. This is only achievable due to the automation driving AIM-IT and the lack of human oversight needed to generate the texts. The tables demonstrate that those adversarial texts generated by AuthorCAAT-VI typically outperform those of AIM-IT head-to-head. This superior performance is likely a result of the human interaction and intelligence

Train	LSVM		teahan03		koppel11		teahan03, koppel11	
Test	teahan03	koppel11	teahan03	koppel11	teahan03	koppel11	teahan03	koppel11
1000_4	✓	✓	✓	TF	✓	TF	✓	✓
1001_4	✓	✗	✓	✗	✓	✗	✓	✗
1002_4	✓	✓	✓	✓	✓	✓	✓	✓
1003_4	✓	✓	✓	✓	✓	✓	✓	✓
1004_4	✓	✓	TF	TF	TF	TF	TF	TF
1005_4	✓	TF	✓	✓	✓	✓	✓	✓
1006_4	✓	✓	✓	✓	✓	✓	✓	✓
1007_4	✓	✓	✓	✓	✓	✓	✓	✓
1008_4	✗	✓	✗	✓	✗	✓	✗	✓
1009_4	✓	✓	✓	✓	✓	✓	✓	✓
1010_4	TF	FT	TF	✓	TF	TF	TF	TF
1011_4	✓	✓	✓	✓	✓	✓	✓	✓
1012_4	✓	✗	✓	✗	✓	✗	✓	✗
1013_4	✓	✓	✓	✓	✓	✓	✓	✓
1014_4	TF	✗	TF	✗	✓	✗	TF	✗
1015_4	✓	✓	✓	✓	✓	✓	✓	✓
1016_4	✓	✓	✓	✓	✓	✓	✓	✓
1017_4	TF	✓	✓	✓	✓	✓	✓	✓
1018_4	✓	✓	TF	✓	TF	TF	TF	TF
1019_4	✓	✓	✓	✓	✓	✓	✓	✓
1020_4	✓	✓	✓	✓	✓	✓	✓	✓
1021_4	✗	✗	✗	✗	✗	✗	✗	✗
1022_4	✓	✗	✓	✓	✓	✓	✓	✓
1023_4	✓	✓	✓	✓	✓	✓	✓	✓
1024_4	✓	✗	✓	✗	✓	✗	✓	✗
Δ Acc.	-12%	0%	-16%	-8%	-12%	-16%	-16%	-12%

Table 5.2: COMPLETE AIM-IT RESULTS

Train	LSVM		teahan03		koppel11		teahan03, koppel11	
Test	teahan03	koppel11	teahan03	koppel11	teahan03	koppel11	teahan03	koppel11
1002_4	✓	TF	TF	✓	TF	TF	TF	TF
1004_4	✓	✓	TF	TF	TF	TF	TF	TF
1005_4	TF	TF	TF	TF	✓	TF	TF	TF
1007_4	✓	✓	TF	TF	TF	TF	TF	TF
1009_4	✓	✓	TF	TF	TF	TF	TF	TF
Δ Acc.	-20%	-40%	-100%	-80%	-80%	-100%	-100%	-100%

Table 5.3: AUTHORCAAT-VI RESULTS

that is at the very core of AuthorCAAT-VI. It should be noted that when running AIM-IT, the system has the option to terminate running if a text is incorrectly attributed or the masking efforts have stalled. This varies from the way AuthorCAAT-VI, which is run until the text is incorrectly attributed. This method could not be used with AIM-IT because this would very likely cause the system to run indefinitely. But these results perfectly demonstrate the viability of a hybrid system that retains the performance of AuthorCAAT-VI and the speed of AIM-IT.

5.4 Summary

In this chapter we built on the work of chapter 4 by utilizing the AASs in our approach to creating adversarial text. As previously mentioned, the attribution performance of keselj03 can be achieved between a combination of koppel11 and teahan03. The results above demonstrate that koppel11 and teahan03 are closely connected when attempting to classify adversarial texts.

The ultimate goal of this work is to discover the relationships between AASs and the potential to exploit these relationships to the advantage of anonymity moving forward. The vast number of AASs currently known to the public is so large that it would be nearly impossible to learn to evade them all individually. Also, it would be considerably difficult just to run all of these systems in a reasonable amount of time. However, there may exist a subset of AASs whose performances subsume that of all other AASs. In this case, it would not be necessary to use all of the AASs for training in order to evade attribution. Rather, we could just use those dominant systems to achieve the same result in a fraction of the time.

Note that AIM-IT is capable of generating a much larger volume of adversarial texts than its counterpart AuthorCAAT-VI. This is only achievable due to the automation driving AIM-IT and the lack of human oversight needed to generate the texts. The tables demonstrate that those adversarial texts generated by AuthorCAAT-VI typically outperform those of AIM-IT head-to-head. This superior performance is likely a result of the human interaction and intelligence that is at the very core of AuthorCAAT-VI.

It should be noted that when running AIM-IT, the system has the option to terminate running if a text is incorrectly attributed or the masking efforts have stalled. This varies from the way AuthorCAAT-VI, which is run until the text is incorrectly attributed. This method could not

be used with AIM-IT because this would very likely cause the system to run indefinitely. But these results perfectly demonstrate the viability of a hybrid system that retains the performance of AuthorCAAT-VI and the speed of AIM-IT.

Chapter 6

The JohariMAA Concept

6.1 Attempts to Further Improve AuthorCAAT's Mutation Process

After seeing how well the adversarial texts performed in [39] when they were created in AuthorCAAT-V and then ran through AIM-IT, we tried incorporating the author masking techniques into AuthorCAAT. AuthorCAAT-VII adds Castro, Mihaylova, and Rahgouy to the list of operations that the user can choose from to create the CT. We attempted to add Castro, Mihaylova, and Rahgouy into the Hill-Climbing as well. This addition resulted in the Hill-Climbing taking several minutes longer to run due to each of the author masking techniques taking over a minute to run individually. They also did not improve the quality of the texts that were created with the Hill-Climbing. So not only did adding Castro, Mihaylova, and Rahgouy increase the time it took do a single run of Hill-Climbing, it prolonged the overall time it took to use the system because the Hill-Climbing would need to be ran more times to get a suitable CT. The paraphraser was added into the Hill-Climbing, as the time it takes to run is similar to that of the languages and the CT that is created does not seem to be negatively affected by its addition.

6.2 Limitations of AuthorCAAT

Using AuthorCAAT can be a very tedious process. AuthorCAAT has several steps that require the user to select from a list of options. Because of this, there are many different combinations of choices that can be made which could be overwhelming for a user.

AuthorCAAT displays multiple windows which gives the user more information than necessary. There is also the issue with how long it takes to anonymize text with the system. The hill climbing process can take several minutes to run and it also may need to be ran multiple times before a text is anonymized. With these long wait periods, a user can experience an increase in frustration and fatigue. A user might also decide to work on another task while waiting for their text to finish mutating which could lead to a longer time spent using the application.

Figures 6.1, 6.2, and 6.3 show the user interface of AuthorCAAT-VI and VII. There are three windows. The main window is where the user enters the text and assists in the anonymization process. The author scores window displays a visual representation of how similar the parent text is to each of the 25 authors from the background dataset based on the selected feature set. The authorship attribution window is where the user has the option to check the anonymity of their text against three authorship attribution systems: keselj03, koppel11, teahan03. AuthorCAAT currently has 10 buttons, three dropdown menus, and three windows. All of these components can be a bit overwhelming for the user. The way AuthorCAAT has been constructed leaves a very narrow opening for innovation.

6.3 Vision for JohariMAA

Our main goal is to create a human-AI collaborative system that performs adversarial authorship effectively. Our basic idea for JohariMAA is a very straightforward, streamlined design. The framework of JohariMAA consists of three main components: selection, mutation, and prediction. This framework gives more flexibility to explore different methods of performing adversarial authorship while maintaining the same interface. Keeping the process of using JohariMAA uniform as we explore various configurations of the framework components will be helpful for testing purposes. It will allow for users to produce adversarial text samples more easily as they will not have a complicated process to learn with each configuration change.

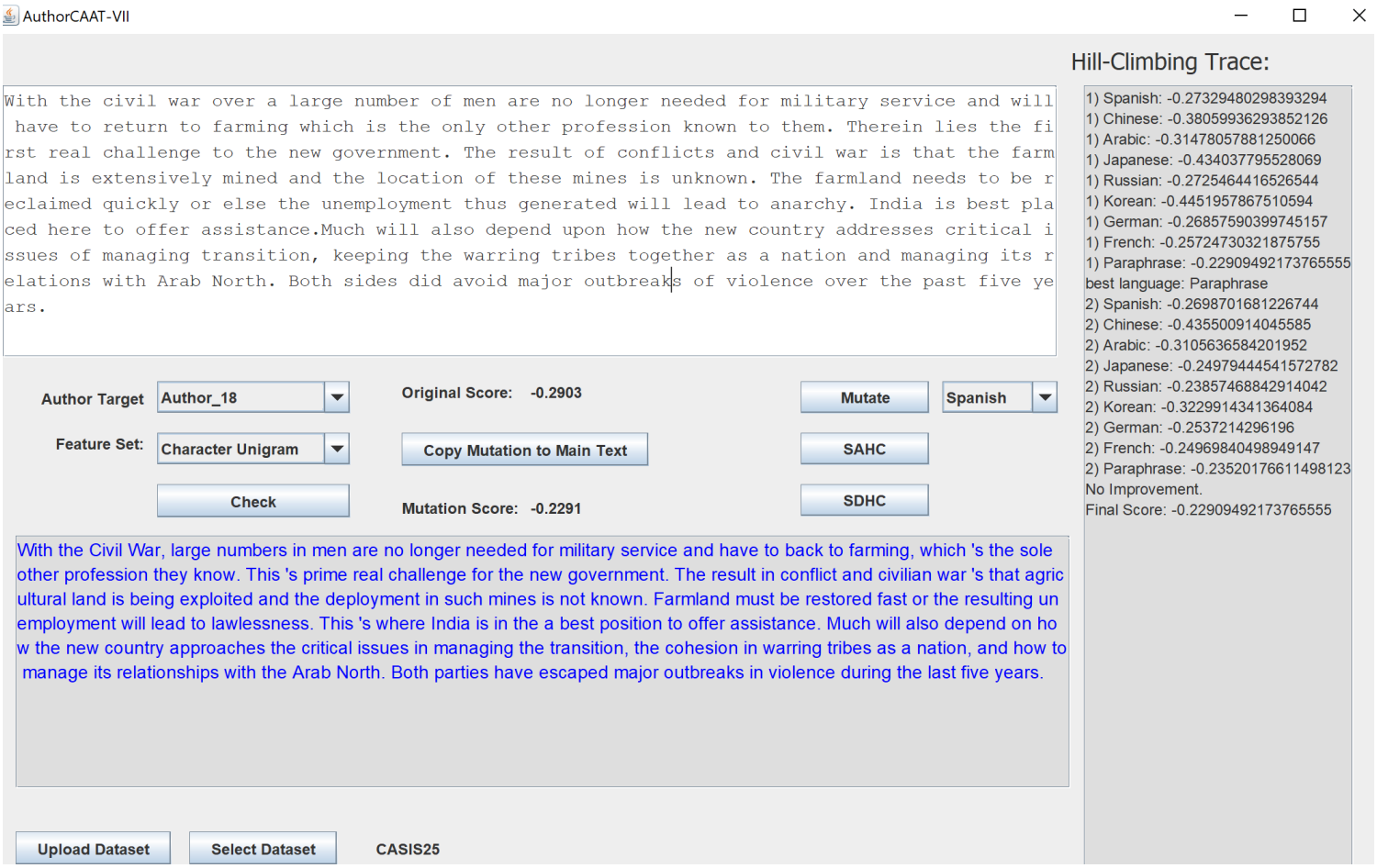

Figure 6.1: The AuthorCAAT-V Interface

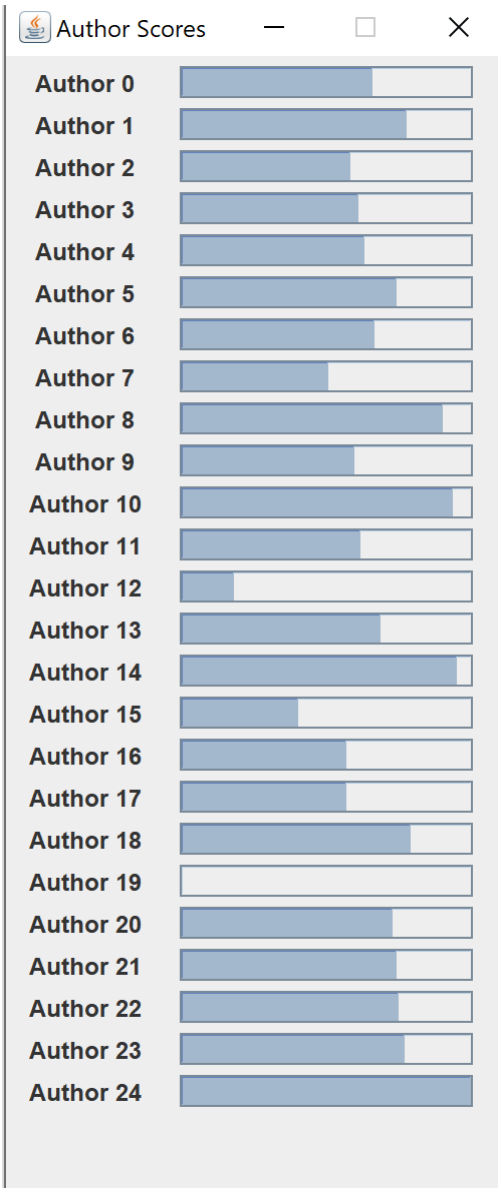

Figure 6.2: The Author Scores Window of the AuthorCAAT-VI Interface

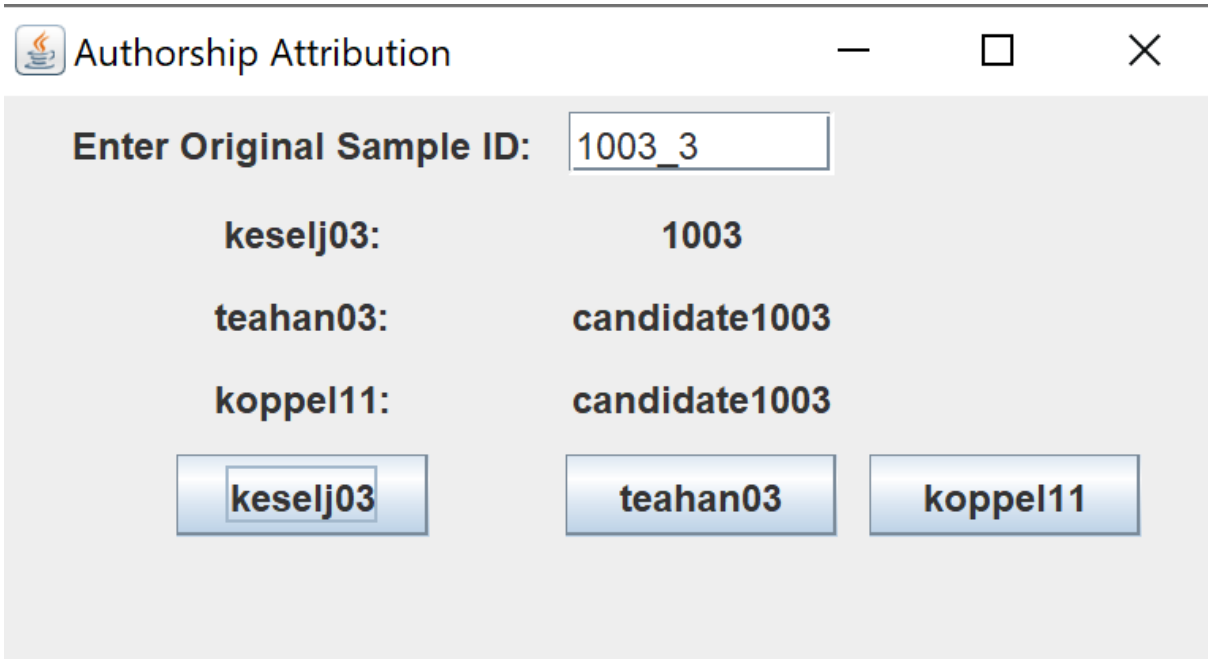

Figure 6.3: The Authorship Attribution Window of the AuthorCAAT-VI Interface

Chapter 7

The Interaction Design of JohariMAA: Reducing the Complexity of Design

7.1 Examining the Usability of AuthorCAAT

We conducted a preliminary study on AuthorCAAT-V. For the study we used AuthorCAAT-V because it is the last stable version and proved effective in performing Adversarial Authorship [39]. AuthorCAAT-V does not include the authorship attribution window that is included in AuthorCAAT-VII. We asked participants to use AuthorCAAT-V to create an adversarial text. We gave a set of instructions to the participants that guided them through their usage of AuthorCAAT-V. Participants began the process within an initial text sample from the CASIS dataset.The experimental process is to modify this text sample with the aid of AuthorCAAT interactively. After the experiment, a survey was completed consisting of the ten questions from the System Usability Scale [53] and asked participants about any issues encountered when using the program.

Our study gave us a more in depth look at the usability issues within AuthorCAAT. One of the issues users experienced was the program failing. When this would happen the user would only be able to identify the cause of the error if they ran the program from the command line because AuthorCAAT would just stop responding without any error messages. This would cause the user to have to close the program and restart it.

Some users still experienced confusion about the process of using AuthorCAAT even though instructions were given and were unable to complete the task of creating adversarial text. There were ten steps included within the instructions to run AuthorCAAT. Some of the steps are not as straightforward as others. For example, after users enter the text that they would

like to anonymize and choose which feature set they would like to utilize, they must look at the 25 author scores and select an author target that they would like their text to mimic. This choice is entirely up to the user. While there is not necessarily a correct choice, there can be choices that are unhelpful in the anonymization process. For example, if a chosen target is too far away from the original, getting the text identified as the chosen author's target could be impossible. But this could also be the case if the choice is too close to the original author. This part of the process is a bit of trial and error. The user picking an unhelpful choice leads to prolonging their time spent using the program. One user reported spending over three hours trying to create the adversarial text before giving up. Some of the other decisions left up to the user include the mutation method and whether or not they will accept the resulting text as their new PT. All of these decisions made by the user influences the process significantly. The more unhelpful choices the user makes, the longer the process takes. The longer the process takes, the more fatigued the user becomes. To give the users a better chance of creating adversarial text, and doing so in a reasonable amount of time, we need to reduce the potential for unhelpful choices.

7.2 Designing JohariMAA

Now that we have examined the usability issues within AuthorCAAT, we will use this section to discuss the three designs we went through to create the user interface for JohariMAA. JohariMAA is meant to be a generalized playground used by researchers to experiment with various approaches to adversarial authorship. The functional requirements for JohariMAA include:

1. User should be able to enter text
2. User should be able to select a feature set
3. System should be able to calculate a list of author scores when given a text and feature set
4. User should be able to select an author target
5. User should be able to mutate text

6. User should be able to replace the parent text with the mutated text

Improving the flow of information is a key factor in improving how the user experiences a program. Using either multiple pages or combining and simplifying options to reduce the clutter should make it easier for a user to understand how to properly use the program. Displaying every option at once can be confusing for the user, especially when there is no clear labeling to show the order that the steps are to be completed. Giving only the necessary information and steps in a program like JohariMAA will make learning the program easier for the user.

The design will also need to include several other features that were missing from AuthorCAAT, such as displaying error messages and better error handling. Having a tutorial option and tooltips can assist with the learnability of the program. Instead of displaying the hill-climbing trace to notify the users that the mutation process is in progress, we can display processing messages. The user experience would benefit from a reduction in choices for the author target. This could be done by presenting the user with a reduced set of potential author targets based on the feature set that they select. Another possibility to explore is getting rid of the choice entirely and just running the mutation function for each author target in the reduced set and presenting the user with the mutated text for each option.

Figure 7.1 shows our first iteration of the design for JohariMAA's user interface. Our main goal with the initial design was to reduce the number of windows being used. We combined the author scores window with the main window by eliminating the hill climbing trace panel. If we're going to have a more generalized user interface, we would need to remove the hill climbing trace panel so that we can experiment with approaches to mutation that do not involve the hill climbing search algorithm. This design keeps the two horizontal text boxes for the parent and mutated text. The panel in between the text boxes has the dropdown menu for the feature sets to the left of the author target label. These items appear in the opposite ordering of AuthorCAAT because the feature set should be selected before the author target, being that the author prediction scores are based on the selected feature set. The author target label changes based on the selection made by the user. As previously mentioned, we removed the hill climbing trace panel and replaced it with the list of potential author targets. Each author target option consists of a numbered author label and a bar used to visualize the prediction score. Once the

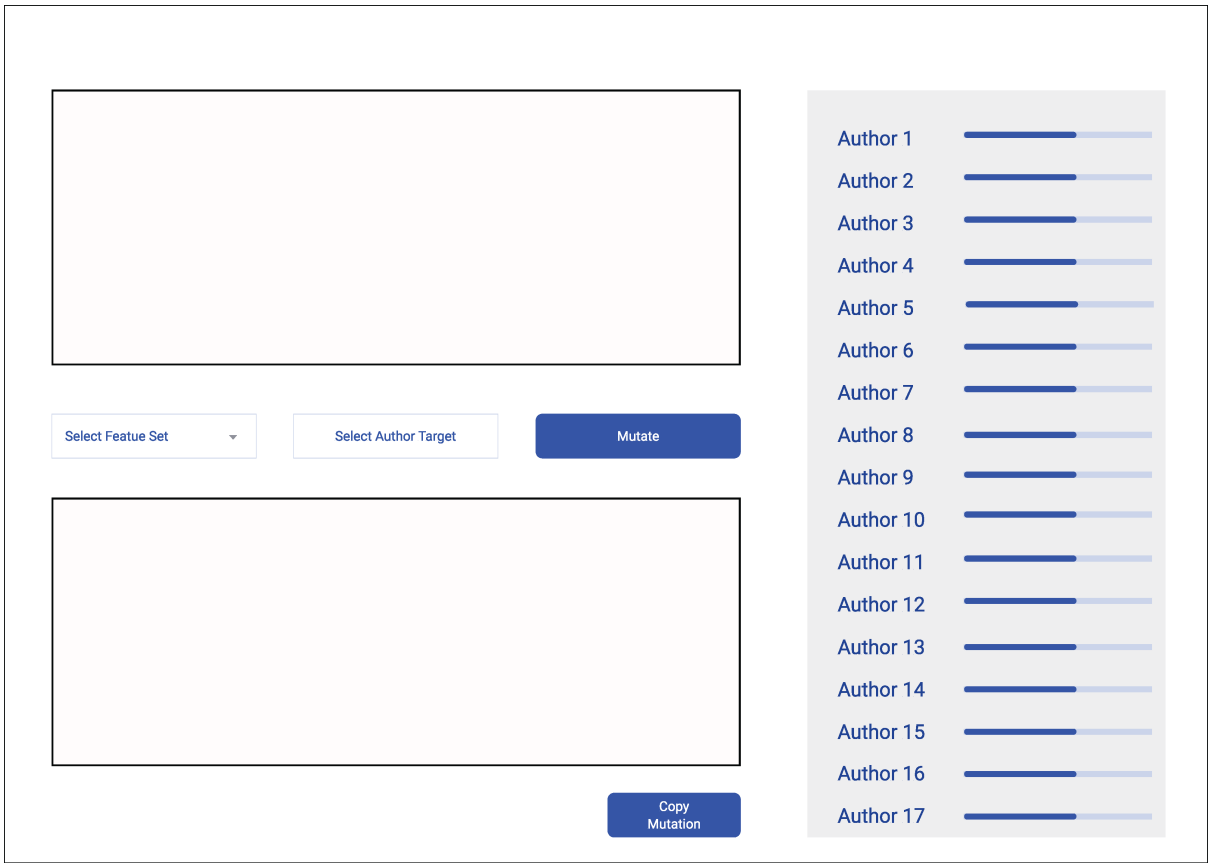

Figure 7.1: Initial interface design for JohariMAA.

user selects an author target from the panel, the author number and the numerical representation of the prediction score is presented in between the feature set dropdwon and mutate button. The button to copy the mutated text to the parent text box is located below the text box for the mutated text.

Figures 7.2, 7.3, and 7.4 show another iteration of the design for JohariMAA where we moved further away from the AuthorCAAT design. Both text boxes for the parent and mutated texts are now displayed vertically side by side. Between these two text boxes lies a column of four buttons. The buttons are ordered so that the button on top is the first action that the user takes after entering their text. The user then proceeds through the process in the order that the buttons appear. Clicking the first button triggers a dropdown menu to pop up that presents the user with the available feature sets. Once the user makes a selection, the dropdown menu closes and the author scores are calculated. Then the author scores button can be clicked so that another dropdown menu opens to present the user with a list of potential author targets. Each

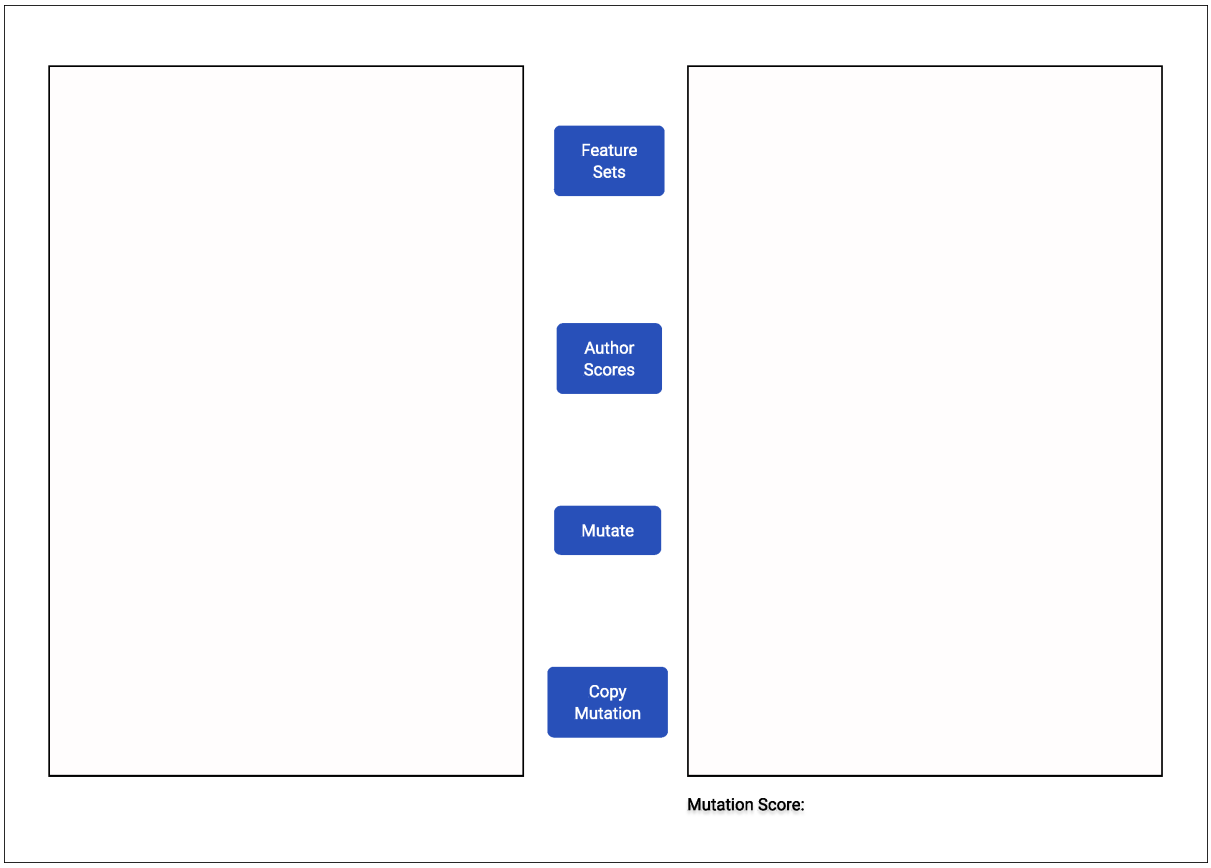

Figure 7.2: The main view of the second iteration for the interface design of JohariMAA.

author target option consists of a numbered author label and a radial progress bar depicting the prediction score converted to a percentage.

The design shown in Figure 7.5 is most similar to our final design for JohariMAA. This design consists of a text box where users can enter their text and view the mutated texts by using the tabs labeled for each mutation. The tabs for the mutated text will appear dynamically depending on how many mutations the user chooses to create. Once a tab is selected for the mutated text there will be an option to move that text to the parent tab. To the right of the text box is three buttons that each expand into a panel of more options and information about each step of the process. Above this set of buttons is a menu button. The menu button gives users access to options like downloading the adversarial text and instructions for the program. A user can only select a step in the process if the previous step has been configured (e.g. a user can only select author target(s) if they select a feature set first). The buttons that are unavailable for use will be denoted by being a lighter color. When you click the first button it expands into a

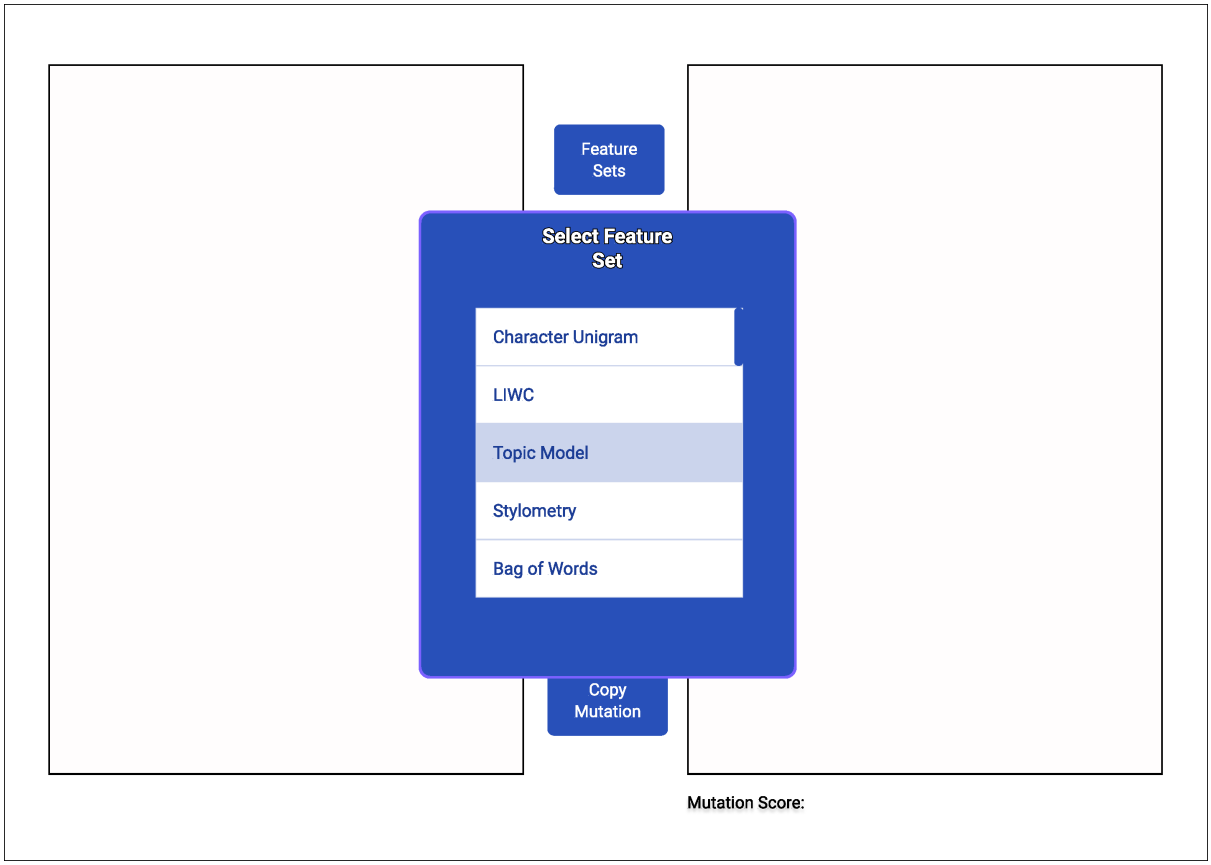

Figure 7.3: View of the feature selection function for the second iteration of the interface design for JohariMAA.

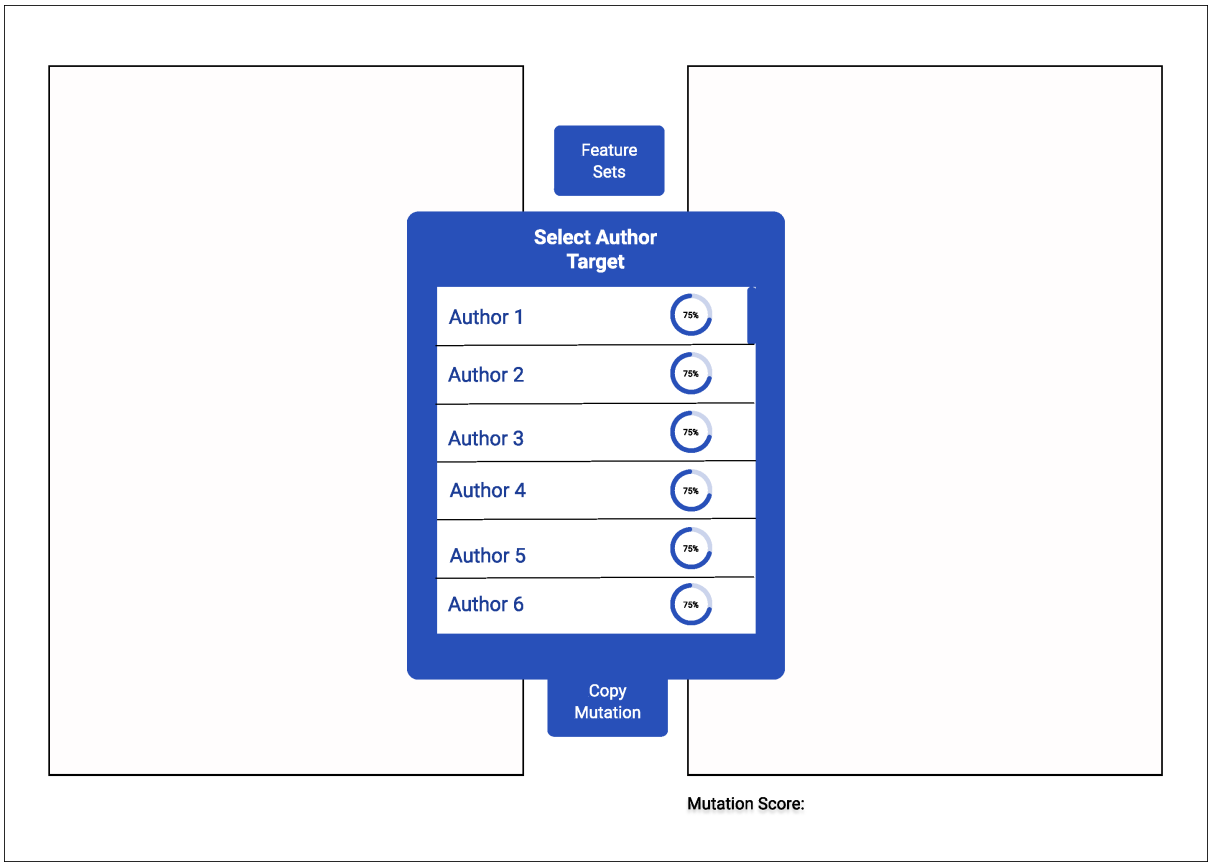

Figure 7.4: View of the author target selection function for the second iteration of the interface design for JohariMAA.

Figure 7.5: The main view of the interface design for JohariMAA.

panel that shows a dropdown bar consisting of all the available feature sets (see Figure 7.6). After making the feature set selection, the panel collapses back into a button, and the button for the next step is then available to be selected. These are some of the potential design choices that are meant meant to lead the user through the process without them having to guess what step is next. The options are concealed for each step until they are in use to minimize interface clutter.

We decided that it would be best to get rid of multiple buttons for mutation and use a single mutate button that performs steepest ascent hill climbing. Steepest ascent hill climbing was chosen as the sole mutation method because using a single language to translate does not change the text enough so that the prediction scores are affected. The steepest descent hill climbing button also was less effective than steepest ascent hill climbing. When using steepest descent hill climbing, we experienced longer run times and incomprehensible text.

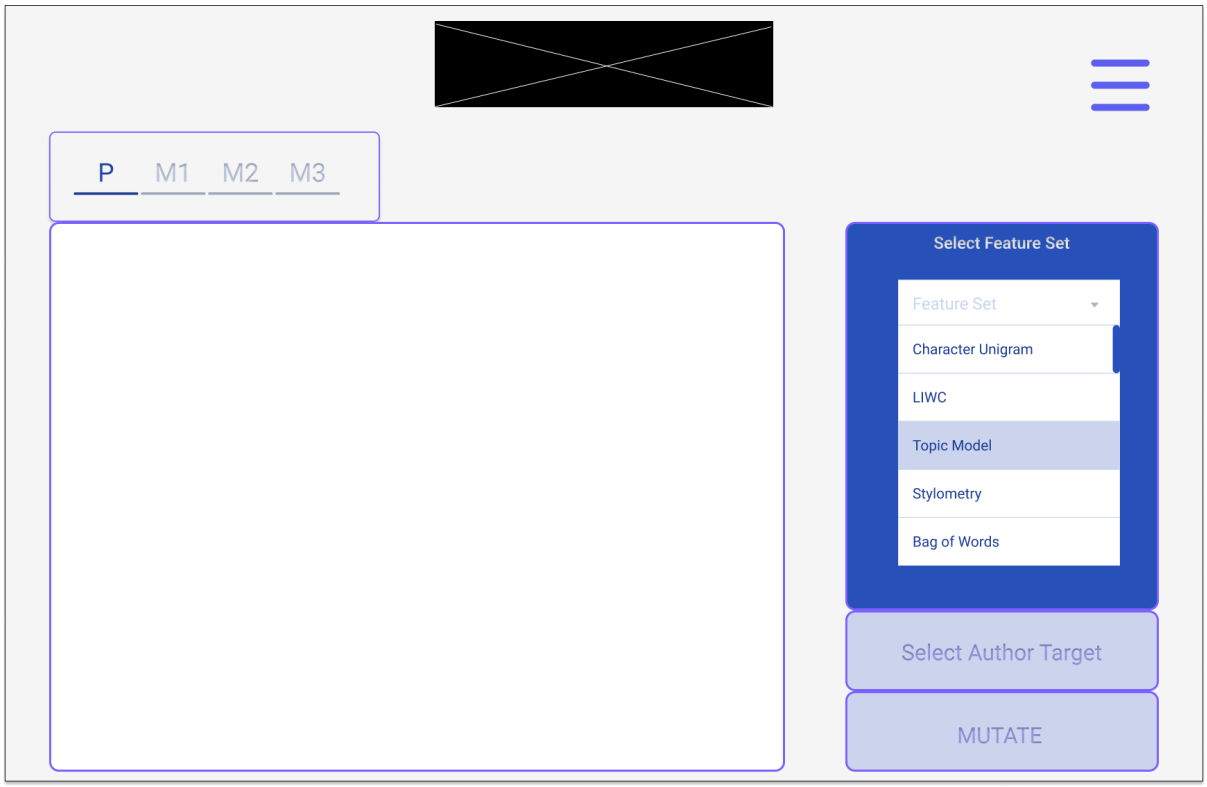

Figure 7.6: JohariMAA interface design depicting the select feature set button expanding into a dropdown menu.

7.3 Implementation

JohariMAA was developed in Microsoft Visual Studio 2022 using the Windows Presentation Foundation (WPF) framework. WPF was chosen because of the way it separates design and programming with XAML and C#. Python 3 is also used in JohariMAA to script some functionalities. The XAML version of the Material Design UI library was utilized for the user interface. AuthorCAAT was previously developed in Apache Netbeans using Java. Java Swing and Abstract Window Toolkit were used for the user interface. For our initial implementation of JohariMAA, we used the same approach to adversarial authorship as AuthorCAAT since it has proven to be successful for anonymization. This means that we are still using machine translation within a hill climbing algorithm. Also, we are using the same feature sets and datasets for the list of author target options.

Figures 7.7, 7.8, 7.9, and 7.10 show the JohariMAA user interface. The text boxes for the parent and mutated texts are organized in a tabbed format. The selection for feature sets

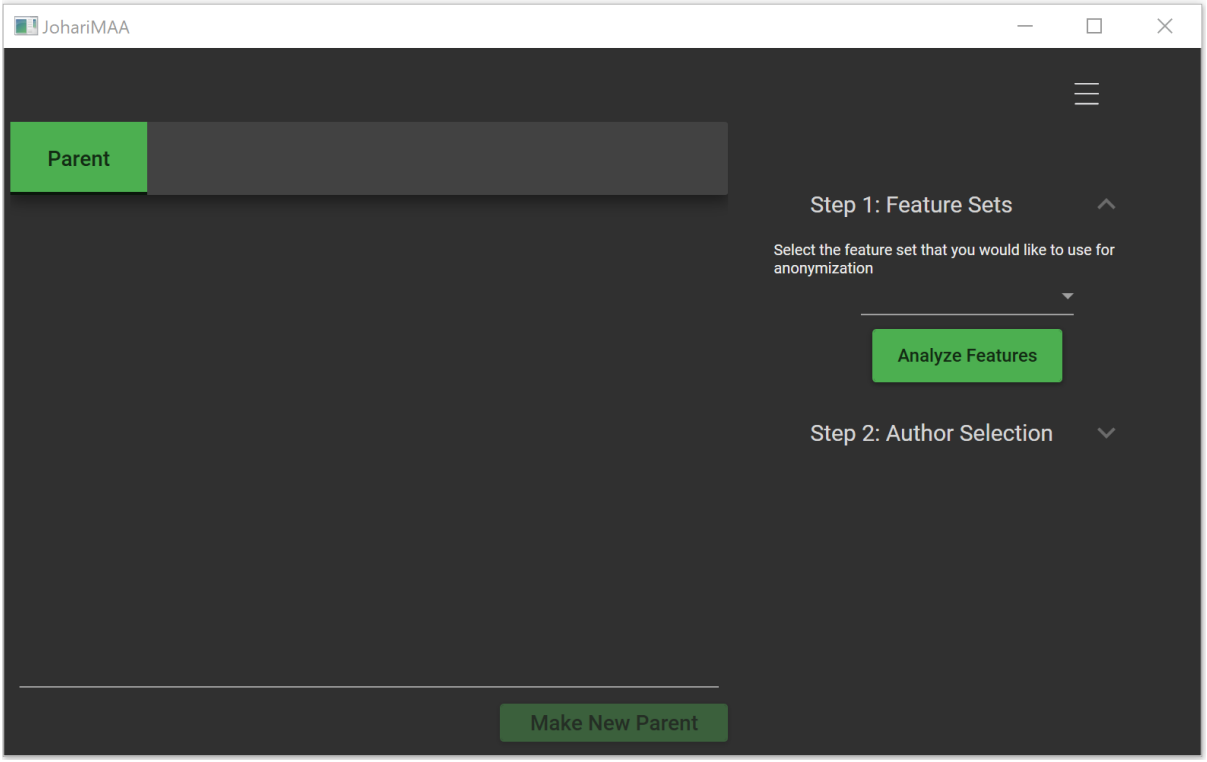

Figure 7.7: The user interface of JohariMAA.

and author targets are located in two expandable areas. The application starts with only the area for the feature sets expanded to help reduce any confusion that the user may have on which order selections need to be made. When the user makes their selection and clicks the ”Analyze Features” button, step 2 expands and step 1 collapses. A circular progress bar is used to visualize the author identification scores. The numerical values next to them indicate the similarity in the form of a percentage. Previously in AuthorCAAT, we had a separate window depicting the author score and the selection of the author target would be done in the main window. In JohariMAA, we combine these two components so the user just clicks directly on the author target. Once the text is mutated, the second tab appears where the mutated text is displayed. The ”Make New Parent” button is only enabled when the mutation tab is selected. This button copies the text from the mutation tab to the parent tab and deletes the mutation tab.

7.3.1 Feature Extraction and Prediction

We created models for each feature set. The character unigram and LIWC models were created in Python using scikit-learn. The model for the topic model feature set was created in C# using

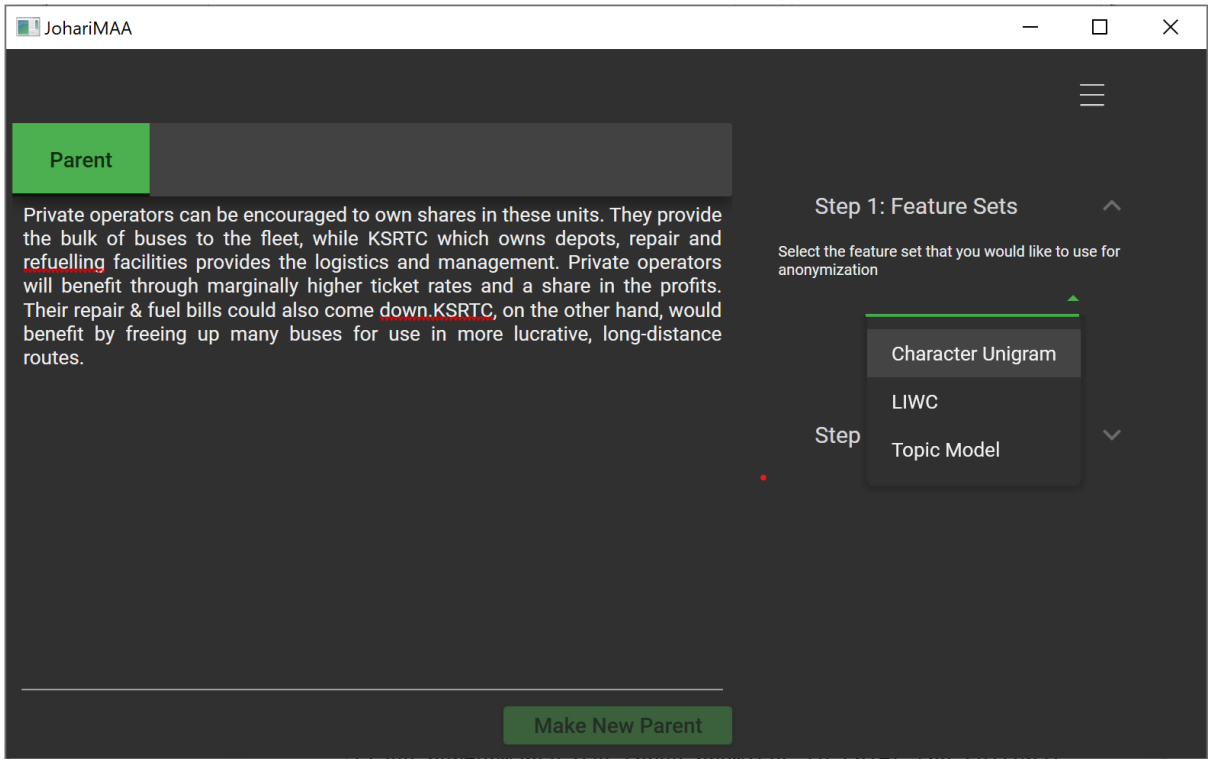

Figure 7.8: Selecting a feature set in JohariMAA.

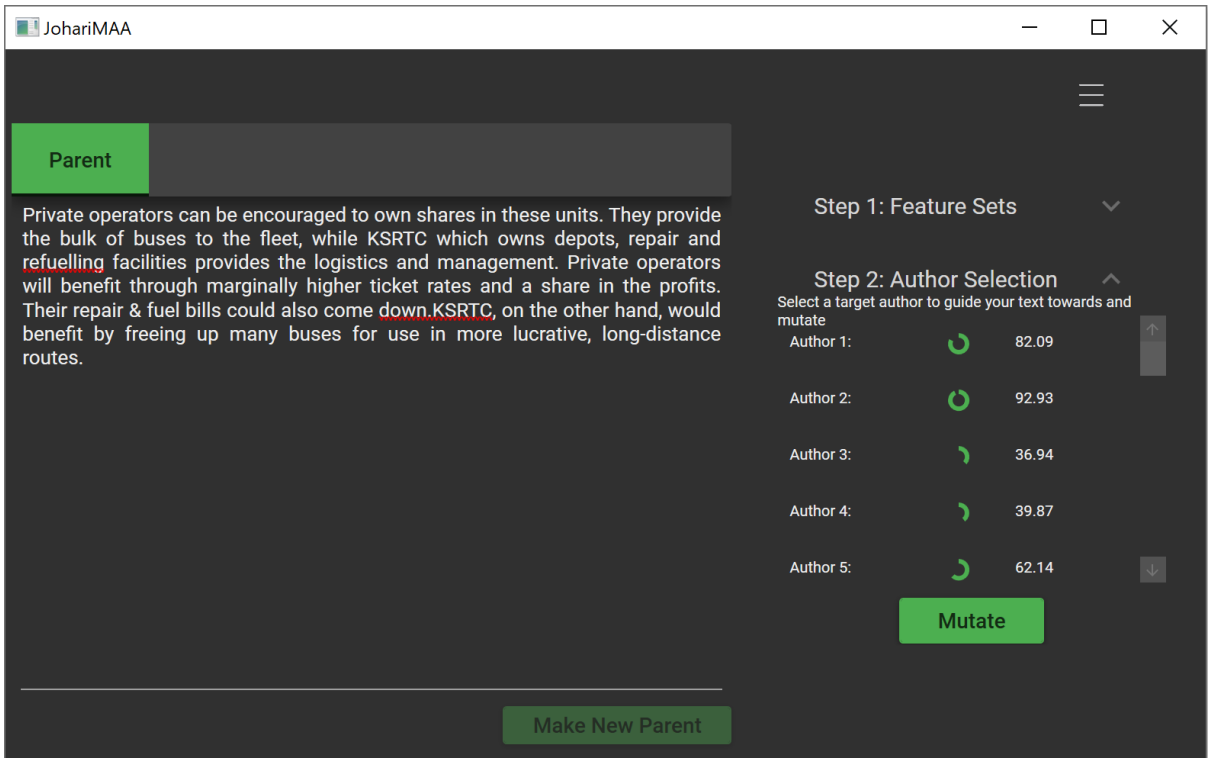

Figure 7.9: Selecting an author target in JohariMAA.

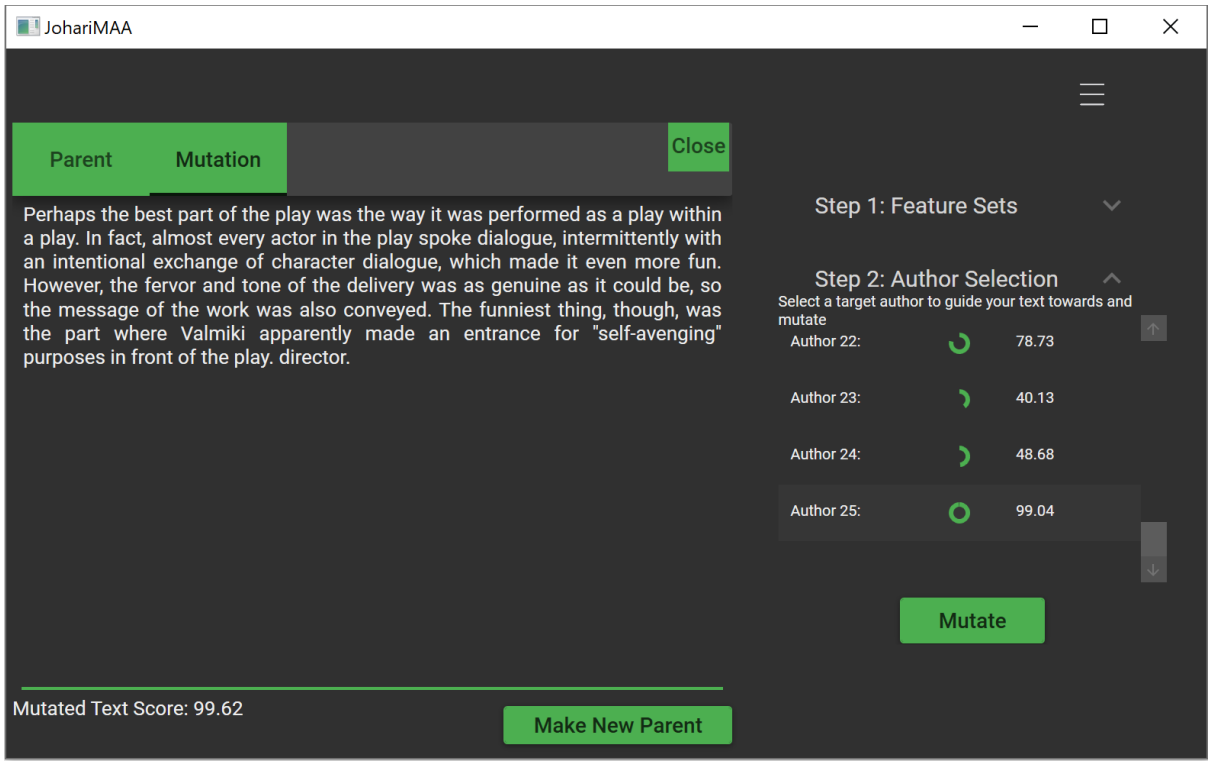

Figure 7.10: Mutation in JohariMAA.

the ML.NET machine learning library. The models are created for every feature set and dataset (mentioned in chapter 3) pair.

For our author identification system we use a python script that loads the previously mentioned models and passes the feature vector representing the text into the model to return the scores from the decision function for each author. In AuthorCAAT, the models were not saved. This meant that each time we were to run the prediction script, we preprocessed the features of the entire dataset before passing in the feature vector of the text we were trying to predict. We also got rid of some unnecessary overhead by sending the feature vector directly to the python script whereas in AuthorCAAT the feature vector was written to a file that was read by the python script.

7.3.2 Mutation

Our method for mutating the text employs the Google Translate API for our hill climbing algorithm. Our hill climbing algorithm starts by taking the user's text and passing it to the

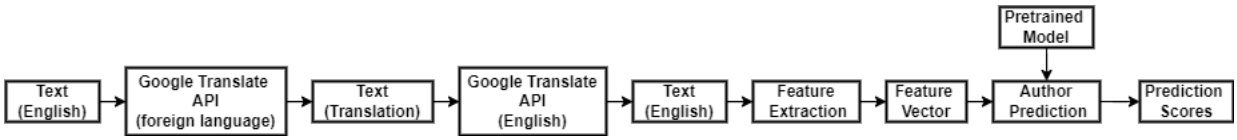

Figure 7.11: A single mutation iteration within the hill climbing algorithm.

Google Translate API to translate it to one of eight foreign languages (Spanish, German, Chinese, Korean, Arabic, Japanese, French, Russian) before passing this translation back to the Google Translate API to retrieve the English translation. Features are then extracted from this new English text and then passed through the pretrained model to get the author identification scores (see Figure 7.11). This is done for each language and the text where the author target's prediction score has improved the most is chosen as the text to send back through the Google Translate API for each language.

In AuthorCAAT, the hill climbing algorithm would run for a single language at a time, or synchronously. For JohariMAA we have made the algorithm asynchronous. This means that all of the calls to the Google Translate API for each language will start immediately rather than waiting on one language to completely go through the translation and prediction process before moving on to the next language.

7.3.3 Offline Version

We created a separate version of JohariMAA that does not rely on APIs so that it can be used without an internet connection. This means that the APIs for the LIWC feature set and Google Translate were removed and replaced. In the place of LIWC, we use Empath[57]. Empath is a feature set similar to LIWC that consists of 194 built-in topical and emotional categories of 59,690 words that were generated by using the neural embeddings of approximately 1.8 billion words of fiction and the resulting vector space to determine their cosine similarity. Empath also offers the option to generate new categories with seed terms given by the user. Some examples of categories within Empath include school, fear, contempt, and social media.

The Google Translate API was replaced with an mBART model that has been fine-tuned for multilingual machine translation [58]. This model allows for translations between any pair

of 50 langauges. We used the same languages that were used with the Google Translate API (Spanish, German, Chinese, Korean, Arabic, Japanese, French, Russian).

7.4 Summary

In this chapter, we evaluated AuthorCAAT and observed issues within the tool that contributed to user frustration and fatigue. Some of those issues were directly related to the user interface while others were more so related to the tool's functionality. These issues were taken into account when designing and implementing JohariMAA. AuthorCAAT's user interface had many components that we reduced to a less complex, more user-friendly design. The latest version of AuthorCAAT consisted of 10 buttons, three dropdown menus, and three windows. JohariMAA consists of four buttons, one dropdown menu, and one window. We organized the layout in a way that is less cluttered, therefore making the program easier to navigate. We also made sure to include better feedback to the user through error messages and loading messages. We improved the hill climbing algorithm by implementing it asynchronously as opposed to the synchronous configuration in AuthorCAAT. We were able to reduce the time complexity of the hill climbing algorithm from quadratic time to linear time. We also improved the author identification mechanism by utilizing saved models rather than recreating them each time.

Chapter 8

Leveraging Model Interpretability for Adversarial Authorship

8.1 Introduction

In this chapter, we begin looking into ways to make improvements on the components of the JohariMAA framework. We examine how clustering documents before performing adversarial authorship affects the ability of the classifier to correctly attribute a text to its original author. We do this by comparing a dataset that has been labeled in three different ways: by the original author, by clustering based on the TF-IDF representation, and by clustering based on a word embedding representation. We finetuned three DistilBERT models with these 3 sets of labels from the same dataset. We then utilize an model interpretability library to perform an attack on each model. Our hope is that mimicking a cluster rather than an individual author could potentially help better obfuscate an author by hiding amongst a more generalized feature profile and making it more difficult to pinpoint a specific author. It could also offer an opportunity to use a larger, more diverse dataset to make generating adversarial texts for different types of text (e.g. emails, blog posts, tweets, etc.) easier.

8.2 Dataset

Our experiments were performed on the C50 dataset from PAN [63]. The C50 dataset consists of 50 authors where each author has 100 samples. The dataset was split 60% for training, 20% for validation, and 20% for testing. Stratified sampling was used to keep the authors balanced across each set. The C50 dataset was preprocessed by removing URLs, emails, and extra whitespace. When clustering the datasets we tried two different representations for the

texts: Term Frequency–Inverse Document Frequency (TF-IDF) and word embeddings. TF-IDF is used to represent the words in a set of documents based on how rare they are in the set. Words that are rarer are given more weight whereas words that are common are assigned a lower weight. For the word embeddings, we utilized the Gensim library with a pretrained word vector model from the FastText library [64][66]. The pretrained model consists of 1 million word vectors trained with subword information from Wikipedia 2017, UMBC webbase corpus, and statmt.org news dataset. Each word embedding is 300 dimensions. For dimensionality reduction, we used truncated singular value decomposition (SVD).

For clustering the TF-IDF and word embedding representations of our dataset, we used K-means clustering. To determine a viable number of clusters we evaluated K values first by their silhouette score to narrow down our search to a few different K values, then we evaluated them using the Calinski-Harabaz Index and the Davies-Bouldin Index. For TF-IDF we went with a K value of 6 and for the word embeddings we went with a K value of 5.

8.3 Adversarial Text Generation Approach

For our initial attempt we trained a machine learning model on the embedding representation of our dataset using a pretrained FastText model with Gensim and then proceeded to perform synonym substitution on the words that were most influential on the prediction of the model. This approach did not work because the synonyms have similar embedding values, therefore the embedding representation would appear practically unchanged to the classifier even if the actual words were different. This would lead to none of the substitutions causing the classifier to misclassify the texts.

Our method leverages the interpretability of a transformer model. We use pretrained DistilBERT models for our experiments. DistilBERT is a smaller, faster version of BERT. This version of DistilBERT has six layers and 66 million parameters rather than the original version's 12 layers 110 million parameters. We use a library called Transformers Interpret that uses PyTorch's model interpretability tool Captum [68][69]. Captum provides the ability to interpret neural networks at the feature level. Transformers Interpret simplifies the process of using Captum with the transformers library. When a sample is passed to the interpreter, we

```
word_attributions = cls_explainer(text)
print(f"Text: {text}")
word_attributions
```

0.3s Python Python

```
Text: although a canadian federal election is not expected to be called until this weekend the mudslinging has already begun

[('[CLS]', 0.0),
 ('although', 0.16466716587022343),
 ('a', 0.3276135240589652),
 ('canadian', 0.2505077284974645),
 ('federal', 0.15497067175033694),
 ('election', 0.15669370051232598),
 ('is', 0.24785596228637974),
 ('not', 0.1762708935991498),
 ('expected', 0.11468274592209601),
 ('to', 0.22708135519372954),
 ('be', 0.15275231067569991),
 ('called', 0.26012392512293114),
 ('until', 0.38119367811449123),
 ('this', 0.30652626886531825),
 ('weekend', 0.03981159119581413),
 ('the', 0.20561479622806153),
 ('mud', 0.05337092448099583),
 ('##sling', 0.012339008779694845),
 ('##ing', 0.07203904787454071),
 ('has', 0.3289383542447527),
 ('already', 0.1703913812678424),
 ('begun', 0.27831273007095736),
 ('[SEP]', 0.0)]
```

Figure 8.1: Word Attributions Returned When Running Transformer Interpret on a Text Sample

can see exactly which tokens contribute to the prediction of the classifier both negatively and positively. These tokens are given scores where a positive number is indicative of a positive contribution and a negative number is indicative of a negative contribution to a predicted class. The sum of these scores is called the attribution score. The interpreter also offers you the ability to visualize the classifier's prediction by highlighting both the positive and negative words in green and red, respectively. Figures 8.2 and 8.3 highlight both the positive and negative attributions within a prediction. Figure 8.1 shows the words broken down into tokens and their contribution values.

For our method, we take the list of tokens and their scores to determine which words contribute to a text sample being correctly classified so that we will know which words to focus on when generating the adversarial sample. Because we are using a BERT language model, when the text is tokenized it uses a WordPiece tokenizer. This type of tokenizer breaks words down into subwords based on their frequency and distribution within the training data. The list of tokens that are returned by the interpreter consists of words and subwords along with their individual attribution scores. Before we know definitively which words have the highest scores, we recombine the tokens that are subwords into complete words and add their scores accordingly. Being that we are trying to get our classifier to misclassify the text, we are only

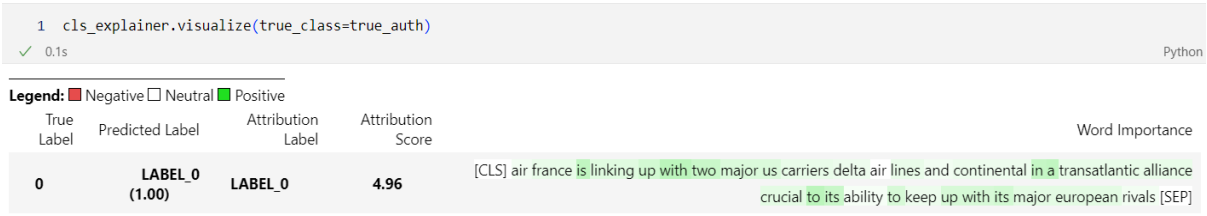

Figure 8.2: A visualization of the attribution of a correctly classified text using the Transformer Interpret Library

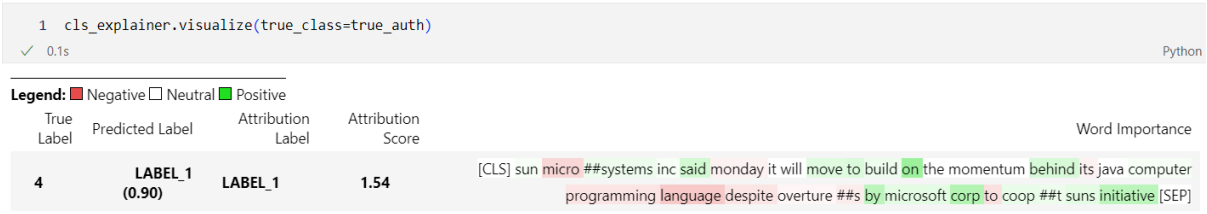

Figure 8.3: A visualization of the attribution of an incorrectly classified text using the Transformer Interpret Library

focusing on the words with positive scores. We take the list of positive words and get a set of synonyms for each word. We ignore stop words and named entities. For synonym retrieval we tried both WordNet and a pretrained word vector model from FastText [65][66]. We then went through the synonyms for each word one by one and replaced the original word within the text sample and reclassified the sample with the interpreter. If the synonym changed the classification of the text or decreased the overall attribution score, it would stay in the text sample. This process was repeated for each word that positively contributed to the correct attribution made by the classifier until the classification changed or every word had been tested in the text sample. The number of synonyms for each word was limited to 5.

8.4 Experiments

For our experiments, we finetuned three DistilBERT models for the task of text classification [67]. We used the Transformers library by HuggingFace and PyTorch to finetune the models. The first of our 3 classification models was trained on the 50 authors of the C50 datasets for 25 epochs using a dropout rate of 0.14 with the AdamW optimizer with a weight decay value of 0.02 and a learning rate of 0.00001. Figure 8.4 depicts a plot of the accuracy of the model during finetuning and 8.5 depicts a confusion matrix for the accuracy per class of the test set.

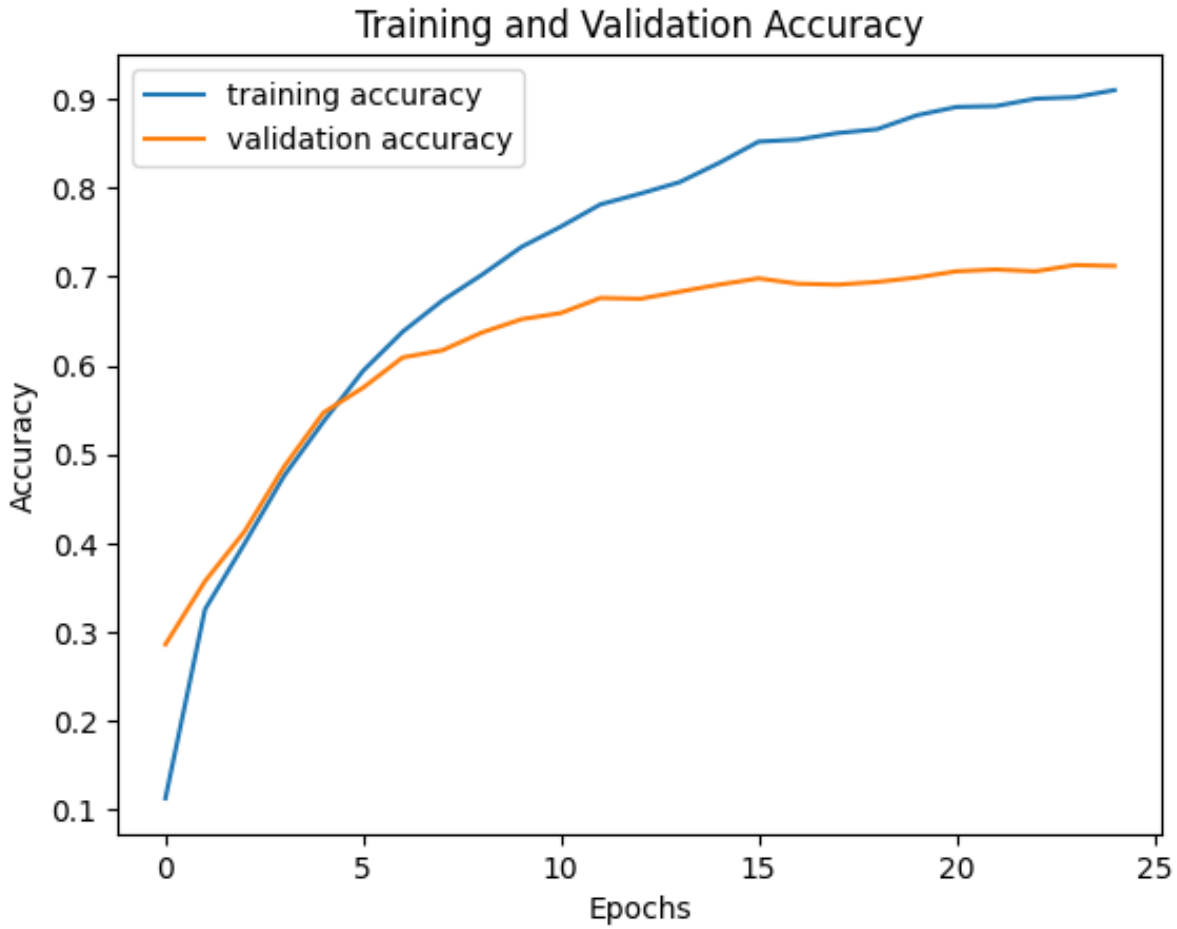

Figure 8.4: Accuracy while Finetuning a DistilBERT Model with the Author Labels of the C50 Dataset

The second model was trained on the 5 clusters of authors that were formed using K-means clustering on the C50 dataset where the text samples were transformed using the TF-IDF. Figure 8.6 depicts a plot of the accuracy of this model during finetuning and 8.7 depicts a confusion matrix for the accuracy per class of the test set. This model was trained for 25 epochs using a dropout rate of 0.14 and the RMSprop optimizer with a weight decay value of 0.01 and a learning rate of 0.00001. Another model was trained on 6 clusters of authors that were formed using K-means clustering of the C50 dataset where the text samples were transformed into word embeddings using a pretrained FastText model. This model was trained for 25 epochs using a dropout rate of 0.14 and the AdamW optimizer with a weight decay value of 0.01 and a learning rate of 0.00001. Figure 8.8 depicts a plot of the accuracy of this model during finetuning and 8.9 depicts a confusion matrix for the accuracy per class of the test set.

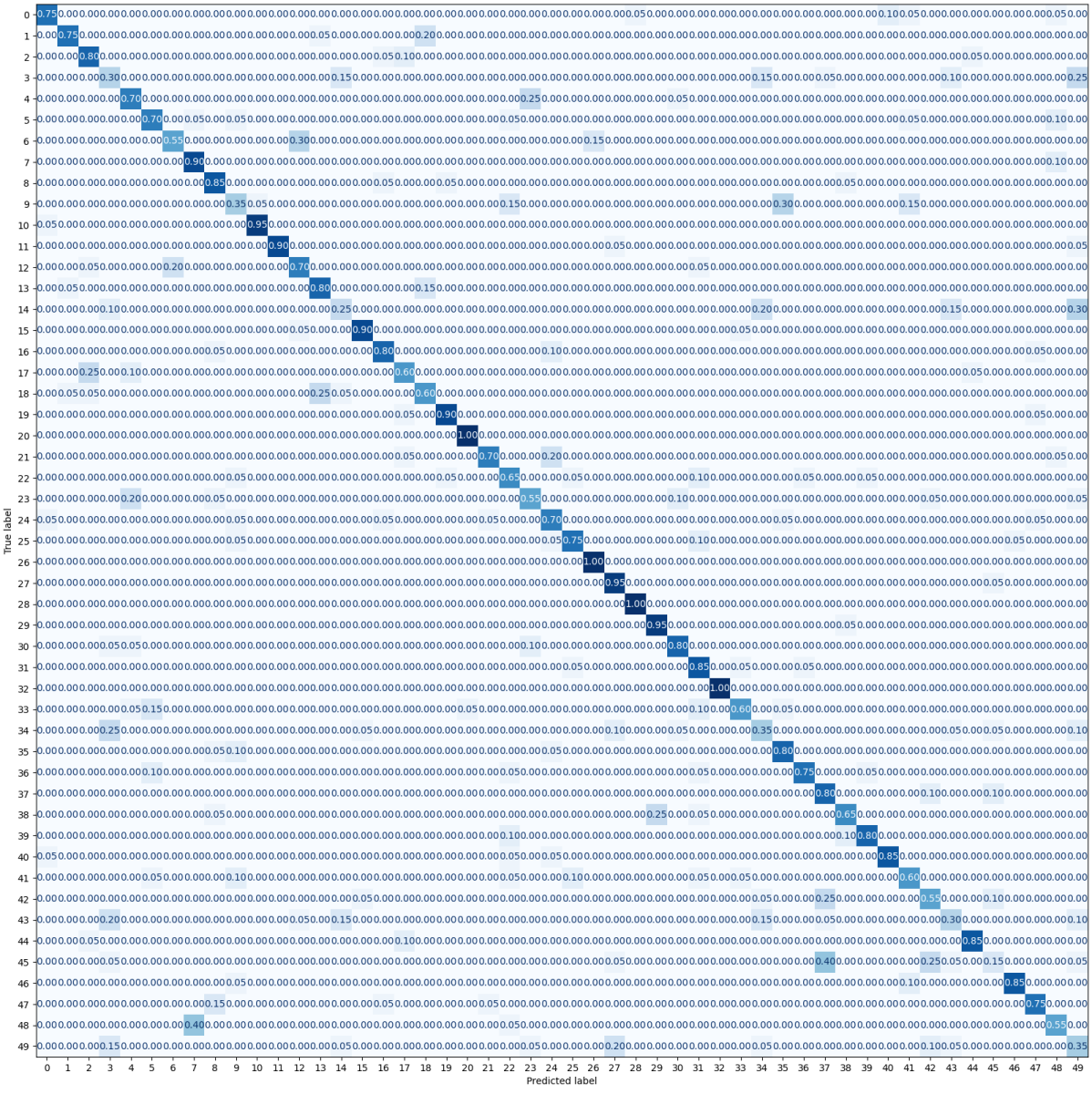

Figure 8.5: Normalized Confusion Matrix of the DistilBERT Model Finetuned with the Author Labels

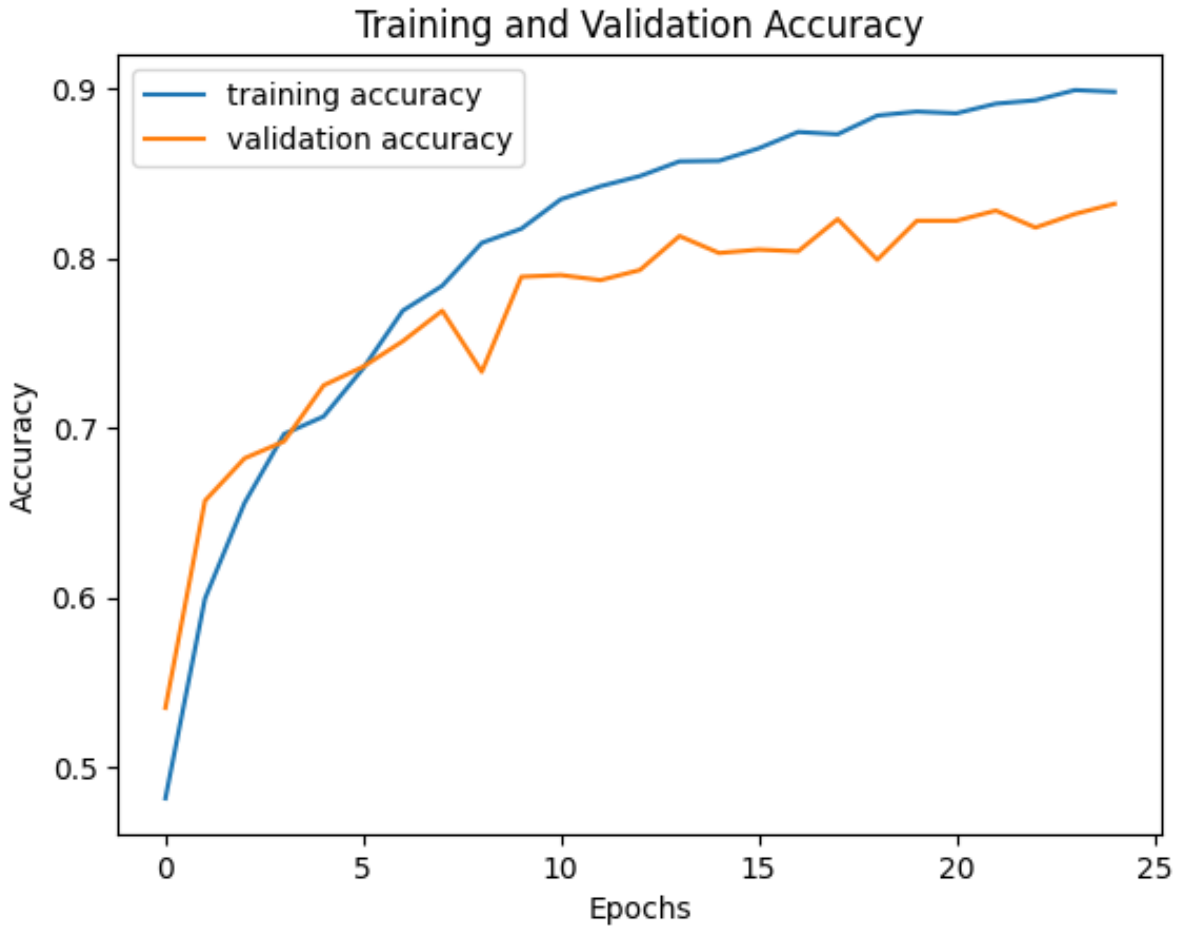

Figure 8.6: Accuracy while Finetuning a DistilBERT Model with the Author Labels of the C50 Dataset

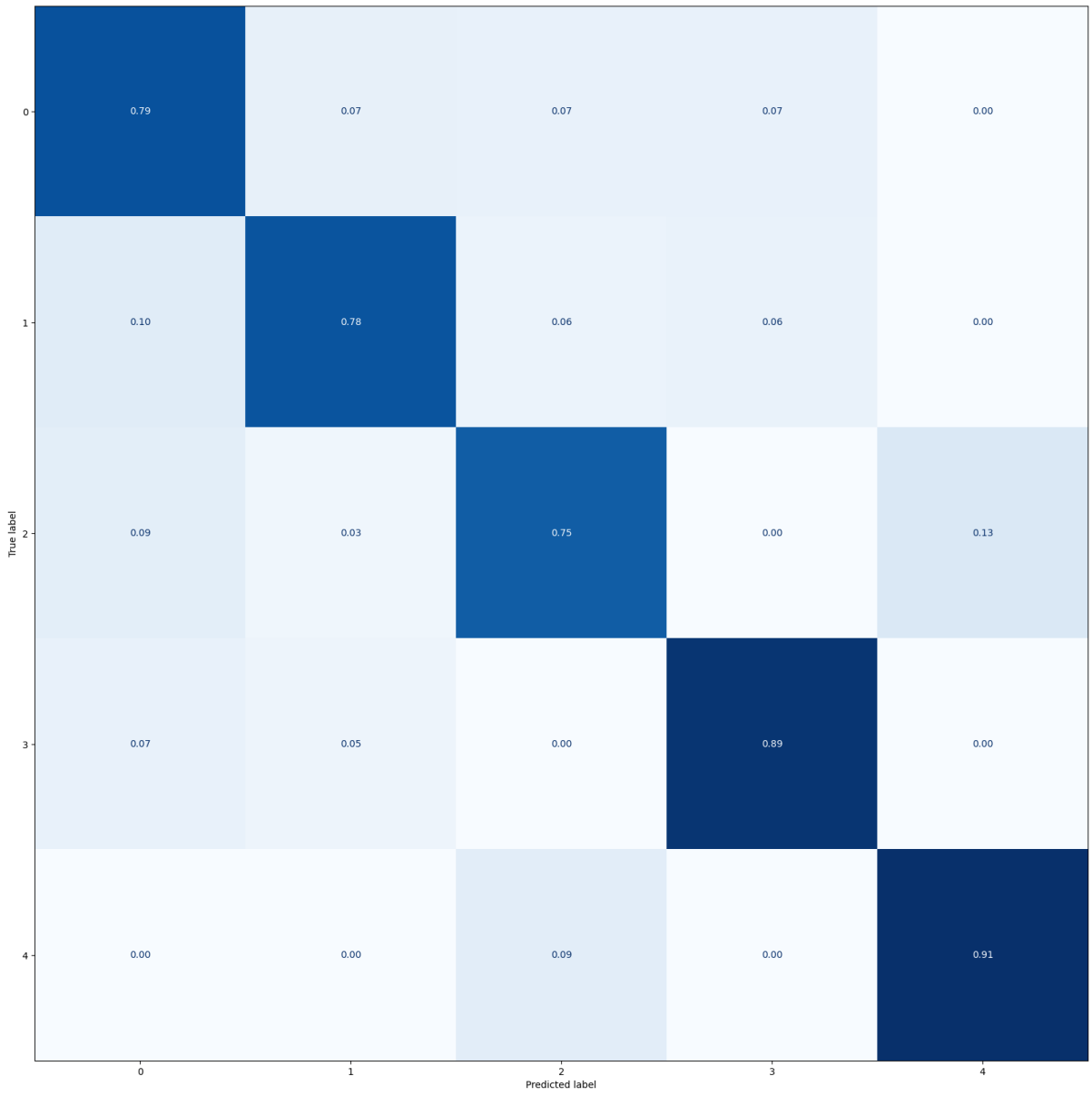

Figure 8.7: Normalized Confusion Matrix of the DistilBERT Model Finetuned with the TF-IDF Cluster Labels

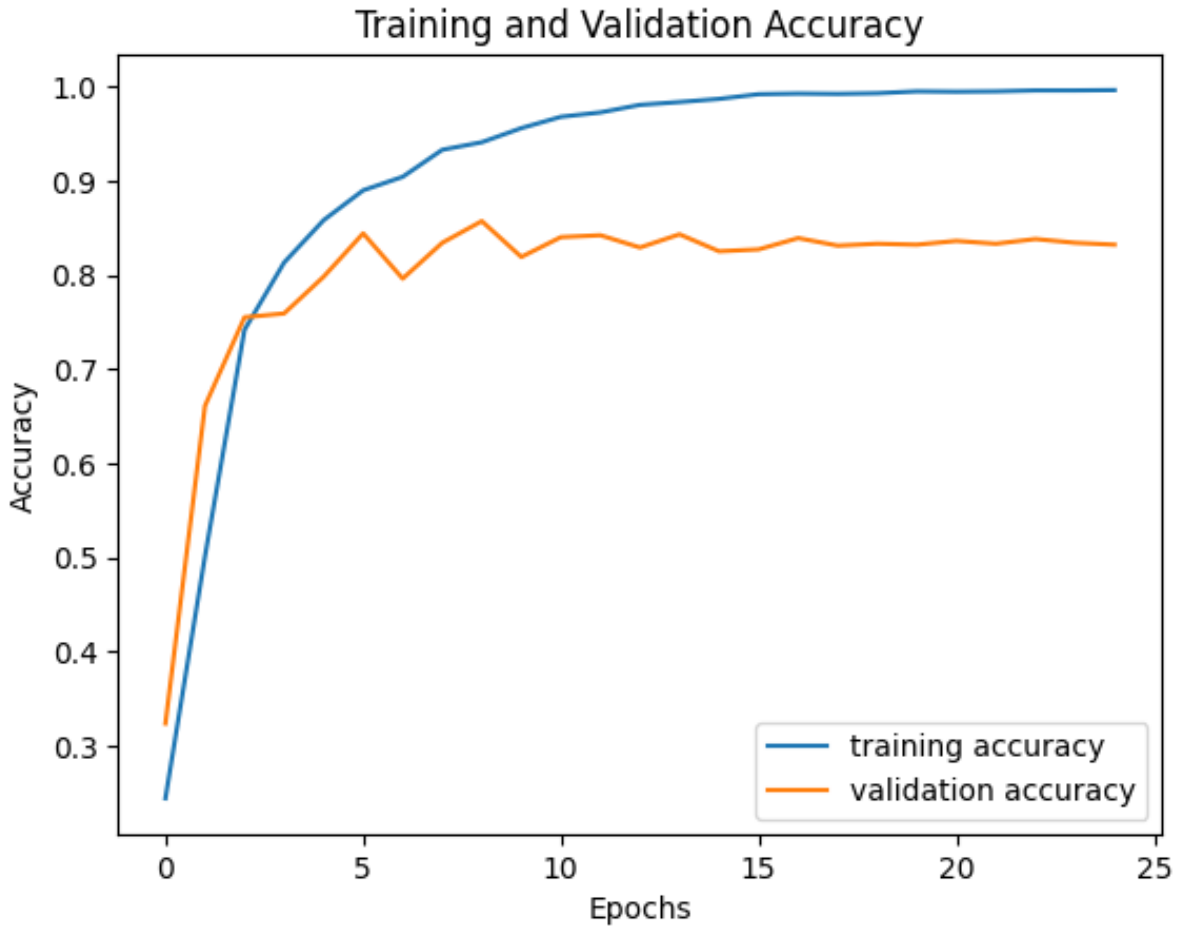

Figure 8.8: Accuracy while Finetuning a DistilBERT Model with the Author Labels of the C50 Dataset

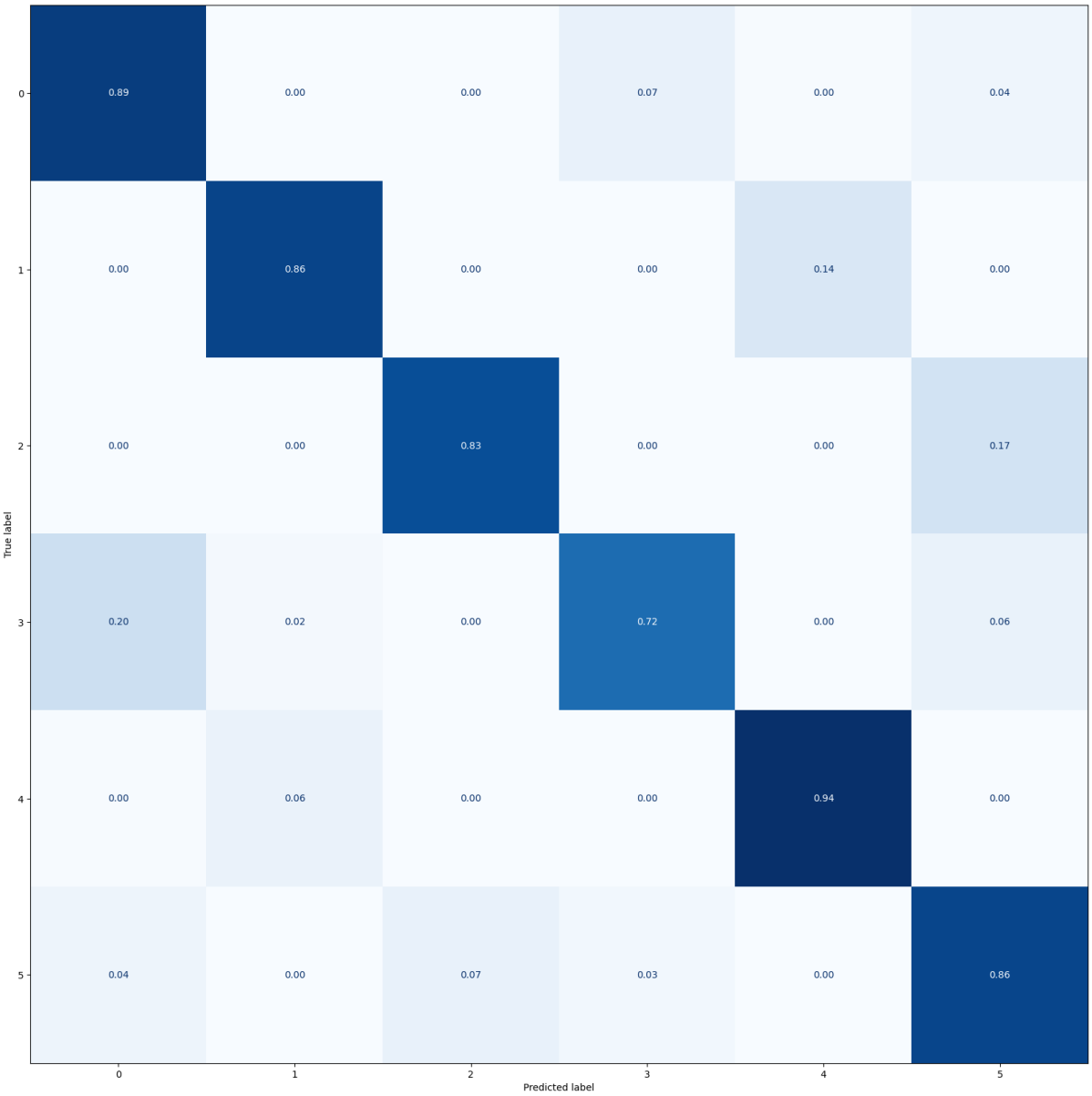

Figure 8.9: Normalized Confusion Matrix of the DistilBERT Model Finetuned with the Word Embedding Cluster Labels

Label	Accuracy
Author	71%
TF-IDF Cluster	81%
Word2Vec Cluster	85%

Table 8.1: Accuracies of Each Finetuned Model on the Test Sets

Original	Adversarial
the british government on thursday referred a proposed merger of key crosschannel ferry services provided by britains pampo and swedens stena line to its main competition watchdog for investigation	the britsh government on thursday referred a proposed merger of key crosschannel ferry services provided by britains pampo and swedens stena line to its main competition watchdog for investigation
america online inc said monday it was starting a major expansion of its online network and planned to spend 250 million through the end of its fiscal year in june to build capacity and improve service	america on-line inc said monday it was starting a major expansion of its on-line network and planned to spend 250 million through the end of its fiscal year in june to build capacity and improve service

Table 8.2: Adversarial Text Created Using Our Adversarial Text Generation Algorithm

After training each model, we take their respective test sets and classify them. As shown in Table 8.1, the model finetuned with the author labels reached an accuracy of 71%. The model finetuned with the TF-IDF cluster labels reached an accuracy of 81%. The model finetuned for the word embedding clusters also reached an accuracy of 85%. The texts that were correctly classified were then taken and used in our algorithm to generate the adversarial texts. We also took the adversarial texts that were generated from our cluster classification models and used our author classification model to see how well moving from one cluster to another affects the the classification at the author level.

8.5 Results

The test set with author labels initially had 71% of texts correctly classified. When we ran those correctly classified samples through our adversarial text generator, we were able to cause the classifier model to misclassify 49% of the texts when using WordNet to retrieve synonyms and 56% of them when using pretrained word vectors. We then took these misclassified texts and evaluated them using cosine similarity and BLEU score. For the texts generated with the synonyms from WordNet, 98% of the misclassified texts had a cosine similarity score of at

	WordNet	**FastText**
Misclassified Overall	49%	56%
Cosine Sim (≥ 0.9)	98%	99%
BLEU (≥ 0.7)	57%	69%
Cosine Sim & BLEU	57%	69%
Cosine Sim & BLEU Overall	28%	38%

Table 8.3: Misclassification Rates for the Adversarial Text with the Author Classification Model

least 0.9 and 57% of the misclassified texts had a BLEU score of at least 0.7. 41% of the texts had both a cosine similarity score of at least 0.9 and BLEU score of at least 0.7. For the texts generated with the synonyms from the pretrained word vectors, 99% of the misclassified texts had a cosine similarity score of at least 0.9 and 69% of the misclassified texts had a BLEU score of at least 0.7. 49% of the texts had both a cosine similarity score of at least 0.9 and BLEU score of at least 0.7. See Table 8.3.

The test set that was labeled with respect to the cluster of the TF-IDF representation of each text belonged to initially had 83% of texts correctly classified. When we ran those correctly classified samples through our adversarial text generator, we were able to cause the classifier model to misclassify 58% of the texts when using WordNet to retrieve synonyms and 57% of them when using pretrained word vectors. We then took these misclassified texts and evaluated them using cosine similarity and BLEU score. For the texts generated with the synonyms from WordNet, 99% of the misclassified texts had a cosine similarity score of at least 0.9 and 40% of the misclassified texts had a BLEU score of at least 0.7. 41% of the misclassified texts had both a cosine similarity score of at least 0.9 and BLEU score of at least 0.7. For the texts generated with the synonyms from the pretrained word vectors, 100% of the misclassified texts had a cosine similarity score of at least 0.9 and 40% of the misclassified texts had a BLEU score of at least 0.7. 40% of the texts had both a cosine similarity score of at least 0.9 and BLEU score of at least 0.7. See Table 8.4.

The test set that was labeled with respect to the cluster of the word embedding representation of each text belonged to initially had 85% of texts correctly classified. When we ran those correctly classified samples through our adversarial text generator, we were able to cause the

	WordNet	FastText
Misclassified Overall	58%	57%
Cosine Sim (≥ 0.9)	99%	100%
BLEU (≥ 0.7)	40%	40%
Cosine Sim & BLEU	40%	40%
Cosine Sim & BLEU Overall	23%	23%

Table 8.4: Misclassification Rates for the Adversarial Text with the TF-IDF Cluster Classification Model

	WordNet	FastText
Misclassified Overall	34%	33%
Cosine Sim (≥ 0.9)	99%	100%
BLEU (≥ 0.7)	75%	76%
Cosine Sim & BLEU	74%	76%
Cosine Sim & BLEU Overall	25%	25%

Table 8.5: Misclassification Rates for the Adversarial Text with the Word Embedding Cluster Classification Model

classifier model to misclassify 34% of the texts when using WordNet to retrieve synonyms and 33% of them when using pretrained word vectors. We then took these misclassified texts and evaluated them using cosine similarity and BLEU score. For the texts generated with the synonyms from WordNet, 99% of the misclassified texts had a cosine similarity score of at least 0.9 and 75% of the misclassified texts had a BLEU score of at least 0.7. 74% of the misclassified texts had both a cosine similarity score of at least 0.9 and BLEU score of at least 0.7. For the texts generated with the synonyms from the pretrained word vectors, 100% of the misclassified texts had a cosine similarity score of at least 0.9 and 76% of the misclassified texts had a BLEU score of at least 0.7. 76% of the misclassified texts had both a cosine similarity score of at least 0.9 and BLEU score of at least 0.7. See Table 8.5.

Label	Accuracy	Drop
Author	44%	-27%
TF-IDF Cluster	62%	-19%
Word2Vec Cluster	64%	-21%

Table 8.6: Accuracy of Test Set After Applying the Adversarial Text Generation Algorithm

Label	Original	Adversarial
TF-IDF (WordNet)	81%	78%
TF-IDF (FastText)	85%	84%
Word2Vec (WordNet)	85%	77%
Word2Vec (FastText)	83%	77%

Table 8.7: Adversarial Texts Generated from the Cluster Models and then Classified Using the Author Classification Model

Table 8.6 summarizes the best results for each model. These results come from applying the adversarial text generation algorithm to each test set using the synonyms obtained with the FastText pretrained word embeddings. We also are only considering the samples that had the cosine similarity score of at least 0.9 and the BLEU score of at least 0.7. The model trained on the word embedding clusters had the most significant change in accuracy when applying the adversarial text generation algorithm. It should be noted that the same pretrained word embedding model was used when creating the clusters and selecting synonyms within the algorithm. The Author classification had the largest drop in accuracy at 27%. This could be due to the fact that there are 50 author labels, whereas the cluster models only had five or six labels.

Table 8.7 shows the results from taking the adversarial texts that were created from the cluster models and then using their original author as a label to test how well moving from one cluster to another aides in obfuscating the original author. The two sets of adversarial texts that were generated from the word embedding based clusters had the largest drops in accuracy with an 8% drop for the adversarial text generated using WordNet and 6% for the adversarial text generated using the FastText embeddings.

8.6 Summary and Future Work

In this chapter, we finetuned DistilBERT models for the task of text classification. One model was trained using author labels, while the other two models were trained using labels that were based on the texts being clustered in two different ways: TF-IDF and word embedding representation. Once these models were trained, interpretations of their predictions were used in an algorithm to generate adversarial texts. Utilizing the interpretations helped us drop the

Original	Adversarial
apple computer inc said tuesday it will consolidate its independent marketing and development units into fewer groups to cut costs and to concentrate on selling computers in key markets	pears computer inc said tuesday it will consolidate its independent marketing and development units into fewer groups to cut costs and to concentrate on selling computers in key markets
the chinese authorities acted decisively on friday to stop extraordinary price movements on the countrys two stock markets by imposing a limit of 10 percent in the movement of any stock on one day	the non-Chinese authorites acted decisively on friday to stop extraordinary price movements on the countrys two stock markets by imposing a limit of 10 percent in the movement of any stock on one day

Table 8.8: Adversarial Text Generated With Poor Substitutions

accuracies of all three models by as much as 27%. When we classified the adversarial texts generated using the interpretations from the cluster classification models by predicting their original author with our author classification model, we were able drop the accuracy of the author classification model by as much as 8%. The adversarial texts generated from the word embedding clusters contributed the largest drops in accuracy.

For future work, more experimentation needs to be done with different word embedding models and other datasets. We plan on observing how different datasets can be clustered together and subsequently classified. Being able to combine datasets for clustering would be very useful when it comes to data gathering for adversarial authorship. It is difficult to find individual authors with a sufficient amount of text samples. While looking for a dataset to finetune our DistilBERT models, we had trouble finding one with, both, enough authors and an adequate amount of texts per author. Two datasets we tried to finetune, a blog dataset and a tweet dataset, failed to achieve over 40% accuracy. We also plan on finetuning other sequence classification models to see how various parameters affect the classification. There also needs to be work done to improve the text generated when using the word embeddings to retrieve the synonyms. Because the nearest word embeddings do not just include synonyms but also include antonyms, sometimes the adversarial text that is generated loses it's meaning (see Figure 8.8). Every potential substitution candidate should be checked for contextual and semantic similarity to ensure the meaning of the text is not altered.

Chapter 9

Summary and Future Work

9.1 Summary

In this dissertation, we begin by presenting a method for creating adversarial text in an effort to conceal an author's identity for online privacy. We demonstrate the efficacy of our adversarial authorship method that utilizes machine translation in a hill climbing algorithm. We tested this method, along with other successful AMTs, against high performing authorship attribution algorithms. Our method was able to drop the accuracy of the authorship attribution algorithms by as much as 20%. We also demonstrated that when pairing our tool with an algorithm that combines the AMTs, and automates the creation of adversarial text, we were able to drop the accuracy of the authorship attribution algorithms by as much as 52%. We proceeded by incorporating the authorship attribution algorithms within our adversarial authorship methods (AuthorCAAT, AIM-IT, and the hybrid) to demonstrate that performance improves within a partially observable environment. Additionally, we explore the challenges of operating AuthorCAAT before designing a framework for adversarial authorship, JohariMAA, that allows for a uniform user experience regardless of the adversarial authorship approach being implemented. Finally, we repackage the functionality of AuthorCAAT within the implementation of our framework while being mindful of AuthorCAAT's shortcomings. We then make a move to improve the author selection and prediction aspects of the JohariMAA framework and utilize the interpretability of deep learning models to implement an algorithm for generating adversarial text.

9.2 Future Work

JohariMAA allows us to continue exploring different methods of adversarial authorship through its framework. The framework gives us the ability to explore approaches for mutation methods beyond a hill climbing configuration and machine translation in general.

9.2.1 Author Selection

Future work should focus on improving the ability to target a specific author rather than just moving away from the original author. In AuthorCAAT, targeting an author did not necessarily lead to the adversarial text being identified as the specified author target. There should also be work done to examine a method for automating the selection of an author target, as a user-selected author target may not be the best option.

Future work should also examine how using groups of authors, or clusters, as the potential author targets can affect the detection of an author by authorship attribution algorithms. This work should explore clustering authors from various datasets and document types (e.g. blogs, emails, etc.).

9.2.2 Mutation Methods

Future work should explore more approaches to mutating text. This work should include methods that utilize word embeddings. When developing mutation approaches, there should be a focus on preserving semantics in an effort to minimize the user's need for editing the text. This can be explored using WordNet and HowNet. Mutation methods that utilize language models should also be explored. In recent years, the transformer, a deep learning model that uses a self-attention mechanism within a encoder-decoder architecture, has become popular within the field of natural language processing [60]. Pre-trained transformer models like BERT (Bidirectional Encoder Representations with Transformers) and GPT-n (Generative Pre-trained Transformer) are trained on large datasets and serve as base models that are fine-tuned for specific tasks [61][62].

9.2.3 Author Identification

Future work should also focus on strengthening the underlying author identification mechanism. This will improve the anonymity of the adversarial text by making it more difficult for an author to be undetected. Having a higher threshold for being anonymized could help the adversarial text evade more authorship attribution algorithms. Ensemble learning should be considered when constructing a more robust author identification approach. State of the art authorship attribution algorithms should be considered when exploring approaches for author identification.

References

[1] Meredith, Sam. “Facebook-Cambridge Analytica: A Timeline of the Data Hijacking Scandal.” CNBC, 10 Apr. 2018, www.cnbc.com/2018/04/10/facebook-cambridge-analytica-a-timeline-of-the-data-hijacking-scandal.html.

[2] Chan, Rosalie. “The Cambridge Analytica whistleblower explains how the firm used Facebook data to sway elections.” Business Insider, 5 Oct. 2019, https://www.businessinsider.com/cambridge-analytica-whistleblower-christopher-wylie-facebook-data-2019-10.

[3] T. Neal, K. Sundararajan, A. Fatima, Y. Yan, Y. Xiang, and D. Woodard, “Surveying Stylometry techniques and applications,” ACM Computing Surveys. 2017.

[4] S. Elmanarelbouanani and I. Kassou, “Authorship Analysis Studies: A Survey,” Int. J. Comput. Appl., 2014.

[5] A. W. E. McDonald, S. Afroz, A. Caliskan, A. Stolerman, and R. Greenstadt, “Use fewer instances of the letter ‘i’: Toward writing style anonymization,” 2012.

[6] Day, S., Williams, H., Shelton, J., Dozier, G. (2016) Towards the Development of a Cyber Analysis Advisement Tool (CAAT) for Mitigating De-Anonymization Attacks. Modern Artificial Intelligence and Cognitive Science Conference col. 1584, no. 10, pp 41-46.

[7] Day, S., Brown, J., Thomas, Z., Gregory, I., Bass, L., and Dozier, G. (2016, August) Adversarial Authorship, Author Webs, and Entropy-Based Evolutionary Clustering. International Conference on Computer Communications and Networks.

[8] A. Caliskan and R. Greenstadt, "Translate once, translate twice, translate thrice and attribute: Identifying authors and machine translation tools in translated text," in Proceedings of the 2012 IEEE Sixth International Conference on Semantic Computing, ser. ICSC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 121–125. [Online]. Available: https://doi.org/10.1109/ICSC.2012.46

[9] Brennan, M. R., Greenstadt, R. (2009, April). Practical attacks against authorship recognition techniques. In Twenty-First IAAI Conference.

[10] M. Brennan, S. Afroz, and R. Greenstadt, "Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity," ACM Trans. Inf. Syst. Secur., vol. 15, no. 3, pp. 12:1–12:22, Nov. 2012.[Online]. Available: http://doi.acm.org/10.1145/2382448.2382450

[11] A. W. E. McDonald, J. Ulman, M. Barrowclift, and R. Greenstadt, "Anonymouth Revamped: Getting Closer to Stylometric Anonymity," 2012.

[12] H. Takagi, "Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation," Proc. IEEE, 2001.

[13] G. Dozier, "Evolving robot behavior via interactive evolutionary computation: From real-world to simulation," in Proceedings of the ACM Symposium on Applied Computing, 2001.

[14] T. Back, U. Hammel, and H. P. Schwefel, "Evolutionary computation: Comments on the history and current state," IEEE Trans. Evol. Comput, 1997.

[15] W. M. Spears, K. A. De Jong, T. Bäick, D. B. Fogel, and H. De Garis, "An overview of evolutionary computation," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1993.

[16] K. De Jong and W. Spears, "On the State of Evolutionary Computation," Proc. of the Fifth Int. Conf. on Genetic Algorithms, 1993.

[17] N. Mack, J. Bowers, H. Williams, G. Dozier, and J. Shelton, "The Best Way to a Strong Defense is a Strong Offense: Mitigating Deanonymization Attacks via Iterative Language Translation," Int. J. Mach. Learn. Comput., 2015.

[18] S. Day, J. Brown, Z. Thomas, I. Gregory, L. Bass, and G. Dozier, "Adversarial Authorship, AuthorWebs, and Entropy-Based Evolutionary Clustering," in 2016 25th International Conference on Computer Communication and Networks (ICCCN), 2016, pp. 1–6.

[19] L. D. Davis and M. Mitchell, Handbook of Genetic Algorithms. 1991.

[20] G. Dozier et al., "GEFeS: Genetic evolutionary feature selection for periocular biometric recognition," in IEEE SSCI 2011 - Symposium Series on Computational Intelligence - CIBIM 2011: 2011 IEEE Workshop on Computational Intelligence in Biometrics and Identity Management, 2011.

[21] H. C. Williams, J. N. Carter, W. L. Campbell, K. Roy, and G. V. Dozier, "Genetic Evolutionary Feature Selection for Author Identification of HTML Associated with Malware," Int. J. Mach. Learn. Comput., 2014.

[22] Y. R. Tausczik and J. W. Pennebaker, "The psychological meaning of words: LIWC and computerized text analysis methods," Journal of Language and Social Psychology. 2010.

[23] A. McCallum, "MALLET: A Machine Learning for Language Toolkit," http://Mallet.Cs.Umass.Edu, 2002.

[24] Ceaparu, I., Lazar, J., Bessiere, K., Robinson, J., and Shneiderman, B. 2004. Determining causes and severity of end-user frustration. International Journal of Human Computer Interaction, 17, 3 (2004), 333-356 .

[25] Lazar, J., Jones, A., and Shneiderman, B., Workplace user frustration with computers: An exploratory investigation of the causes and severity. Behaviour Information Technology, 25, 3 (2006), 239-251

[26] Nielsen, J. (2012). Usability 101: Introduction to Usability. Retrieved from http://www.nngroup.com/articles/usability-101-introduction-to-usability/

[27] Teahan, William J., and David J. Harper. "Using compression-based language models for text categorization." In Language modeling for information retrieval, pp. 141-165. Springer, Dordrecht, 2003.

[28] Koppel, M., Schler, J. and Argamon, S., Authorship attribution in the wild. Language Resources and Evaluation, 45(1), 2011.

[29] Oh, C., Song, J., Choi, J., Kim, S., Lee, S. and Suh, B., I Lead, You Help but Only with Enough Details: Understanding User Experience of Co-Creation with Artificial Intelligence. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. ACM, 2018.

[30] Roy, Q., Zhang, F., and Vogel, D., "Automation accuracy is good, but high controllability may be better." In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1-8. 2019.

[31] J. Herz and A. Bellaachia. "The authorship of audacity: Data mining and stylometric analysis of Barack Obama speeches." In Proceedings of the International Conference on Data Mining (DMIN'14). 1. 2014

[32] S. Afroz, A. C. Islam, A. Stolerman, R. Greenstadt, and D. McCoy, "Doppelg anger finder: Taking stylometry to the underground," in Proceedings of the 2014 IEEE Symposium on Security and Privacy, ser. SP'14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 212–226. [Online]. Available: https://doi.org/10.1109/SP.2014.21

[33] J. Gaston et al., "Authorship Attribution vs. Adversarial Authorship from a LIWC and Sentiment Analysis Perspective," in Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018, 2019.

[34] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," Language Resources and Evaluation. 2005.

[35] E. Riloff, J. Wiebe, and T. Wilson, "Learning subjective nouns using extraction pattern bootstrapping," 2007.

[36] E. Riloff and J. Wiebe, “Learning extraction patterns for subjective expressions,” 2007.

[37] T. Wilson, J. Wiebe, and P. Hoffmann, “Recognizing contextual polarity in phrase-Level sentiment analysis,” in Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-2005), 2005.

[38] J. Gaston et al., “Authorship Attribution via Evolutionary Hybridization of Sentiment Analysis, LIWC, and Topic Modeling Features,” in Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018, 2019.

[39] J. Allred, S. Packer, G. Dozier, S. Aykent, A. Richardson, and M. King, “Towards a Human-AI Hybrid for Adversarial Authorship,” in IEEE Southeast Con, 2020, to be published.

[40] D. Castro, R. Ortega, and R. Muñoz, “Author Masking by Sentence Transformation,” 2017.

[41] T. Mihaylova, G. Karadzhov, P. Nakov, Y. Kiprov, G. Georgiev, and I. Koychev, “SU@ PAN’2016: Author Obfuscation—Notebook for PAN at CLEF 2016,” Conf. Labs Eval. Forum, 2016.

[42] M. Rahgouy, H. B. Giglou, T. Rahgooy, H. Zeynali, S. Khayat, and M. Rasouli, “Author Masking Directed by Author’s Style,” pp. 1–6, 2018.

[43] V. Keselj, F. Peng, N. Cercone, and C. Thomas, “N-gram-based Author Profiles for Authorship Attribution,” 2003.

[44] P. Shrestha, S. Sierra, F. A. González, P. Rosso, M. Montes-Y-Gómez, and T. Solorio, “Convolutional neural networks for authorship attribution of short texts,” in 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference, 2017.

[45] Engelbrecht, A.P., Computational intelligence: an introduction. John Wiley Sons. 2007

[46] Biles, J. ”GenJam: A genetic algorithm for generating jazz solos.” ICMC. Vol. 94. 1994.

[47] Eiben, A. E., Smith, J. E. Introduction to evolutionary computing (Vol. 53, p. 18). Berlin: springer. 2003.

[48] M. Narayanan et al., “Adversarial Authorship, Sentiment Analysis, and the AuthorWeb Zoo,” in Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018, 2019.

[49] J. Gaston et al., “Authorship Attribution vs. Adversarial Authorship from a LIWC and Sentiment Analysis Perspective,” in Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018, 2019.

[50] J. Gaston et al., “Authorship Attribution via Evolutionary Hybridization of Sentiment Analysis, LIWC, and Topic Modeling Features,” in Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018, 2019.

[51] Rangel F, Rosso P. Overview of the 7th author profiling task at PAN 2019: bots and gender profiling in twitter. In Working Notes Papers of the CLEF 2019 Evaluation Labs Volume 2380 of CEUR Workshop 2019.

[52] C. Faust, G. Dozier, J. Xu, and M. C. King, “Adversarial authorship, interactive evolutionary hill-climbing, and author CAAT-III,” in 2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings, 2018.

[53] Brooke, John, others: Sus-a quick and dirty usability scale. Usability evaluation in industry 189(194), 4–7 (1996)

[54] M. Potthast, F. Schremmer, M. Hagen, and B. Stein, “Overview of the author obfuscation task at PAN 2018: A new approach to measuring safety,” CEUR Workshop Proc., vol. 2125, 2018.

[55] Darren Shou, “The Next Big Privacy Hurdle? Teaching AI to Forget,” Wired, 2019. [Online]. Available: https://www.wired.com/story/the-next-big-privacy-hurdle-teaching-ai-to-forget/. [Accessed: 05-Dec-2019].

[56] M. H. Jarrahi, “Artificial intelligence and the future of work: Human-AI symbiosis in organizational decision making,” Bus. Horiz., vol. 61, no. 4, pp. 577–586, Jul. 2018.

[57] Fast, E., Chen, B., Bernstein, M. S. (2016, May). Empath: Understanding topic signals in large-scale text. In Proceedings of the 2016 CHI conference on human factors in computing systems (pp. 4647-4657).

[58] Tang, Y., Tran, C., Li, X., Chen, P. J., Goyal, N., Chaudhary, V., ... Fan, A. (2020). Multilingual translation with extensible multilingual pretraining and finetuning. arXiv preprint arXiv:2008.00401.

[59] Bryan Klimt and Yiming Yang. 2004. The Enron Corpus: A New Dataset for Email Classification Research. Springer, Berlin, 217–226. DOI:http://dx.doi.org/10.1007/978-3-540-30115-8_22

[60] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems, 30.

[61] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[62] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. and Agarwal, S., 2020. Language models are few-shot learners. Advances in neural information processing systems, 33, pp.1877-1901

[63] Houvardas, John, Stamatatos, Efstathios. (2006). Houvardas06 Author Identification: C50-Attribution [Data set]. In CLEF 2015 Labs and Workshops, Notebook Papers. Conference title: PAN at Conference and Labs of the Evaluation Forum 2015 (PAN at CLEF 2015). Zenodo. https://doi.org/10.5281/zenodo.3759068

[64] Radim Řehůřek and Petr Sojka. ”Software Framework for Topic Modelling with Large Corpora.” In Proceedings of the LREC 2010 Workshop on New Challenges for NLP

Frameworks, pages 45-50, Valletta, Malta, May 22, 2010. ELRA. http://is.muni.cz/publication/884893/en.

[65] Princeton University "About WordNet." WordNet. Princeton University. 2010.

[66] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin. Advances in Pre-Training Distributed Word Representations

[67] Sanh, V., Debut, L., Chaumond, J., Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

[68] Pierse, C.D. (2021). Transformers-Interpret [Computer software]. GitHub. https://github.com/cdpierse/transformers-interpret

[69] PyTorch Captum. (n.d.). Captum.ai. Retrieved May 2, 2023, from https://captum.ai/