Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Effect of Context Vectors in the Task of Authorship Verification

Sai Kaushik Mudigonda, B.Tech

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Intelligent Systems)

Supervisor: Dr. Carl Vogel

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Sai Kaushik Mudigonda

August 19, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Sai Kaushik Mudigonda

August 19, 2022

Effect of Context Vectors in the Task of Authorship Verification

Sai Kaushik Mudigonda, Master of Science in Computer Science
University of Dublin, Trinity College, 2022

Supervisor: Dr. Carl Vogel

Authorship Verification is a growing area in the field of natural language processing and text analysis. Due to the high availability of text online there is even more research into identifying and understanding the author. Other areas where this would be helpful would be plagiarism detection, citing proper authors, authorship profiling, obtaining information about the author to be used for marketing and product usage, building adaptive intelligent systems to better suit the product to their users, and determining authors of unknown texts for authenticity or other reasons.

Context vectors help in representing word tokens based on their neighboring words and their usage, this helps in capturing useful information about the style of an author which is often lost when using other word embedding methods like Word2Vec.

Statistics from a similarity matrix generated after comparing the context vectors of tokens are used to capture stylistic features. For e.g, the style of an author who uses a particular set of vocabulary or unique words in relation to each other can be captured by these statistics.

This paper explores the amount by which the style of an author determined by context vectors can influence the task of authorship verification. Usage of context vectors has not been done before in this task. The paper proposes to use context vectors alongside other commonly used features like word n-grams, character n-grams, and others using various kinds of similarity measures and configurations.

It was found that using context vectors helps in increasing the similarity predictions of document pairs. The top n Frequent tokens perform the best followed by singleton tokens and using every token. Cosine similarity has also been found to perform better than euclidean distance and The imposter strategy also gives lower absolute error than the universum strategy.

Acknowledgments

I would like to thank Dr. Carl Vogel for his constant guidance and support throughout the year as my supervisor. I would like to thank Dr. Erwan Moreau for his help with understanding the authorship verification system.

I would like to thank my parents, Rajender and Deepthi for their care and support and also my friends, Ritika, Yash, Tolga and Oliver for their motivation, help and friendship.

Finally I would like to thank Allah SWT, without whose will none of this would have happened.

Sai Kaushik Mudigonda

University of Dublin, Trinity College
August 2022

Contents

List of Figures

Nomenclature

AI	Artificial Intelligence
AV	Authorship Verification
POS	Part of Speech
SVM	Support Vector Machine
BERT	Bidirectional Encoder Representations from Transformers
IDE	Integrated Development Environment
MPNET	Masked and Permuted Pre-training for Language Understanding
T5	Text-To-Text Transfer Transformer
CNN	Convolutional Neural Network
NLP	Natural Language Processing
CBOW	Continuous Bag of Words
GloVe	Global vector for word representation
LSA	Latent Semantic Analysis
ELMo	Embeddings from Language Models
LSTM	Long short-term memory
RNN	Recurrent neural networks
GPT	Generative Pre-trained Transformer
LDA	Latent Dirichlet Allocation
OOV	Out-of-Vocabulary
VSM	Vector Space Model
IEEE	Institute of Electrical and Electronics Engineers
SVD	Singular Value Decomposition
KNN	K-Nearest Neighbors
TTR	Type-Token Ratio
TF-IDF	Term Frequency-Inverse Document Frequency
DNA	Deoxyribonucleic Acid
WEKA	Waikato Environment for Knowledge Analysis
ROC	Receiver Operating Characteristic Curve
ANOVA	Analysis of Variance
ID	Identity Document
CSV	Comma Separated Value

Chapter 1

Introduction

This chapter gives a basic overview of the paper, starting with the motivation behind pursuing this dissertation, questions to be answered from the study, and the contributions and challenges faced during the research.

With many viable areas to be developed, the task of authorship verification has seen a lot of research and study. Trying to understand an author's style is of much more use here than the content presented within the literature. Though the content gives more details about the state and personality of the author, the stylistic features in the text provide more precise information on detecting the author Juola (2012).

A lot of research and work has been done in checking whether a pair of documents have been written by the same author, especially through PAN Competitions, which are a yearly event involving digital text forensics and stylometry tasks. researchers throughout have used methods ranging from using deep metric frameworks Boenninghoff et al. (2020), SVM Classifiers Koppel et al. (2007), neural networks amongst numerous others.

This paper discusses using context vectors to get information about an author's style. Statistics generated from the similarity matrix obtained from comparing vectors of tokens will help in capturing the style of an author. For e.g., an author might use a particular vocabulary or specific words in context to one another. This is captured using context vectors. This method is different from the word embeddings obtained from Word2Vec or BERT, as these are pre-trained and contain more generic representations of words found in the literature. Even though word embeddings have been used previously to understand and obtain the features within words or their sequences Fabien et al. (2020), word embeddings based exclusively on the author's style have not been used in this task before.

This paper deals with the task of Authorship Verification. In this task, a pair of documents are analyzed to decide whether the documents are written by the same author or not. An extensive set of problems are given, with each issue consisting of a pair of records. The aim is to find whether both of them are authored by the same person.

1.1 Motivation

In recent years, much attention has been given to analyzing literature. One of the earliest and most researched areas was authorship attribution. In today's world, where issues like plagiarizing others' work, fake news, and ensuring individual safety are rampant, It is essential to identify the authors from the text written by them. It can also be helpful in other areas like identifying the author for proper credentialing, plagiarism detection Stein et al. (2011), use in intelligent adaptive applications, protection of minors, adaptive marketing Rangel et al. (2013), forensic analysis Juola (2006) and giving proper citations.

Much information can be hidden behind a text's linguistic and stylistic features. Identifying this can enable us to understand and analyze the author better. Furthermore, intelligent systems that can process and harness this information from literature can significantly help in ways like content moderation, curbing fake news, and automatic citation of authors. All these issues have made identifying an author's features even more sought after and interesting.

1.2 Objectives

The paper is aimed at answering the question,

"Effect of context vectors in the task of authorship verification".

This task would involve steps like,

1. Research into Context Vectors and word sense disambiguation
2. Familiarize with the Linux and Perl environment
3. Exploring other Word Embedding methods like Word2Vec, glove and compare them.
4. Creating an inventory of viable datasets for the task of authorship verification
5. Identifying the finer details of the datasets.
6. Implementation of context vectors generation for all the tokens in the given corpora.
7. Integrating the context vectors into the authorship system.
8. Add features generated by the context vector observation to the existing Machine learning model

9. Analyze the effect of context vector features with different configurations and perform statistical analysis to check if there are significant differences between various groups.

It also proposes to use multiple similarity methods to analyze the effect of context vectors in different configurations. An ablation study of the effect of j most frequent tokens and singleton tokens are also being performed and analyzed.

1.3 Contributions

The paper proposes implementing a new way previously unimplemented, i.e., context vectors with reference to natural text categories (e.g., Authorship).

The research created new observations (context vectors) and measurements (similarity measurements) in the authorship analytics tool. Future researchers in their research could use this.

Ablation studies of the different configurations like j most frequent occurring tokens and haptic tokens present in each sub-category of the text have also been performed. These features are trained along with other commonly used features like word n-grams, and character n-grams, amongst others. Finally, they are compared with features without context vectors to understand and get a qualitative estimate of their effect on natural text categories.

Statistical analysis is performed to understand the relationship between different configurations, strategies and similarity measures.

1.4 Challenges

The greatest challenge posed during the research period was exploring and using the Perl and Linux environments. The authorship analytics tool was also very complicated for someone new to the Perl language. Integration of context vectors involved understanding the architecture of the system.

The machine learning library, Weka, which was used, was also quite limited in its capacity compared to the trending machine learning and analytics libraries present in python. Furthermore, the Perl language also has a limited library collection, leading to implementing many features from scratch.

Another major challenge was the lack of a proper IDE for the Perl language. Debugging is also a difficult task here. Code readability is also very low compared to other languages, which would generally lead new users to take a while to get around.

1.5 Dissertation Overview

1. **Chapter 1 - Introduction**

   This chapter discusses the motivation behind the research and the research problem. It also gives a brief overview of the contributions and the difficulties faced during the dissertation.

2. **Chapter 2 - Background and Literature Review**

   This chapter discusses the Background and previous work done in the word embeddings field and the authorship verification task. It also deals with different strategies and methods used by previous researchers in the field of authorship verification, especially in PAN Competitions.

3. **Chapter 3 - Methodology**

   This chapter discusses the intricate details of the dataset selected and the parameters for its selection. It will also discuss a global corpus with which the uncategorized corpora were compared to calculate the cosine similarities. A comprehensive description and explanation of the methods and algorithms used to analyze the research output are also given. The Different types of similarity methods used and their explanation is also presented in this chapter. This chapter will be the basis of how the output is generated and evaluated. The evaluation details will be discussed in detail in the next chapter.

4. **Chapter 4 - Evaluation**

   This chapter talks about the various evaluation methods used to evaluate the effect of context vectors and understand the relationships and how different configurations effect each other. The chapter includes the process of statistical analysis to determine the relationships and effects.

5. **Chapter 5 - Conclusion**

   This chapter summarizes the implementation and evaluation and the results obtained from the research done. It also mentions ways in which the research can be taken forwards in the form of future works.

Chapter 2

Background and Literature Review

This chapter provides the context for my study, a review of related material and from earlier publications. The different types of features and observations used for the machine learning models and methods are discussed. The chapter also talks about current work in the task of AV and context vectors. Summaries of previous PAN Competitions related to authorship verification are also reviewed.

2.1 Authorship Verification

Authorship Verification is still a hot issue in computational text analysis research, with several applications in modern culture and business. In Authorship Verification, a pair of documents are compared and analyzed to check if they have been written by the same author or not Koppel and Winter (2014b). The Authorship verification application includes instances like plagiarism detection, unidentified notes, citing proper authors, forensic analysis Juola (2006).

Konstantinou et al. (2022) compared word embeddings obtained from pre-trained models like BERT, MPNET, and T5 on the text and selected the most efficient one to verify authorship.

Galicia et al. (2022) model the documents using a graph representation, from which a graph neural network extracts important information. Three methods are described for representing text as graphs based on the co-occurrence of POS labels. A Siamese Network architecture consisting of graph convolutional networks, pooling layers, and classification layers is used to check for authorship similarity.

The work by Crespo-Sanchez et al. (2022) entails examining the important elements of the documents, such as the lexical and syntactical features, which reflects how the author combines the various words in the vocabulary according to grammatical rules; and the semantic content of the papers. This study proposes a neural network-based approach to content spectral analysis.

Huang et al. (2022) proposes a method to derive the relationship between text pairings by segmenting and combining the texts in a certain sequence. A trained model is used to extract the characteristics of text pairings.

In Najafi and Tavan (2022), using the T5 language model, semantic and stylometric characteristics were converted into representation vectors. A CNN neural network retrieves ngram characteristics from sequences of punctuation. The attention module is also used to extract the essential token characteristics and establish their relationships.

Ye et al. (2022) created a model that combines Bert and TextCNN characteristics, and data rearrangement technique to boost model performance is used for determining the similarity measure.

These methods show some of the research done in the task of authorship verification. More research taken up in this area will be discussed later in the chapter.

2.2 Word Embeddings

Humans have used literature to communicate, record events, and express themselves since time immemorial. This immense amount of information is now available online. This vast library of how humans have expressed and lived would help build more intelligent applications that can further help solve many problems troubling society today.

While quantitative data like figures and numbers can be easily analyzed by computers, analyzing textual data is not that simple. Computers cannot understand and analyze textual data directly. While, in some instances, AI systems can recognize patterns in text, understanding them is still not entirely achievable. Nevertheless, one method to solve this issue is through generating word embeddings. Word embeddings are a way to convert text into quantitative data so that machines can analyze them. Each word or token is represented as an array of numbers or vectors based on some technique.

Learning word embeddings from a corpus can be accomplished in numerous ways. The most common is a one-hot encoding, which assigns a unique index to each word in a lexicon made up of n distinct components. It is a vector of zeroes except for a 1 in the correct place representing a word. While learning contextual prediction, word embedding methods turn these one-hot vectors into dense representations with dimensionality much lower than the vocabularies and elements that capture the language data's latent semantics. This is done by learning context-based prediction and then applying it to the data. Better word prediction can be achieved by the acquisition of more dense representations of words.

Word Embeddings were created as a response to the limitations of the bag of words technique. Bag of words is a technique where a vector containing each token's count is maintained for each sentence. However, this method has a lot of problems, such as most of the vector representation of a sentence is filled with zeroes, and information about the relationship between tokens is not recorded. To solve these problems, word embeddings are used.

2.2.1 Word2Vec

Word2Vec is an algorithm where contextually used words often have similar vector representations. Word2Vec considers the context of the words. It also uses cosine similarity distance to determine the similarity between different words. The algorithm has two variant methods to create the vectors: CBOW and Skip-grams Mikolov et al. (2013b) Mikolov et al. (2013c).

According to Rong (2014), the CBOW model and the SG model both employ tiny neural networks to learn the mapping of words to a vector space. Rather than predicting a focus word based on its context (CBOW), the neural network attempts to predict a focus word based on its context.

An empirically tailored range of 50 to 500 is often utilized for word2vec embedding training settings, as is the length of the context window (i.e., how many words before and after the target word should be used as context for training the word embeddings, usually 5 or 10 words). Since each dimension should be able to capture some component of meaning, the embeddings must be large enough to distinguish between words when training with more dimensions.

A continuous bag of words (CBOW) defines a fixed window size around each word. Based on the surrounding context words, a one hot encoding vector is created and is given as input to a CBOW neural network. The CBOW neural network contains a fully connected hidden layer. The output tells how close the target word is to other words in the text.

While CBOW uses the surrounding words of the target word, Skip Grams uses the current word to find its context. It then tries to predict the words before and after the target word.

2.2.2 GloVe (Global vector for word representation)

The GloVe model likewise learns word embeddings using a term co-occurrence matrix rather than a task involving word prediction. A co-occurrence matrix is a n x n matrix, where n is the vocabulary size. Like Word2Vec, GloVe creates word embeddings, but GloVe uses the entire corpus to create the word embedding vectors.

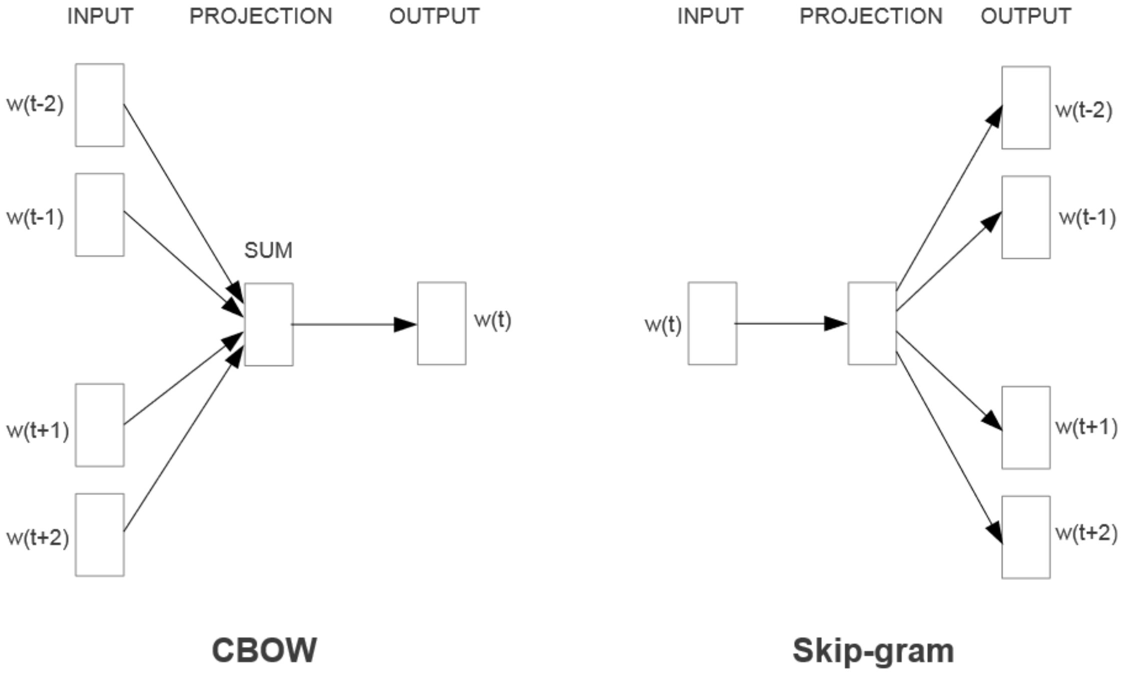


Figure 2.1: Architectures used in Word2Vec Mikolov et al. (2013a)

Each element in the matrix corresponds to the number of occurrences of the supplied vocabulary items inside a context window that traverses the whole corpus. GloVe learns vector embeddings to reduce the reconstruction error between model-predicted co-occurrence statistics and global co-occurrence statistics observed in the training corpus. The model comprises various hyperparameters that must be carefully set, like the vector embedding dimension and the context window size. It uses methods like LSA (latent semantic analysis) and the skip-gram model Agarwal (2021).

GloVe estimates word vectors conceptually similar to word2vec but using a count-based approach instead of word2vec's prediction-based methodology. In addition, because GloVe often computes statistics over more expansive context windows than word2vec, it is possible to capture long-term dependencies, albeit their order will be lost.

Word2Vec uses a window around the target word and captures its context based on its neighboring words, while, GloVe uses the entire corpus. It uses methods like LSA (latent semantic analysis) and the skip-gram model Agarwal (2021). In the case of GloVe, the number of times a word occurs in context with the target word is recorded across the whole corpus, and a matrix is created comprising all the tokens in the corpus. GloVe can be used to find similarities between words like cities, words, and others. The model uses cosine or euclidean distance methods to compute the similarity between words.

Since static word embeddings have limitations, subsequent work has attempted to build context-sensitive word representations. Deep-neural language models such as ELMo Peters et al. (2018), BERT Devlin et al. (2019), and GPT-2 are fine-tuned to build models for a wide range of downstream NLP tasks. Since they are a function of the full input phrase, their internal representations of words are referred to as contextualized word representations.

2.2.3 ELMo

ELMo is a word and character embedding that is contextualized Peters et al. (2018). For each word, ELMo considers the context of the entire phrase when assigning an embedding. The embeddings are generated by an RNN trained on a specific job. The embedding depends on the sentence's next and previous words due to the usage of a bidirectional architecture. An essential feature of ELMo's embedding is that a task-specific coefficient is applied, allowing it to be trained on one task and then adapted to an entirely different job.

2.2.4 BERT

The BERT concept of bidirectional encoder transformers is a more recent development. The model was built to extract context-sensitive properties from the input text. The way this is done is by using pre-trained deep bidirectional representations Devlin et al. (2019). ELMo, introduced by Peters et al. (2018) in 2018, was able to execute a bi-directional sweep of the text before bi-directional models. This approach utilized was, LSTMs can train on sequences going in both directions (left-to-right and right-to-left) and then embed that representation in the sequence. However, attention-based models were unable to collect contextual information.

Models like OpenAI GPT, for example, used attention to record the sequence's context when building its transformer model. However, this model could only capture the context between layers in one direction because it read the sequence from left to right. As a result, this model could not grasp the statement's whole meaning.

BERT, the most common transformer model, was developed due to the shortcomings of prior versions. This model employs a bidirectional transformer that can train the model in both the forward and backward directions. Thus, the model can accurately represent the meaning of words while they are read backward as well as forwards Devlin et al. (2019).

While a lot of these word embedding generation techniques have been used in the authorship related tasks Barlas and Stamatatos (2020), one disadvantage of using these methods is that they generate a generic representation for each token based on whatever general corpus the models were trained on. Using the output of these models as features during the implementation would include a lot of unwanted information, and that would overshadow the style that the author has (stylistic features present in the document).

2.3 Context Vectors

Context vectors used in this research help capture the context of tokens in the document. This is very helpful to understand the style of an author based on his vocabulary set or how he uses particular words in context to one another.

Trying to obtain the context from text has long been an area of interest. Sowmiya and Chandrakala (2014), modeled topics using Latent Dirichlet Allocation (LDA). After modifying the topic, the authors did an unsupervised sentiment analysis on the exact document. The tests were conducted on text documents consisting of Book reviews and Jan Lokpal bill reviews. This research showed three forms of emotions: good, negative, and neutral. The results were comparable to those of supervised sentiment analysis methods.

Using Out-of-Vocabulary (OOV) terms, Sheikh et al. (2016) focused on extracting context from audio/video input. Their context was represented with a document-level semantic vector to recover OOV terms. In addition, context extraction approaches such as Random Pro- projections, LSA, LDA, Skip-gram, CBOW, and GloVe were explored and utilized. With this strategy, they discovered that phonetic search utilizing OOV and Proper Nouns (PNs) is reliable.

Sharma and Jain (2015) employed Vector Space Model (VSM) to determine the degree of similarity between the concept vector and context vector. Google APIs were utilized to gather data from various web applications, including news websites and Twitter. The authors found their approach has a cheap computation cost since it eliminates sparse words during context extraction.

Dhande et al. (2014) have determined the document's context and compiled a vocabulary of either unigrams or bigrams. Using Apriori Algorithm, association mining was applied to these vocabularies. The examinations were conducted using non-standard document sets.

Lee and Chen (2006) introduced four novel feature selection approaches based on context similarity. Support Vector Machine was used to build the models. The testing was performed on two standard datasets, DataBCII and Reuters-21578. The efficiency of selection strategies was analyzed and compared to frequency-based procedures.

Poormasoomi et al. (2011) employed Latent Semantic Analysis (LSA) to capture the context of the Persian language in their study. A multi-document summary was created. The paper also mentions that this system beats previous multi-document systems in Persian.

Kulkarni et al. (2012) worked on context extraction, although she employed Naive Bayes and Apriori Association. The evaluations were conducted on the abstracts of 57 IEEE papers. Bhakkad et al. (2013) suggested an E-VSM approach in which text documents were clustered depending on their density. This system could rank documents.

These works show that context vectors can capture the context of the documents they are applied on. Using this method would help in capturing the unique style that almost every author has, i.e. the context in which they use their vocabulary in.

2.4 Strategies

Two strategies have been used primarily to do ablation tests with context vectors to understand if they affect the task of Authorship verification.

2.4.1 Universum Method

By determining the homogeneity of selected partial texts from the original problem dataset, Vogel et al. (2009) first proposed the Universum Inference method, which focuses on the homogeneity of play scripts. A large corpus that was first tagged into many categories according to the author of the texts is broken up into smaller pieces. Each text chunk from the original category is compared to text chunks from both different categories and the same category. χ^2 tests are used to determine similarity, while the Mann-Whitney rank ordering statistic via the Bernoulli Schema is used to determine homogeneity. Only classifications or categorizations with adequate homogeneity measurements will be kept track of.

This strategy was later adapted to fit the task of authorship verification at PAN in Moreau et al. (2015).

The confusion $Score_i$ is measured using several methods. One idea is to order the six texts according to how similar they are to one another. Texts from the same category should be the most similar, while texts from C_A and C_B should be the least similar. The text C_{mixed} decreases the differences when comparing as it has text from both the documents. This method does not perform as well as the following method, which is the Imposter Strategy.

Algorithm 1 Universum Algorithm

```
Input A pair of Documents, $A$ and $B$
Output 1 (Same Author) or 0 (Different Author)
procedure
    $i \leftarrow 1$
    while $i \neq N$ do
        Split $A$ and $B$ into 3 random parts - $A_1, A_2, A_3$ and $B_1, B_2, B_3$
        Split one of the parts in each category into two parts - $X_3 = X_{31}, X_{32}$
        Reorganize all the parts into 3 parts of equal length,
                and each containing 2 parts - $C_A = \{A_1, A_2\}; C_B = \{B_1, B_2\}$,
                and $C_{mixed} = \{A_{31} \cap B_{31}, A_{32} \cap B_{32}\}$
        Calculate the confusion of the similarities between the 6 texts; this is $Score_i$.
    end while
    $Score = aggregate\{Score_1, Score_2, ...\}$
    $i \leftarrow i + 1$
Return 0 if $Score < \Delta^*$; else 1

end procedure
```

2.4.2 Imposter Method

In the Imposter Strategy Koppel and Winter (2014a), Repeated comparisons are made between various chunks of the tested documents and other external (portions of) external documents (impostors). The same author probably writes the tested documents if the similarity between them is noticeably higher than the similarity between a tested document and an imposter. Tests are run to determine which is closer to the disputed text: the candidate author or the created impostors.

1. Generate a set of impostors $Y_1, ..., Y_m$ (as specified below).

2. Compute $score_X(Y)$ = the number of choices of feature sets (out of 100) for which $sim(X, Y) > sim(X, Y_i)$, for all $i = 1, ..., m$.

3. Repeat the above with impostors $X_1, ..., X_m$ and compute $score_Y(X)$ in an analogous manner.

4. If $average(score_X(Y), score_Y(X))$ is greater than a threshold σ^*, assign $<X, Y>$ to same-author.

The essential concerns are choosing the imposter set and how many impostors to utilize. Assuming X and Y are in English and the impostors are all in a different language, as an extreme example, we will receive a lot of false positives if we choose too few or unconvincing impostors. On the other hand, if we choose too many impostors or impostors for Y that are in the same genre as X but not Y, we would receive a lot of false negatives Koppel and Winter (2014a).

Algorithm 2 Imposter Strategy

```
    Input A pair of Documents, $A$ and $B$; Set of imposter documents. $rate\%, n, k, \delta^*$
    Output 1 (Same Author) or 0 (Different Author)
procedure
    $Score \leftarrow 0$
    for $i = 1$ to $N$ do
        Select rate % of the features at random from the whole feature pool.
        Randomly select n imposters from the imposter document collection.
        if $sim(A,B)sim(A,B) > sim(A, Imposter)sim(B, Imposter), \forall Imposters$ then
        end if
    end for
Return 0 if $Score < \Delta^*$; else 1

end procedure
```

This strategy uses cosine similarity. Potha and Stamatatos (2017a) adds an imposter strategy that uses pre-processed imposter documents. This variation is way better than the original algorithm by using only character 5-gram. However, even this algorithm uses cosine similarity.

It should be noted that the impostors' genre affects the classification's performance; if the texts are cross-genre, the performance will decline.

Both Potha and Stamatatos (2017a) and Seidman (2013) take advantage of search engines to create online imposter corpora using hand-picked word lists, downloaded top web pages, and HTML tags removed. The impostors are then cut to a length similar to the problematic texts.

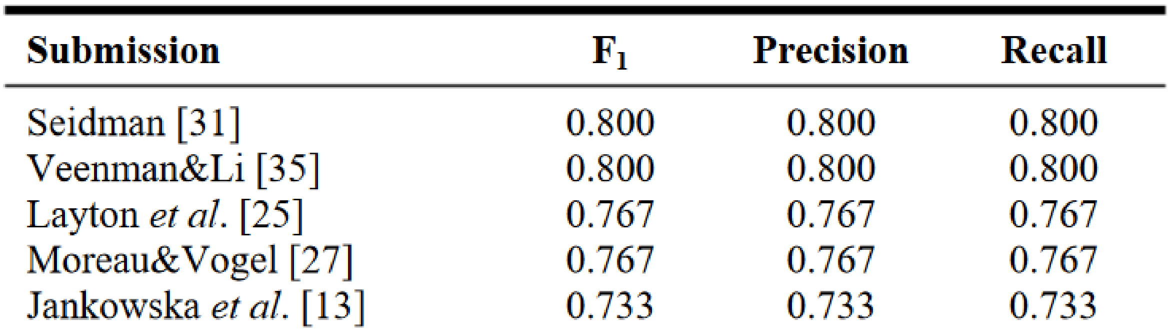
These strategies show promise while calculating the similarity between documents and have already been used in previous authorship verification tasks, with good results.

2.5 Previous PAN Developments

This section talks about the strategies and developments undertaken by researchers in previous PAN Competitions in the task of authorship verification. Since this study uses only English documents, only submissions in English are discussed here.

2.5.1 PAN 13

This was the first time the Authorship verification problem was used in PAN Competitions. Spanish, Greek, and English are all represented in the corpora Gollub et al. (2013). The data in the corpus are drawn from computer science and related textbooks that have been made available online through a repository.



Submission	F_1	Precision	Recall
Seidman [31]	0.800	0.800	0.800
Veenman&Li [35]	0.800	0.800	0.800
Layton *et al.* [25]	0.767	0.767	0.767
Moreau&Vogel [27]	0.767	0.767	0.767
Jankowska *et al.* [13]	0.733	0.733	0.733

Figure 2.2: Top 5 Submissions at PAN 13 Gollub et al. (2013)

The submissions by Potha and Stamatatos (2017b) were ranked the best in the English language. An adaptation of the imposter method was used for the authorship verification task. Features like function words, word n-grams, and character n-grams were used and were evaluated by binary, numeric, and TF-IDF techniques. His algorithm ran well with the English and the Greek corpus but did not achieve the same with the Spanish corpus.

Veenman and Li (2013) ranked second in the English language category. A compression distance measure was used for this task. The Sparse Subspace (LESS) classifier is used to classify articles with the lowest possible error rate Veenman and Tax (2005).

In Layton et al. (2013) Local N-grams are used for authorship verification. Character n-grams that appear concurrently in known and unknown texts in a pair are considered, and the pair's distance is determined. During the training procedure, the optimal distance threshold is discovered.

Moreau and Vogel (2013) used style based features in their implementation. It employs a sophisticated approach to choose the characteristics that best represent the author's work, which may involve using different statistics and distance measurements calculated from the training set. In this adaptation, a collection of characteristics is computed based on various n-gram patterns (e.g., character trigrams, POS bigrams, and others). For this n-grams pattern, each feature denotes the distance between the unknown document and the author's style.

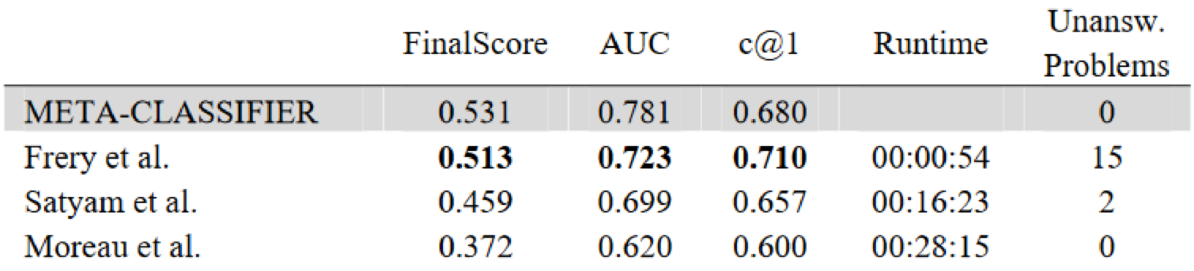
The submission by Jankowska et al. (2013) performed well in all three corpora. The Common N-Gram (CNG) dissimilarity measure was utilized in this algorithm based on the proximity-based approaches for one-class classification (akin to the k-center border algorithm).

2.5.2 PAN 14

Four languages are included in the PAN-2014 corpus: Dutch, English, Greek, and Spanish. The English corpus has 2 subcategories, essays and novels Stamatatos et al. (2014).

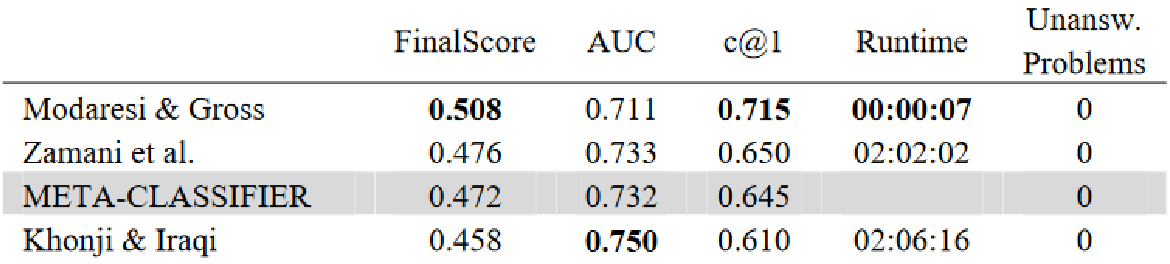
The English essays corpus is taken from the Uppsala Student English corpus; the essays are written electronically by English-as-second-language students at the university level.

The "Cthulhu Mythos" is a science fiction and horror literature genre that comprises the majority of the second category in the collection. Initially penned by H.P. Lovecraft, these fanfictions have been expanded upon by different authors. This area is thought to be more limited because of its focus on fiction and its uncommon language.



	FinalScore	AUC	c@1	Runtime	Unansw. Problems
META-CLASSIFIER	0.531	0.781	0.680		0
Frery et al.	**0.513**	**0.723**	**0.710**	00:00:54	15
Satyam et al.	0.459	0.699	0.657	00:16:23	2
Moreau et al.	0.372	0.620	0.600	00:28:15	0

Figure 2.3: Top 3 Submissions for PAN'14 English essays Stamatatos et al. (2014)



	FinalScore	AUC	c@1	Runtime	Unansw. Problems
Modaresi & Gross	**0.508**	0.711	**0.715**	**00:00:07**	0
Zamani et al.	0.476	0.733	0.650	02:02:02	0
META-CLASSIFIER	0.472	0.732	0.645		0
Khonji & Iraqi	0.458	**0.750**	0.610	02:06:16	0

Figure 2.4: Top 3 Submissions for PAN'14 English novels Stamatatos et al. (2014)

Frery et al. (2014) had the best results for English essays. The implementation used classification and regression trees on cosine similarities and correlation coefficients. In addition, features like character n-grams, phrases, word n-grams, and punctuation were used.

Saha et al. (2014) applies latent semantic analysis on a character n-grams to obtain similarities between document pairs. Parameters like n-gram lengths, SVD cutoff, and local and global weighing strategies are used here.

The implementation by Moreau et al. (2014) uses a SVM Classifier. POS, stopwords, n-grams, TTRs, and token length classes are all extracted from texts using various settings. Elements such as consistency, divergence, confidence, and distance are used to calculate features.

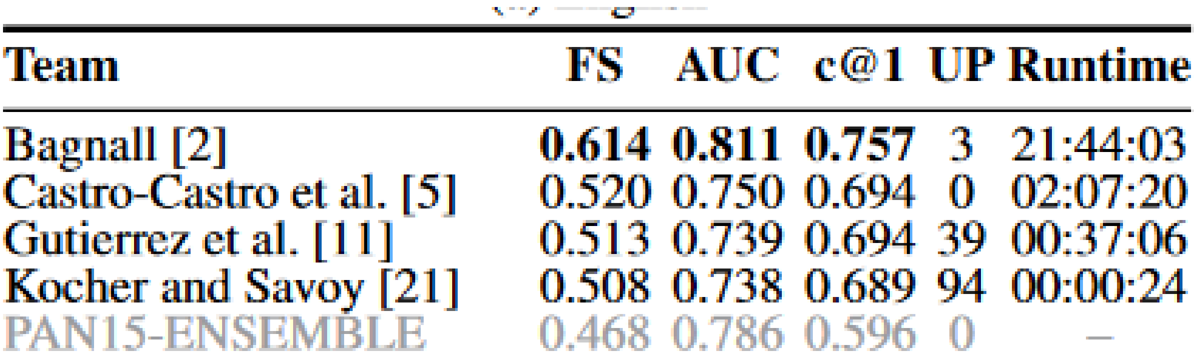
Modaresi and Gross (2014) has the best results in the English novels section. From the fuzzy clusters, membership values of the documents were calculated, whose distribution was used in solving the problem.

Zamani et al. (2014) represents features from each document as a function of the probabilistic distribution. Parameters here ranged from stop words, POS, n-grams, punctuation, and others. A KNN algorithm and dynamic feature selection are used in the ML model.

An adaptation of the Imposter strategy Seidman (2013) is used in Khonji and Iraqi (2014). A wide number of varied characteristics at the letter, word, function word, word form, and word tag levels were used, and an adaptation of the min-max similarity was implemented.

2.5.3 PAN 15

For the PAN 15 authorship verification task Stamatatos et al. (2015), a new corpus was built, which was similar in size to that of the PAN'14. Similarly, English, Spanish, Dutch and Greek languages were used for building it. In the English Corpus, only 1 available document was given for every problem, and the corpus was prepared from dialog lines from plays.



Team	FS	AUC	c@1	UP	Runtime
Bagnall [2]	**0.614**	**0.811**	**0.757**	3	21:44:03
Castro-Castro et al. [5]	0.520	0.750	0.694	0	02:07:20
Gutierrez et al. [11]	0.513	0.739	0.694	39	00:37:06
Kocher and Savoy [21]	0.508	0.738	0.689	94	00:00:24
PAN15-ENSEMBLE	0.468	0.786	0.596	0	–

Figure 2.5: Top PAN'15 Submissions in the English Category Stamatatos et al. (2015)

Bagnall (2016) used a Recurrent Neural Network to model different authors at the same time. Each output from the RNN represented an author. All the outputs depended on a shared recurrent layer which prevented overfitting even if there was a small dataset.

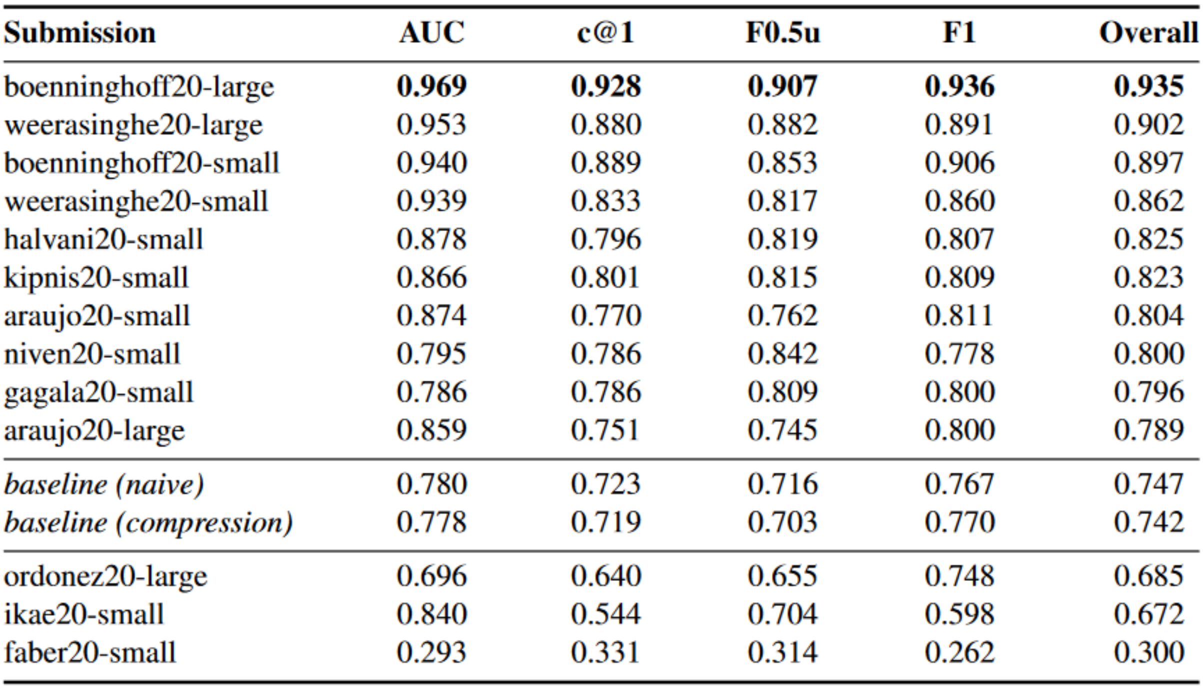
In Castro-Castro et al. (2015), the average min-max similarity of an unknown text is compared with all an author's texts. It is only when a text's average resemblance to other works written by the author surpasses the average of all those works that a text is regarded to have been written by the author and only then is a text of unknown authorship deemed to have been written by the author. Character, lexical, and syntactical features are used for computing the similarity here.

Hernández et al. (2015) uses an adaptation of the Imposter strategy where an aggregation function based on the Homotopy-based Classification procedure is used. A disputed document is generated using a dictionary of known and random imposter documents from a particular author. One can verify the authorship from the number of known documents in the disputed document.

2.5.4 PAN 20

Since the Dataset used for this study was taken from PAN'20. A summary of all the submissions is discussed rather than the top submissions.

The Dataset used in PAN 2020 Kestemont et al. (2021b) is the one used for this study. The Dataset is comprised of fan fiction scraped from fanfiction.net fac. The corpus has a small and a large dataset, and it was built on top of the corpus created by Bischoff et al. (2020).



Submission	AUC	c@1	F0.5u	F1	Overall
boenninghoff20-large	**0.969**	**0.928**	**0.907**	**0.936**	**0.935**
weerasinghe20-large	0.953	0.880	0.882	0.891	0.902
boenninghoff20-small	0.940	0.889	0.853	0.906	0.897
weerasinghe20-small	0.939	0.833	0.817	0.860	0.862
halvani20-small	0.878	0.796	0.819	0.807	0.825
kipnis20-small	0.866	0.801	0.815	0.809	0.823
araujo20-small	0.874	0.770	0.762	0.811	0.804
niven20-small	0.795	0.786	0.842	0.778	0.800
gagala20-small	0.786	0.786	0.809	0.800	0.796
araujo20-large	0.859	0.751	0.745	0.800	0.789
baseline (naive)	0.780	0.723	0.716	0.767	0.747
baseline (compression)	0.778	0.719	0.703	0.770	0.742
ordonez20-large	0.696	0.640	0.655	0.748	0.685
ikae20-small	0.840	0.544	0.704	0.598	0.672
faber20-small	0.293	0.331	0.314	0.262	0.300

Figure 2.6: Performance of all PAN'20 Submissions Kestemont et al. (2021b)

Boenninghoff et al. (2020) created a system that contains a probabilistic layer at the top to calculate the Bayes factor scoring in the learned metric space and Deep metric learning at the bottom that is aimed at learning a pseudo metric that transfers a text of variable length onto a fixed-size feature vector. This system scored well on both the corpora.

Weerasinghe et al. (2021a) employed stylometric characteristics that were taken from the documents and the absolute difference between the feature vectors. A Neural Network-based model and a Logistic Regression model were built for the large and the small datasets, respectively.

Halvani et al. (2020) used topic-agnostic features that quantified the author's style in its classification decision. Features like punctuation n-grams, token n-grams, sentence and clause starters, sentence endings, and masked token n-grams were used. The model used here consists of m distance-based classifiers, each of which tries to accept or reject an unknown document.

Kipnis (2020) uses a binomial allocation model of words between the texts to generate word-by-word p-values and combine these p-values into a single test statistic using HC (High Criticism). Large HC values show that the authors of the two papers are not the same.

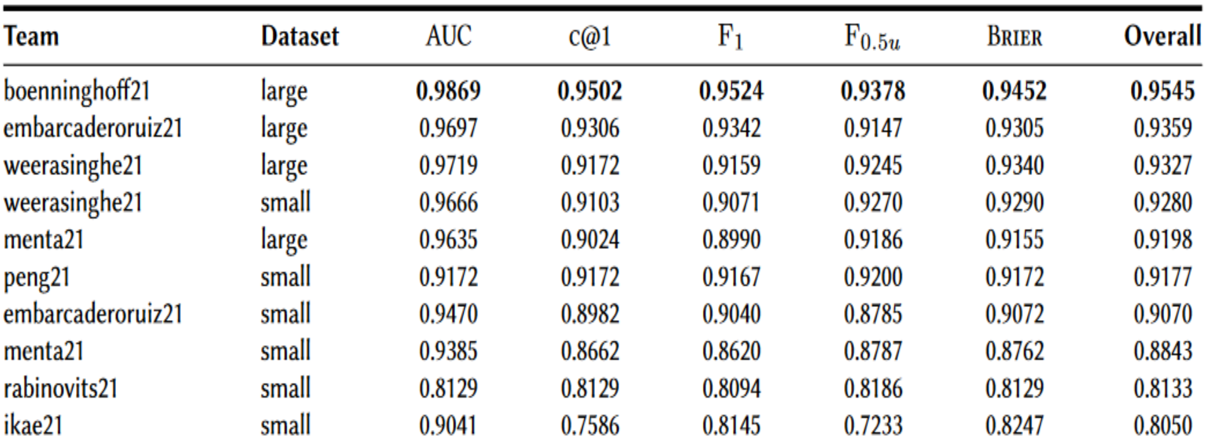
Araujo-Pino et al. (2020) uses a Siamese network architecture that is trained on the n-grams of the characters in the two documents being compared. A siamese network can be thought of as a comparison network in general. It comprises a set of subnetworks that are all the same and a final layer that compares the outputs of the subnetworks. The main idea behind this architecture is to use a subnetwork to extract a set of features from each input and train another network in the same way so that the outputs of the two subnetworks can be compared.

A data compression model derived from a partial matching model prediction and a compression-based cosine similarity method was used by Gagala (2020). In addition, the approach was expanded with context-free grammar character pre-processing, where the most common character n-grams were substituted with a specific symbol to minimize text length and simplify character distribution.

2.5.5 PAN 21

The Dataset used in PAN 21 Kestemont et al. (2021a) was built on top of the Dataset from PAN 20. Fanfiction literature that was scraped from fanfiction.net was used for this task. The training data is the same, but the test data is different.

Bönninghoff et al. (2021) used the same system as they did previously but with a few improvements, adding an uncertainty adaption layer and reducing sensitivity to local fluctuations and an out-of-distribution detector (O2D2) for defining non-responses. This



Team	Dataset	AUC	c@1	F_1	$F_{0.5u}$	Brier	Overall
boenninghoff21	large	**0.9869**	**0.9502**	**0.9524**	**0.9378**	**0.9452**	**0.9545**
embarcaderoruiz21	large	0.9697	0.9306	0.9342	0.9147	0.9305	0.9359
weerasinghe21	large	0.9719	0.9172	0.9159	0.9245	0.9340	0.9327
weerasinghe21	small	0.9666	0.9103	0.9071	0.9270	0.9290	0.9280
menta21	large	0.9635	0.9024	0.8990	0.9186	0.9155	0.9198
peng21	small	0.9172	0.9172	0.9167	0.9200	0.9172	0.9177
embarcaderoruiz21	small	0.9470	0.8982	0.9040	0.8785	0.9072	0.9070
menta21	small	0.9385	0.8662	0.8620	0.8787	0.8762	0.8843
rabinovits21	small	0.8129	0.8129	0.8094	0.8186	0.8129	0.8133
ikae21	small	0.9041	0.7586	0.8145	0.7233	0.8247	0.8050

Figure 2.7: Top PAN'21 Scorers for the Authorship Verification task Kestemont et al. (2021a)

system scored the best among all the submissions, even in PAN 21.

In Embarcadero-Ruiz et al. (2022), the documents were represented in a graph representation, and a graph neural network extracted significant properties from these graph representations. Three methods were described for representing texts as graphs based on the co-occurrence of word POS labels. A Siamese Network architecture comprised graph convolutional networks, pooling, and classification layers.

Weerasinghe et al. (2021b) used a similar algorithm like the one used in PAN 20 Weerasinghe et al. (2021a), with more features and improvements to the classifier. features like character n-grams, POS-tag n-grams, special characters, function word frequency, the average number of characters per word, distributions, POS, stop words, and others were used. Instead of the logistic regression classifier, a Stochastic Gradient Descent training algorithm with a logarithmic loss function was used.

Garuz and García-Serrano (2021) used two neural networks trained on character n-grams and punctuation marks separately. Finally, the network's output was fed into another neural network to get the final result.

Peng et al. (2021) used BERT to extract feature vectors from text fragments and fed them into a neural network.

Embarcadero-Ruiz et al. (2022) represented texts in the form of a graph along with stylistic features and fed it as input into a siamese network. The network consisted of layers to extract the vector representation of each node in the graph.

Pinzhakova et al. (2021) used a random forest model with vector-based features like words, character frequencies, POS tags, stopwords, and others, and absolute differences for scalar features to measure the document similarity.

Ikae (2021) used top-800 TF-IDF weighted word unigrams as input features for different models like linear discriminant analysis, gradient boosting, extra trees, support vector machines, and stochastic gradient descent.

2.6 Conclusion

Different features, models and strategies used for the AV task are discussed and it can be found that previous research involved using word embedding techniques to determine the similarity between document pairs. But context vectors have never been used before for the task of Authorship verification. Using context vectors help in An extensive review of the methods used in previous PAN competitions for Authorship verification has also been done.

Chapter 3

Methodology

This chapter discusses the methods and algorithms used in the research. It also explains the intricate details of the dataset used and the reasons behind its selection. An insight into the various similarity measures and formulae used is also given. Statistical analysis and the tests used during the analysis have also been discussed.

3.1 Dataset

The dataset used for this study was previously used in the PAN Competitions. In addition, it was used in both PAN 2020 and 2021 for authorship verification. The dataset comprises fan fiction, i.e., fictional literature written by amateur writers, which mainly adds literary content to works by existing famous writers (fandoms). These include additions to works like Harry potter, The song of ice and fire, and Star wars, among others. There is an abundance of fanfiction on the internet, and the literary base is also rapidly increasing in size. The dataset used was prepared based on Bischoff et al. (2020). The dataset was made by scraping content from the website fanfiction.net. Only content in the English language was used in the dataset.

In the PAN'20 Authorship verification task, two fan fiction datasets were provided - large and small. The smaller dataset was used in the research. All the text in the dataset was normalized by removing punctuation and improper white spaces and tabs. Every document was prepared such that it had approximately 21,000 characters in it. Documents were categorized according to the author and the fandoms. This disallowed over-representation of a particular author or fandom and allowed for a more uniform comparison of texts in each pair.

The corpus contains 1600 fandoms, and each author has done related work in at least 2 to a maximum of 6 fandoms, with a median of 2 fandoms. Around 148000 pairs of the same author and 128000 different author pairs were scraped from the fanfiction website. Around 41,000 authors were contained in the same author pairs where at least 4 contributed to the same fandom with a maximum of 400 and a median of 29.

Whenever the document had a length greater than 21000 characters, different random portions were used for each pair, and all these pairs were built using all possible pairing of authors and fandoms. The Different Author pairs have 250,000 authors, with each fandom written by at least 2 authors to a maximum of 800 and a median of 51.

The small dataset was created from the large dataset. It contained 28,000 same author pairs and 25,000 different author pairs spanning 1,600 fandoms. The same author pair collection had 6,400 authors, each contributing to at least 4 to 68 fandoms (median 7). In contrast, the different author pair collection had 48,500 authors with 2 to 63 per fandom and a median of 38.

The test data contains 10,000 same author pairs and 6900 different author pairs. These collections had 3500 and 12000 authors, respectively, from 400 fandoms. The text in the test data is not present in the training data. The same author pair collection in the test data has authors who have contributed to 2 to 400 fandoms with a median of 14 and 4 to 800 fandoms in the different author pairs with a median of 20. In addition, each author has written works in 2 to 6 fandoms with a median of 2 Kestemont et al. (2021b).

3.2 Context Vectors

Context Vectors are vector representations used for word sense disambiguation. These vectors capture the occurrence of words around a target word and help identify a word's context. They are of a fixed length and provide insight into how often different words occur in the vicinity of a particular word. In addition, a window size is used, which helps to capture the frequency of different words around the target word within that window.

Machine learning algorithms like neural networks and SVM's can quickly analyze these word representations since they are of fixed length Gallant (1998). Moreover, each of these vectors can be normalized and used in relation to other context vectors to find the similarity between words. For example, 3.3 shows a representation of a target word with a window size of 2. When creating context vectors, the whole corpus is iterated. For each input word, the frequency of the corresponding context word within the window is increased as it is encountered.

In this study, context vectors are used in an ablation study to understand if an author's specific style can be analyzed and used in the task of authorship verification. Rather than using pre-trained models of Word2Vec and GloVe, the aim is to capture the style through the author's text locally using context vectors.

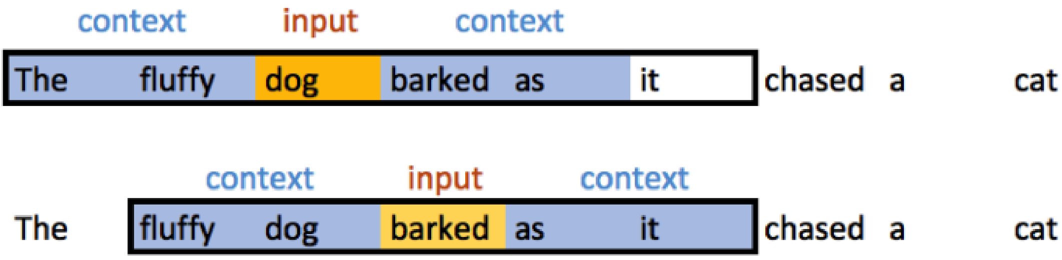

context
input
context
The fluffy dog barked as it chased a cat
context
input
context
The fluffy dog barked as it chased a cat

Figure 3.1: Target and its context words in a window size of 2 Ahire (2021)

3.3 Similarity measures

Text data must be translated into mathematical form before a computer can deal with it. As a result, vectors are commonly used to represent text units (characters, words, phrases, paragraphs, documents). The objective is to represent these text units as numerical vectors, allowing for various operations such as information retrieval, document clustering, and language translation. As a result, the vector space model selected for a specific problem should allow for quick comparisons of documents by measuring the distance between their corresponding vectors, i.e., their similarity. Furthermore, vectors representing the structure or semantics of text units can be computed in various ways.

3.3.1 Euclidean Distance

Euclidean Distance is a token-based similarity distance. Simply put, it is the distance between 2 points. Hence, it is also called the l2 distance.

$$EuclideanDistance = D_{euclidean}(x, y) = \sqrt[2]{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Euclidean distance is the square root of the sum of the squares of the differences between the points in each dimension Rathod et al. (2016). The distance is always positive in value.

3.3.2 Cosine Similarity

In Natural Language Processing, cosine similarity is one of the metrics used to assess the text-similarity of two documents of varying sizes. A word is represented as a vector, and

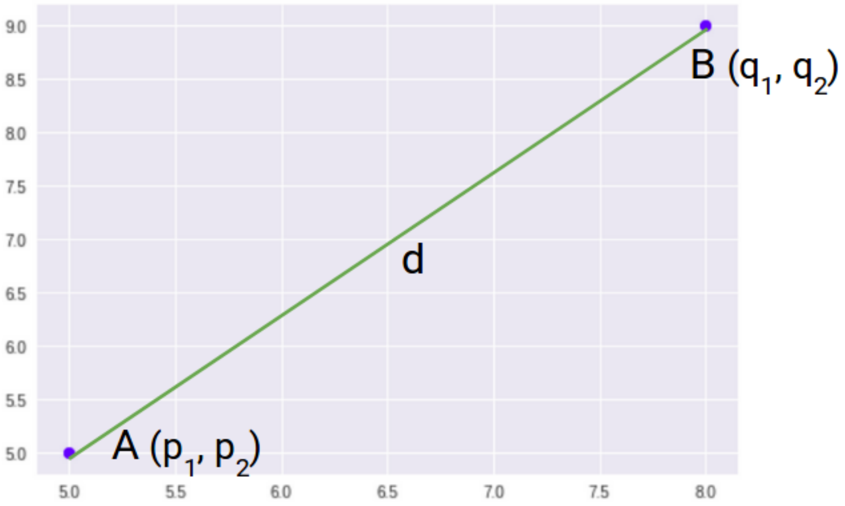


Figure 3.2: Euclidean Distance Sharma (2020)

the text documents are represented as vectors in n dimensions.

Cosine similarity is a mathematical metric that calculates the cosine of the angle between two n-dimensional vectors projected in a multi-dimensional environment. The Cosine similarity between two tokens or words will be between 0 and 1. If the Cosine similarity score is 1, it signifies that the orientation of two vectors is the same. The closer the value is to 0, the less similar the two tokens are.

$$CosineSimilarity = \cos(\theta) = \frac{A.B}{\|A\| . \|B\|} = \frac{\sum_{i=1}^{n} A_i.B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}.\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Cosine Similarity is a better measure than Euclidean distance because even though two text documents are far away in terms of Euclidean distance, they may be near in terms of context.

3.4 Features

Other than context vectors, other features were also used as input to the machine learning model.

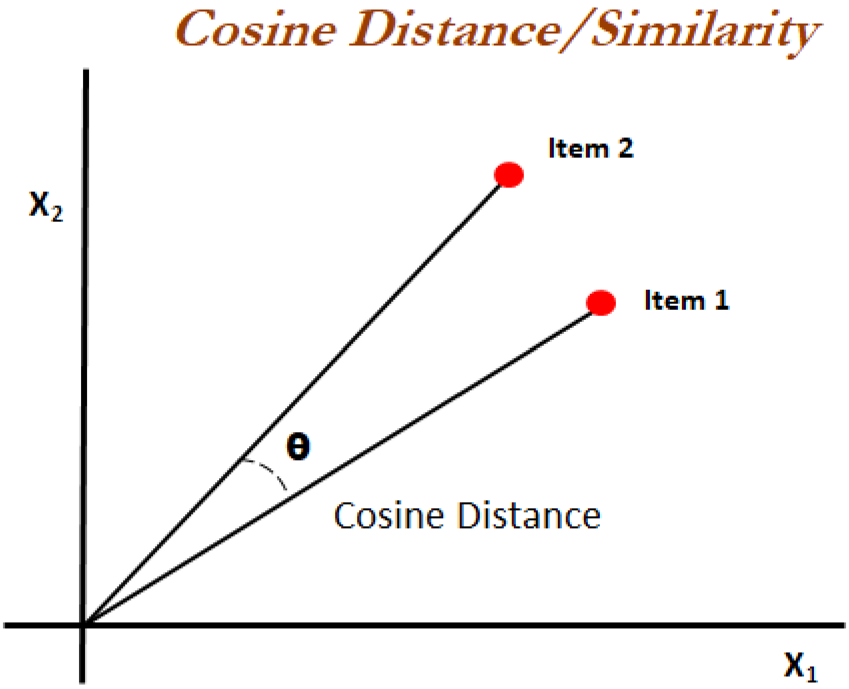


Figure 3.3: Cosine Similarity Dangeti (2021)

3.4.1 Word n-grams

Word N-grams are widely employed in text mining and natural language processing. They are essentially a collection of consequent occurring words inside a specific frame, and when computing n-grams, one word is generally advanced.

e.g., trigrams from the text, *"The house was located at the top of the hill"* are,

```
The house was
house was located
was located at
located at the
at the top
the top of
of the hill
```

NLP uses of n-grams include sentence auto-completion, auto spell-check, and semantic analysis. They are also employed in DNA sequencing and other computational linguistics.

3.4.2 Character n-grams

Character n-grams are one of the most prominent features in authorship identification and are commonly utilized in text classification difficulties. Their key benefit is that they may be translated to a new language with little additional work. In addition, character n-grams that have been typed provide information about their content and context Kruczek et al. (2020).

for the text *we have pasta*, and a window of 3, the character n-grams are,

```
w e _
e _ h
_ h a
h a v
a v e
v e _
e _ p
_ p a
p a s
a s t
s t a
```

3.5 Machine Learning

The task of Authorship verification involves classifying documents if they are from the same author or have been written by a different author. For this research, decision tree regression has been used to analyze the effects of context vectors and other features.

3.5.1 Decision Tree Regressor

Decision Tree is one of the most often employed, practical supervised learning methods. It may be used to tackle both Regression and Classification problems, with the latter being more applicable in practice.

It is a tree-based classifier with three distinct node types. The Root Node is the starting node representing the total sample, which may be subdivided into other nodes. The Interior Nodes describe the data collection characteristics, whereas the Branch Nodes represent the decision rules. Finally, the Leaf Nodes reflect the conclusion. This method is highly beneficial for handling issues involving decisions.

A given data point is run through the entire tree by answering boolean questions until it reaches the leaf node. The final prediction is the mean of the dependent variable's value in that specific leaf node. The tree can accurately forecast the data point's value through numerous iterations. Decision trees offer the advantages of being simple to comprehend, requiring minimal data cleansing, non-linearity having little effect on model performance, and requiring practically no hyper-parameter tuning.

J. R. Quinlan's ID3 core technique for creating decision trees performs a top-down, greedy search over the space of feasible branches without backtracking. The ID3 technique may be used to create a decision tree for regression by substituting Standard Deviation Reduction for Information Gain saed sayad (2020).

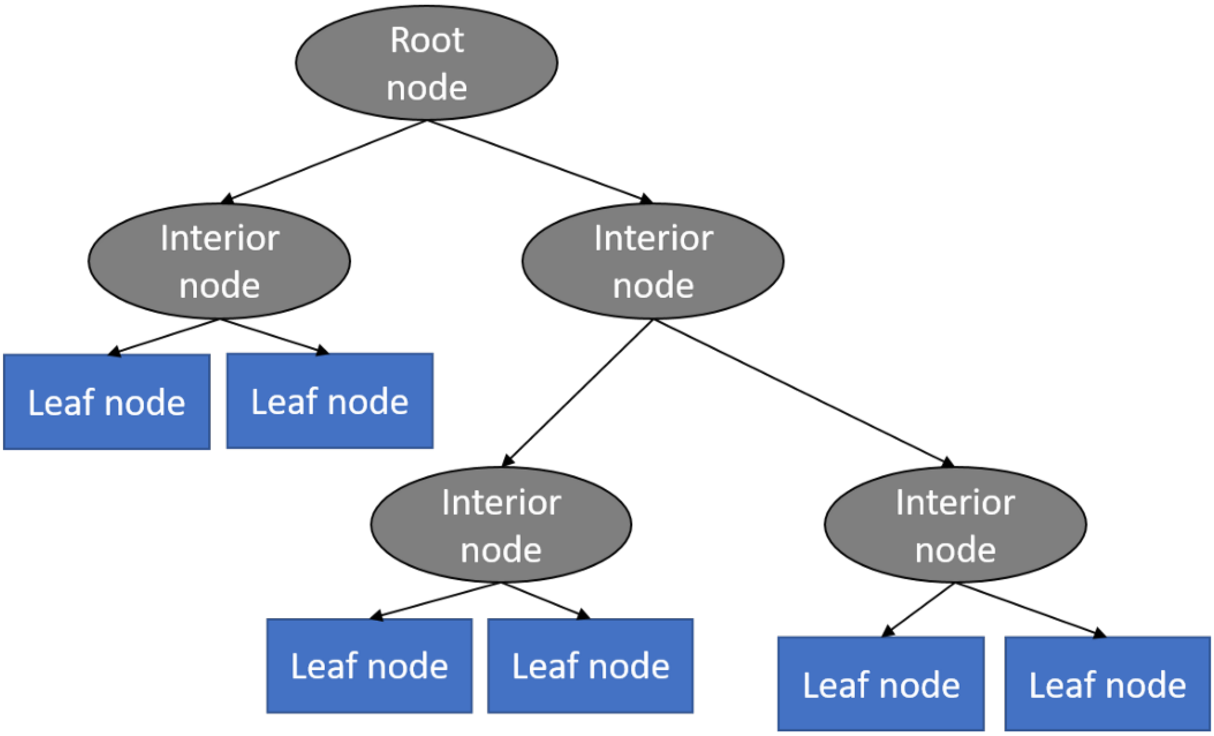


Figure 3.4: Decision Tree Schlagenhauf (2022)

3.5.2 Weka

WEKA is a library with a plethora of machine learning resources that may be accessible in various ways, including through the use of a graphic user interface. WEKA was designed to help users rapidly comprehend the essential methods so that a larger audience might benefit from machine learning. This is accomplished in part by allowing users to access machine learning resources without the need for the user to be comfortable with writing code to apply machine learning algorithms and other components.

WEKA works with other machine learning tools such as Deeplearning4j, sci-kit-learn, and R. Additional information about compatible applications and helpful lesson supplements may be found on the WEKA website.

The data is accessed in WEKA using the WEKA GUI, which loads the dataset with options to change, organize, specify the target feature for classification, and other possibilities. For WEKA to properly utilize the data, it must first be transferred to a .arff file. This is straightforward, utilizing file functionalities included with WEKA's program. WEKA has been involved in "education, research, and industrial applications," according to their website.

3.6 Authorship Verification System

The authorship verification system was created for PAN'13 Author profiling task Moreau and Vogel (2013). Later, more features were added to the system, along with a genetic algorithm and various verification strategies. Some use cases of this system are, given 2 sets of texts A and B, the system creates a model to determine if two given unknown documents or document sets are written by the same author or not.

Another is, Given two sets of documents $A1, .., An$ and $B1, .., Bm$ where all Ai were written by A and all Bj were written by B, are A and B the same author?

The Author verification system has the following features.

Verification Strategies

There are three verification strategies - basic, Universum, and imposter strategies. These strategies take two sets of documents as input, where all of them in one set are written by one author. They return a collection of feature values deemed necessary for resolving the issue of whether the same author writes two sets of documents. Each strategy can have multiple parameters given through a config file.

Genetic Algorithm System

This applies a genetic algorithm to determine the best parameters and verification system. Finally, all these models are combined using a stacked generalization method to give rise to good accuracy.

Extensible and Scalable

The system is designed to be extensible. Extending it by adding new strategies, features, and learning methods are effortless. It also uses distributed computing and has packages on it.

Supervised and Unsupervised Learning

The system supports both supervised and unsupervised learning through the use of weka, which is an open-source machine learning library in java.

3.7 Evaluation Methods

Evaluation of Machine Learning models is the final step before a model's production deployment. First, we analyze Machine Learning models to verify that they function as predicted and are adequate for the purpose they were designed for. The evaluation phase follows the completion of model training.

Various strategies are employed based on the nature of the problem and algorithm. Most assessment procedures compare training data with test data derived from training data. The evaluation methods used in this research are explained below.

3.7.1 Accuracy

This is one of the machine learning models' most common evaluation methods. Accuracy is the ratio of correct predictions to the total number of predictions. It can be calculated by

$$accuracy = \frac{CorrectPredictions}{TotalPredictions} = \frac{(TP + FN)}{(TP + FN + FP + TN)}$$

where,

TP is True Positive, which are values that are true and are classified true.

FP is False Positive, values which are false but were incorrectly predicted as true.

TN is True Negative, values that are true but have been incorrectly predicted as false.

FN is False Negative, which are values that are False and are correctly predicted as false.

3.7.2 AUC (Area under Curve)

A ROC curve (receiver operating characteristic curve) is a graph that illustrates the performance of a classification model's overall categorization levels. This graph illustrates two parameters: True Positive Rate and False Positive Rate.

True Positive Rate, also called recall, can be computed by,

$$TPR = \frac{TP}{TP + FN}$$

False positive Rate is computed by,

$$FPR = \frac{FP}{FP + TN}$$

A ROC curve compares TPR to FPR at various categorization levels. By lowering the categorization threshold, more items are classified as positive, increasing both False Positives and True Positives.

The Area Under the Curve (AUC) is a summary of the ROC curve that shows how well a classifier can tell the difference between different categories. It shows how well the model differentiates between positive and negative categories. Models with higher AOCs classify better and more accurately than those with lower AOCs.

3.7.3 Statistical Analysis

Collecting and evaluating data, either large or small, to discover patterns and trends is known as statistical analysis. It is part of data analytics. Statistical analysis may be used for various purposes, including obtaining research interpretations, statistical modeling, and creating investigations.

The tests here can be used in hypothesis testing, for example, to check if there are relationships between variables or if some variables affect the overall experiment.

Statistical tests are based on the null hypothesis, which states no link or difference between groups. The observed data are next examined to see if they fall beyond the range of outcomes predicted by the null hypothesis.

These tests compute a p-value (probability value). The p-value estimates the likelihood that the difference reflected by the test statistic would be seen if the null hypothesis of no association were true. If the test statistic has a more extreme value than the null hypothesis statistic, a statistically significant association can be inferred between the predictor and outcome variables.

Suppose the test statistic's value is less significant than the null hypothesis's. In that case, it can be concluded that there is no statistically significant association between the predictor and outcome variables.

Different tests can be implemented depending on the probability distribution of the data. There are two types of tests,

1. Parametric Tests: These are Statistical tests that, among other things, presume that the data follows a normal distribution. e.g., ANOVA, T-Tests.

2. Nonparametric Tests: These are Statistical tests that make no assumptions about

the data distribution. They are also known as distribution-free tests, e.g., Mann-Whitney. Nonparametric tests are based on the rankings of several data items.

Shapiro test

The shapiro test is a test used for checking the normality of the data. The null hypothesis of the test states that the data is normally distributed while the alternative hypothesis says otherwise. The test statistic is,

$$W = \frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

where,
x_i is the ith Statistic,
$\bar{x}$ is the sample mean.

Welch's two sample test

The Welch's Test for Unequal Variances is a version of the Student's t-test used to determine whether or not two sample means are significantly different. This test is also known as the Welch's t-test, the Welch's adjusted T, or the unequal variances t-test. This change is being made to the degrees of freedom that are being applied to the test, which has the effect of increasing the test power for samples that have uneven variance. The null hypothesis for this test is that the means are equal, while the alternative hypothesis says its not.

It can be calculated by,

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}}}$$

where,
S_1 and S_2 are the standard deviations,
$\bar{X}_1$ and $\bar{X}_2$ are the sample means,
and N_1 and N_2 are the sample sizes.

Kruskal-Wallis Test

The Kruskal - Wallis test is a nonparametric test used to find differences between multiple independent groups. It is the nonparametric counterpart to the single factor analysis of variance.

Ordinal variables are adequate in the Kruskal-Wallis test since nonparametric approaches employ rank places rather than value differences. As a result, the Kruskal-Wallis test is known as rank variance analysis after Kruskal and Wallis. It is a nonparametric alternative to one-way ANOVA and an extension of the Mann-Whitney U test that allows for comparing more than two independent groups.

The null hypothesis for this test is that there is no difference in the rank sums, while the alternative hypothesis would be that at least one group differs in the rank sums.

This is calculated using,

$$K = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N-1)$$

where,

R_i is the sum of ranks for the ith group

3.8 Conclusion

The methodology chapter reviews the intricate details about the dataset selected along with the global dataset. Context vectors and their working were also discussed. All the similarities for comparing the vectors and creating the similarity matrix along with the other features are also discussed. The machine learning library and the algorithm used are discussed. Finally, the evaluation methods used in the research to understand how context vectors effect and produce relationships between its different configurations are also discussed.

Chapter 4

Implementation

This chapter implements the concepts discussed in the methodology for the task of authorship verification using context vectors. The chapter discusses the data preparation for the authorship verification system and implementation of context vectors, including integration and usage. Finally, using supervised learning to create a machine learning model that can classify documents for the task in question.

4.1 Data Collection and Preparation

As mentioned, the dataset used for this research was first used in PAN competitions in 2020 and 2021. The dataset contains parts of fanfiction literature scraped from the website fanfiction.net fac. All the documents in the dataset are in the English language.

The data was obtained for the official PAN website - . The data is free to use but has restricted access. One has to request to use the data. The data has two parts, one which contains the data, i.e. document pairs, and the latter which contains the truth values of the document pairs.

The file with the document pair contains data such as fandom names, author ID, and the literature pairs. This data is in the *jsonl* format, where each line is a JSON value, and a new line separates every line. The truth value file has each corresponding entry with information if the pair has been written by the same author or not and also includes the author ID.

This had to be converted into a format that the authorship verification system could accept and process. The system accepts the following types of files,

1. The Input text documents present in an Input folder

2. A text file called cases.txt, which contains data in the format,

```
<document1> <document2> truth_value
```

a tab precedes the truth value.

3. A text file named test-cases.txt, which contains the test cases. The entries in this file are in the format,

```
<document1> <document2>
```

4. A text file named test-cases.gold.txt contains the test cases and their truth values.

To prepare the dataset for usage, a python program was written that iterates through each line of the jsonl file. Every line in the jsonl file contains a problem with information like the ID, fandoms, and document pairs. The script creates a pair of text files from the document pair present in each problem, and writes the names of the text files along with the truth value corresponding to the pair retrieved from the truth value file. Currently, the file name is given according to an iterator value which is converted into a string.

This is how the dataset provided in the PAN competition has been converted into a format that the authorship verification system can use.

The code 4.1 is used to generate the text pairs and the truth value.

```
import json

def createDataset():
    with open('pan20-authorship-verification-training-small.jsonl', 'r')
        as json_file:
        with open('pan20-authorship-verification-training-small-truth.
            jsonl', 'r') as truth_file:
            truth_list = list(truth_file)
            with open("truthvalues.txt", "a") as cases:
                json_list = list(json_file)
                i = 0
                for entry in json_list:
                    js = entry
                    result = json.loads(js)
                    textPair = result['pair']
                    #first file
                    with open("data/" + str(i) + ".txt", 'x', encoding
                        ='utf-8') as f1:
                        f1.write(textPair[0])
                        f1.close()
                    i += 1

                    #second file
                    with open("data/" + str(i) + ".txt", 'x', encoding
                        ='utf-8') as f2:
                        f2.write(textPair[1])
                        f2.close()
                    i += 1

                    truthjs = truth_list[i]
                    tresult = json.loads(truthjs)
                    ans = 0
                    if (tresult['same'] == True):
                        ans = 1

                    cases.write(str(i-2) + ".txt " + str(i-1) + ".txt "
                        + str(ans) + "\n")

                cases.close()

createDataset();
```

4.1.1 Data Preprocessing

The documents used as input to the machine learning algorithm were first processed. Punctuation was removed, and excessive white spaces and tabs were also removed. Stop words were not removed since even they contributed to the style and the context analysis. The text was converted into lower case so that there will not be duplicate entries while generating context vectors due to case.

4.1.2 Global Corpus

To find similarity measures, the context vectors of the tokens from a document must be compared with something akin to a reference. For this purpose, a data dump that contained fanfiction was scraped from the same website, fanfiction. fac was selected.

The data dump was obtained from the subreddit datasets u/nerdguy1138 (2019). All the files in the data dump were preprocessed to include only the literature, which was in English since the data from the PAN competition dataset was also in English.

Only the literature content from each file present in the corpus was processed and used in creating the global corpus. The files were then appended into a single global text file to be used later as the global corpus to be compared with the context vectors of the document to be analyzed.

4.2 Context Vector Implementation

An implementation of context vectors has been done in the authorship verification system. First, the document is iterated, and the frequencies of each token are stored in a hash. This hash is sorted so that the most frequently occurring tokens occur in the beginning. A corresponding hash recording the indexes of the tokens in the other hash is also maintained. The former hash contains the token and frequency as the key and value pairs, and the latter has the token and the index as the key-value pairs. Another hash with the corresponding index and token is also maintained.

The global corpus is also iterated, and all the words present in the global corpus and not present in the document are added to the hashes, and their frequency is set to zero. One issue here was the large size of the corpus. So a stream was used to read the file rather than load it all at once in the RAM. The program was also changed so that everyline that is read from the text file is properly processed with regards to the window size and the words came before the line and were in context with it. The same was done to the words which would come later on in the context of the token currently in analysis.

The document is again iterated, and for each token, the frequency of n words after and before the target token is recorded in two different hashes. The indexing and positioning of each occurrence are done with the help of the frequency and the ranking hashes. Each token's before and after context word frequency arrays are appended and processed into a flat array.

The process is repeated with the documents interchanged and context vectors generated again. These context vectors are compared using a similarity measure to create a similarity matrix.

Various statistics like mean, min, max, various quantiles, std, median, and others are calculated from the similarity matrix. Context vectors capture the style of the author of the text. as in, if the author uses a particular vocabulary in context with other words, then this is captured by the technique used above.

The implemented context vectors algorithm accepts a few parameters like window size, type of config, number of tokens, and similarity type.

1. Window Size: Determines the number of context words before and after the input token to be considered.

2. Config Type: This option determines the type of configuration, which are all tokens, m most frequent tokens, and singleton tokens.

3. Similarity Type: This option specifies the similarity type to be used while calculating the similarity matrix. Options include cosine similarity and euclidean similarity, among others.

This context vector implementation is added as an observation type in the authorship verification system. A configuration file is used to apply a strategy or even train a model.

The context vector implementation code is present in the Appendix, A.

4.2.1 Measures

To calculate the similarity matrix, multiple similarity measures were used. Measures like Cosine similarity and euclidean similarity were used for this study.

The authorship verification system was extensible in terms of adding new observation types like context vectors and measures like similarity calculators, but one drawback of the system was that it was not designed to handle vectors. One of the assumptions it was built on was that only scalar values would be used during implementation.

Since context vectors used vector values, the system had to be modified to accommodate even vectors. The system was modified by using the decorate design pattern to accommodate different types of input given to it.

These measures can be given through the config file and in the case of context vectors, based on the similarity type provided in the parameters discussed previously, that particular similarity would be used while calculating the similarity matrix.

4.2.2 Configurations

Configuration files contain a wide range of vital information like the different parameters to be used while applying a strategy, information about the machine learning parameters, etc.

Weka, which is an open source java machine learning library, is used for machine learning. For this study, the algorithm used is a decision tree regressor. All this information, along with strategy type, learning parameters, and other general options like tokenization, are present in the config files.

While running the system, different configurations were used in both Imposter and Universum strategies. For each of these strategies, cosine similarity and euclidean distance measures were used with configurations like all tokens, top 30 tokens, and singletons for context vectors and word unigram, bigram, and character trigram. The similarity measurements were added as measures in the system. These are specified in the config file.

The script 4.2.2 shows the configuration for the baseline model of the General Imposter method.

```
strategy=GI

obsType.CHAR.CCC.lc1.sl0.mf3=1
obsType.WORD.T.lc1.sl0.mf5=1
obsType.WORD.TT.lc1.sl0.mf5=1
obsType.VOCABCLASS.MORPHO.mf5=1

basic.simMeasure=minmax

# general options
multipleProbeAggregate=random
wordTokenization=1
formatting=0

# supervised learning parameters
confidenceTrainProp=0
confidenceLearnMethod=simpleOptimC1
learnMethod=M5P-M4
```

For the context vectors, the observations are specified in the form of
$obsType.CONTEXT.lc(01).sw(01).n(i).nfs(123).fn(i).sm(123).mf(i)$
where,
lc stands for lower case and takes the values 0 or 1.

sw stands for stop words and takes the values 0 or 1.

n represents the window size while computing the context vector and can have any positive value.

nfs represents the context vector configuration type, including all tokens, top *i* tokens, or singleton tokens. This value takes in a number between 1 and 3.

fn represents how many top n occurring tokens to consider if that configuration is selected in the previous option. This can take any value. The study considered the top 30 tokens to perform the ablation test.

sm represents the similarity measure, which takes the values between 1 and 2; depending on this, a similarity measure is used. One is used to select Cosine similarity and 2 to select Euclidean distance.

mf stands for minimum frequency

4.3 Baseline

For the baseline models, 3 features were used - Word Unigrams, Word Bigrams, and Character Trigrams.

These features are used along with measures like cosine similarity, euclidean distance and minmax. The baseline model was used with the same dataset and measures that were used in the other tests so that the effect of context vectors in different ablation tests could be compared with this.

4.4 Machine Learning

For supervised learning, a particular standard directory structure is to be maintained. Three folders are used: input, model, and output. The input folder contains all the input documents which need to be analyzed. The trained model will be stored in the model folder, and the output folder will store the output after testing the trained model on the test dataset.

A case in the system is a pair of sets of documents, with the texts within a group presumed to be written by the same author. A case can be labelled, that is, supplied with the gold-standard solution indicating whether or not the two sets of documents are from the same author. A set of labelled instances may, of course, be used to train a supervised model. The supervised learning task is implemented as a regression task.

During training, the cases are given a response of 0 (different author) or 1. (same author). For numerous such examples, the model is trained to anticipate this numerical value based on the characteristics returned by the script, verify-author.pl.

When the model is used, the predicted value for the answer might range from 0 to 1. A score close to 0 implies that the author is most likely different, whereas a value around 0.5 indicates doubt.

5-Fold cross-validation is also applied to train and evaluate the model to avoid overfitting. Here the dataset to divided into 5 chunks where each chunk is further divided into equal-sized parts. One of these parts is selected randomly as the testing data, and the trained model is tested on it.

The machine learning algorithm is a decision tree classifier since the task at hand is a classification one. This is given as input using the config file mentioned above. The Weka library is used here to train and test the machine learning algorithm.

After training and testing the model, the predictions are compared with the gold-standard values given. The components of the classification matrix are calculated from the evaluation. The classification matrix contains values like true positives, false positives, true negatives, and false negatives. The AUC (Area under curve) and accuracy can be calculated and obtained from this matrix.

4.5 Flow of Control

First the case file which contains the training data is iterated one after another. Each file from the document pair is read using the name present in the case file. The configuration file is read and the different parameters present in it, like strategy type, observations (features to be obtained) along with their options, the measure type are all processed. The format of the observation types, whether they are valid is checked.

Various preprocessing steps such as stop words, punctuation, lower case, and others are performed and the text is sent to the next stage where the related observation type is extracted from it.

Depending upon the parameters in the configuration file, the text is processed and features are obtained. The measure helps in computing the features after the text has been analyzed and the observations(e.g. trigrams) have been generated.

A set of features are obtained from each problem and these are provided as input to the machine learning algorithm. Training is done on the machine learning algorithm which is decision tree classifier is done to create a model. This model is then applied on the testing set and the similarity measure is obtained as the output.

The process is repeated for all the other configurations to be tested and the results are recorded.

4.6 Conclusion

The chapter talks about how the research was implemented. It contains details about the datasets used and how they were processed, the algorithm used in the implementation of the context vectors. It also talks about how the authorship verification system works, its drawbacks, the machine learning process and the different features and configurations used in the research. Finally an overview of the flow of control of the system is also provided.

Chapter 5

Evaluation

This Chapter deals with the evaluation of the results obtained after running different configurations of context vectors with different similarity measures and strategies. Statistical analysis has been performed including t tests to understand the effect that context vectors have on the AV task.

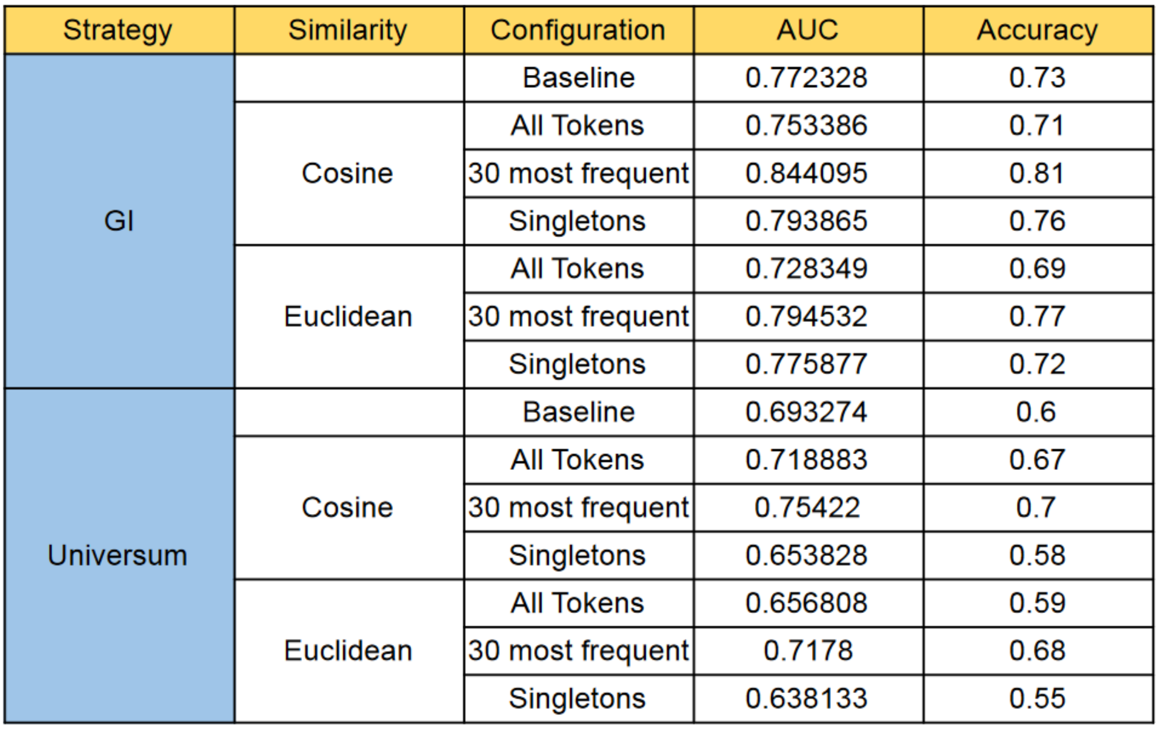
5.1 Evaluation of aggregated results

After running each configuration of the context vectors along with the baseline models with different strategies, the following results have been obtained. These are aggregations based on all individual instances run.

There is a slight improvement in some configurations and decrease in accuracy in others. Statistical analysis was done on these aggregated results to check if there is any significant change in the categories after context vectors have been added

The tests done are mentioned below,

```
D <- read.csv("thesis-results.csv",header=TRUE,stringsAsFactors=TRUE)
summary(D)
shapiro.test(D$AUC)
shapiro.test(D$Accuracy)
with(D,t.test(AUC~Strategy))
with(D,t.test(Accuracy~Strategy))
with(D[D$Strategy=="GI",],kruskal.test(AUC~Similarity))
with(D[D$Strategy=="Universum",],kruskal.test(AUC~Similarity))
with(D[D$Strategy=="Universum",],kruskal.test(Accuracy~Similarity))
with(D[D$Strategy=="GI",],kruskal.test(Accuracy~Similarity))
```



Strategy	Similarity	Configuration	AUC	Accuracy
GI		Baseline	0.772328	0.73
	Cosine	All Tokens	0.753386	0.71
		30 most frequent	0.844095	0.81
		Singletons	0.793865	0.76
	Euclidean	All Tokens	0.728349	0.69
		30 most frequent	0.794532	0.77
		Singletons	0.775877	0.72
Universum		Baseline	0.693274	0.6
	Cosine	All Tokens	0.718883	0.67
		30 most frequent	0.75422	0.7
		Singletons	0.653828	0.58
	Euclidean	All Tokens	0.656808	0.59
		30 most frequent	0.7178	0.68
		Singletons	0.638133	0.55

Figure 5.1: Accuracy and AUC of Each Configuration

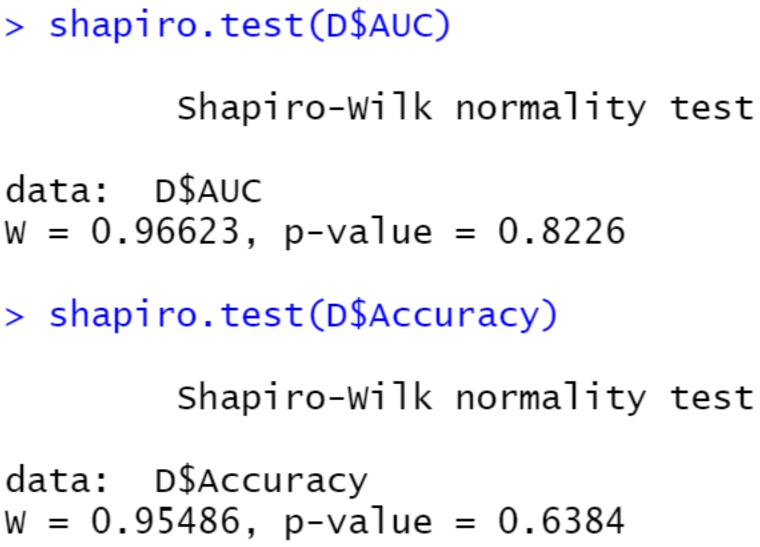
After the CSV file was read, Shapiro test was done on the accuracy and AUC fields to test for normality, which resulted in 5.2.

This shows that the null hypothesis can be accepted and the data is normally distributed. Welch two sample test 5.3, 5.4, was then used to compare the means by strategy in both AUC and Accuracy, this test gave p values less than 0.05 in both the strategies.

This implies that for both AUC and Accuracy there is a significant difference between the two strategies.

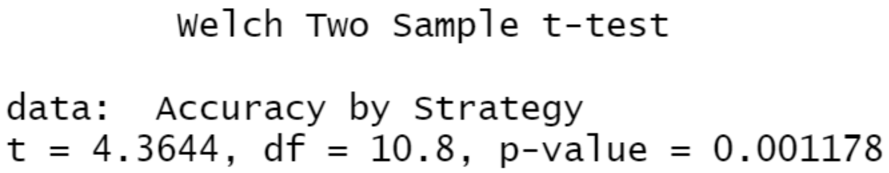
Next, Kruskal-wallis rank sum test was done to check if there is significant differences between Similarities and Configurations wrt AUC and Accuracy 5.5, 5.6. This unfortunately gave p values above 0.05, which might mean with the given data, there is not much significant differences between Similarities or Configurations.

Since these were based on results which were aggregated, these might not be accurate. If the dataset consisted of all instances of the configurations run, the results would be different.



```
> shapiro.test(D$AUC)

        Shapiro-Wilk normality test

data:  D$AUC
W = 0.96623, p-value = 0.8226

> shapiro.test(D$Accuracy)

        Shapiro-Wilk normality test

data:  D$Accuracy
W = 0.95486, p-value = 0.6384
```

Figure 5.2: Results of the shapiro test



```
        Welch Two Sample t-test

data:  Accuracy by Strategy
t = 4.3644, df = 10.8, p-value = 0.001178
```

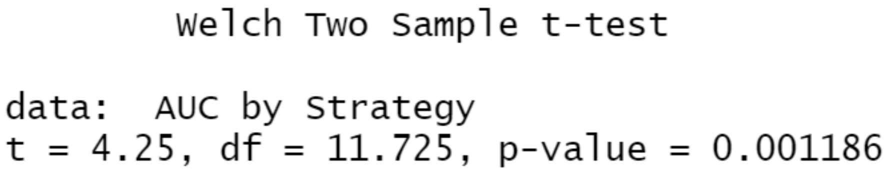
Figure 5.3: Results of the Welch two sample test for accuracy

5.2 Evaluation of detailed results

Statistical analysis was again conducted now on data consisting of every instance of the test set run along with output such as similarity measure from the comparison of the two unknown documents and the overall binary result

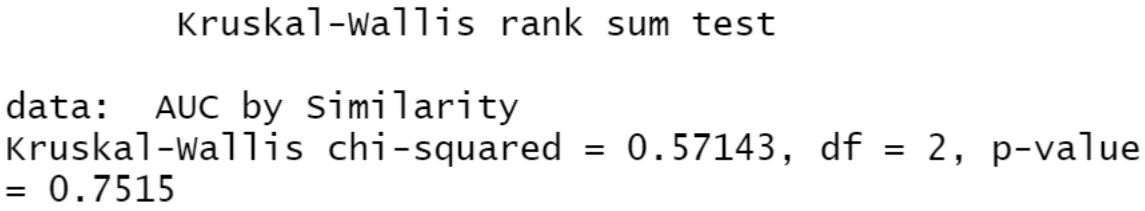
The following tests were run on the extensive results data.

```
DSKM  <- read.csv("final - final.csv",header=TRUE,stringsAsFactors=TRUE)
summary(DSKM)
DSKM$Error <- with(DSKM,(gold.case-predicted))
with(DSKM,lm(abs(Error)~Configuration*Strategy*Similarity))
summary(with(DSKM,lm(abs(Error)~Configuration*Strategy*Similarity)))
with(DSKM,interaction.plot(Configuration,Similarity,abs(Error)))
with(DSKM,tapply(abs(Error),list(Configuration,Similarity),mean))
with(DSKM,interaction.plot(Configuration,Strategy,abs(Error)))
with(DSKM,tapply(abs(Error),list(Configuration, Strategy),mean))
with(DSKM,kruskal.test(abs(Error)~Configuration))
```



```
            Welch Two Sample t-test

data:  AUC by Strategy
t = 4.25, df = 11.725, p-value = 0.001186
```

Figure 5.4: Results of the Welch two sample test for AUC



```
        Kruskal-Wallis rank sum test

data:  AUC by Similarity
Kruskal-Wallis chi-squared = 0.57143, df = 2, p-value
= 0.7515
```

Figure 5.5: Results of the Kruskal-wallis test for AUC in GI

```
11 M <- with(DSKM,lm(abs(Error)~Configuration))
12 summary(M)
```

Here the CSV is first read and statistics on the data are obtained in line 1. Then error for each instance is calculated using the difference between the actual value and the predicted value. A linear model is fitted with error grouped by Strategy, Configuration and Similarity in line 5. From the results below in 5.8, 5.7, 5.9, we see that for some coefficients, the p value is less than 0.05, meaning it gives value to the model, because changes in the predictor's value are related to changes in the response variable.

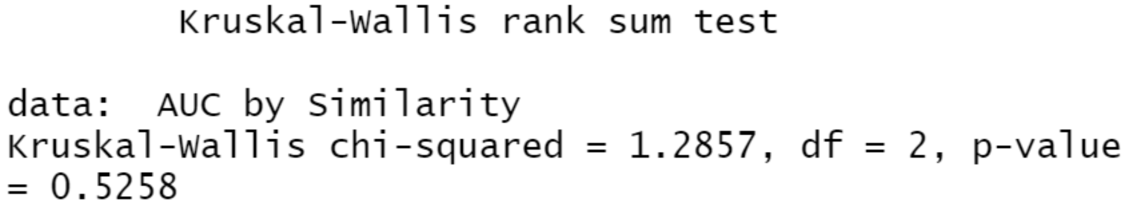
The mean error is then calculated in line 7, after this for different combinations of configuration and strategy which results in 5.10.

Here we can see that the mean errors in configurations using Euclidean distance is more than those using Cosine similarity. Similarly, in all the configurations, the baseline (setup without context vectors) has the highest mean errors. This is followed by the top 30 tokens, Singletons, and then all the tokens.

The values under the minmax column is missing for the context vectors configuration because that type of measure was designed to handle only scalar values and not vectors or arrays.

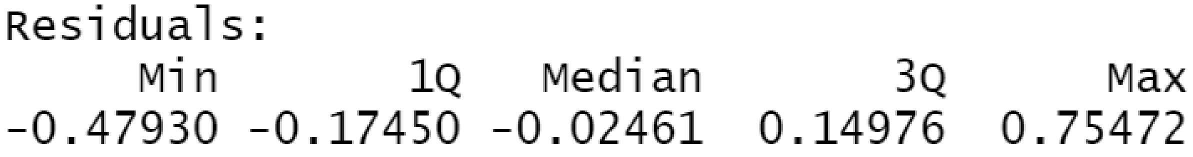
So, there is an interesting pattern in the values of the mean errors using cosine and euclidean measures. 5.11 shows the graph between configurations and mean absolute errors for different similarities.

From 5.11, we can see that the mean absolute error is highest for the baseline, followed by "all tokens", singletons, and "Top 30 tokens". And mean absolute error is lesser for



```
        Kruskal-Wallis rank sum test

data:  AUC by Similarity
Kruskal-Wallis chi-squared = 1.2857, df = 2, p-value
= 0.5258
```

Figure 5.6: Results of the Kruskal-wallis test for AUC in Universum



```
Residuals:
     Min       1Q   Median       3Q      Max 
-0.47930 -0.17450 -0.02461  0.14976  0.75472 
```

Figure 5.7: Residuals of linear model

cosine similarity than euclidean distance.

The mean absolute error is then calculated for different Strategies in line 9. The results are present in 5.12. From the values obtained, it can be seen that the mean error for GI method is less than that of Universum method. Though there is a slight deviation with the universum method with singleton tokens. Apart from this single deviation, we still see that the top 30 frequent tokens have the lowest error, followed by singletons, every token and baseline at the end. The graph in 5.13 shows the mean absolute errors at different configurations and strategies.

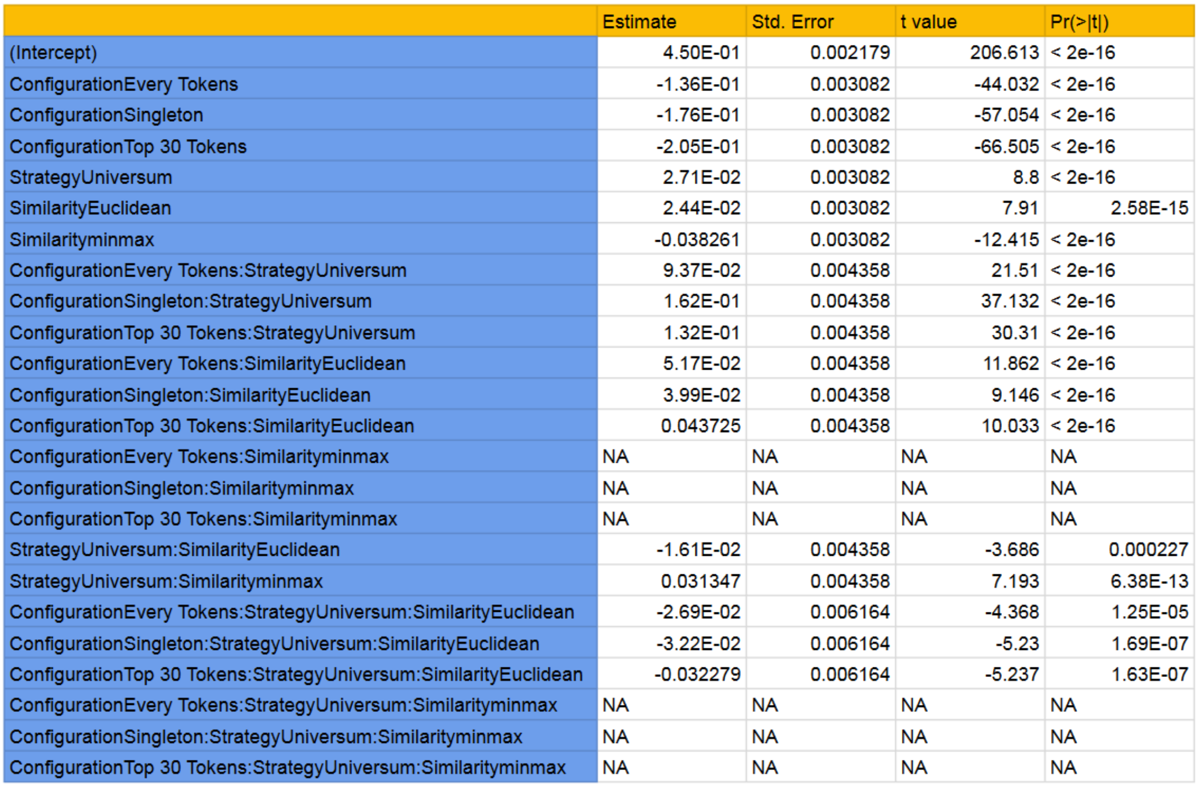
In line 10, the krushkal-wallis test is used on absolute error for different configurations. The result shows the p-value obtained from the test. The p-value is $2.2e-16$ which is less than 0.05. This shows that the null hypothesis, which is there is no significant difference in the different configurations, can be rejected.

This means that there is a significant difference between different configurations and taking into account the previous mean absolute errors we can say that context vectors effect the task of authorship verification.

5.3 Conclusion

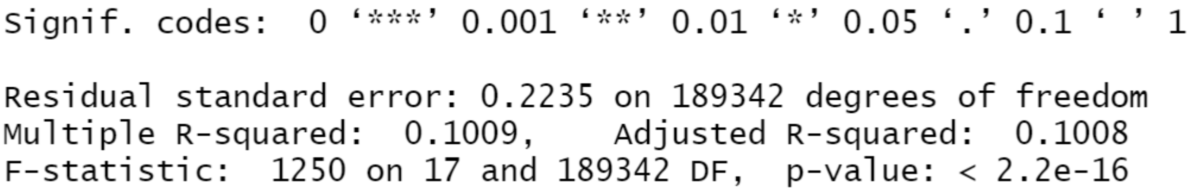
The chapter explains the process undertaken while performing statistical analysis. Analysis on both the aggregated and the detailed results is performed. It also explains the output of each test and what it means with regards to the relationships between different configurations of the usage of context vectors.

The analysis showed that the mean absolute error is lowest when using top n frequent



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	4.50E-01	0.002179	206.613	< 2e-16
ConfigurationEvery Tokens	-1.36E-01	0.003082	-44.032	< 2e-16
ConfigurationSingleton	-1.76E-01	0.003082	-57.054	< 2e-16
ConfigurationTop 30 Tokens	-2.05E-01	0.003082	-66.505	< 2e-16
StrategyUniversum	2.71E-02	0.003082	8.8	< 2e-16
SimilarityEuclidean	2.44E-02	0.003082	7.91	2.58E-15
Similarityminmax	-0.038261	0.003082	-12.415	< 2e-16
ConfigurationEvery Tokens:StrategyUniversum	9.37E-02	0.004358	21.51	< 2e-16
ConfigurationSingleton:StrategyUniversum	1.62E-01	0.004358	37.132	< 2e-16
ConfigurationTop 30 Tokens:StrategyUniversum	1.32E-01	0.004358	30.31	< 2e-16
ConfigurationEvery Tokens:SimilarityEuclidean	5.17E-02	0.004358	11.862	< 2e-16
ConfigurationSingleton:SimilarityEuclidean	3.99E-02	0.004358	9.146	< 2e-16
ConfigurationTop 30 Tokens:SimilarityEuclidean	0.043725	0.004358	10.033	< 2e-16
ConfigurationEvery Tokens:Similarityminmax	NA	NA	NA	NA
ConfigurationSingleton:Similarityminmax	NA	NA	NA	NA
ConfigurationTop 30 Tokens:Similarityminmax	NA	NA	NA	NA
StrategyUniversum:SimilarityEuclidean	-1.61E-02	0.004358	-3.686	0.000227
StrategyUniversum:Similarityminmax	0.031347	0.004358	7.193	6.38E-13
ConfigurationEvery Tokens:StrategyUniversum:SimilarityEuclidean	-2.69E-02	0.006164	-4.368	1.25E-05
ConfigurationSingleton:StrategyUniversum:SimilarityEuclidean	-3.22E-02	0.006164	-5.23	1.69E-07
ConfigurationTop 30 Tokens:StrategyUniversum:SimilarityEuclidean	-0.032279	0.006164	-5.237	1.63E-07
ConfigurationEvery Tokens:StrategyUniversum:Similarityminmax	NA	NA	NA	NA
ConfigurationSingleton:StrategyUniversum:Similarityminmax	NA	NA	NA	NA
ConfigurationTop 30 Tokens:StrategyUniversum:Similarityminmax	NA	NA	NA	NA

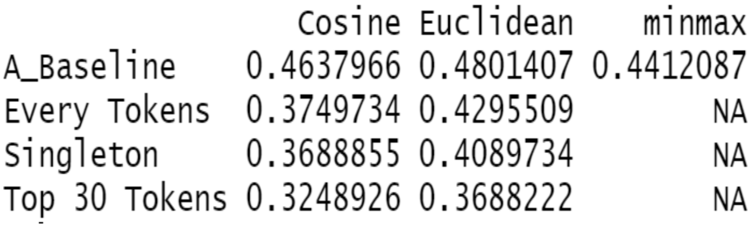
Figure 5.8: Summary of linear model based on error



```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2235 on 189342 degrees of freedom
Multiple R-squared:  0.1009,	Adjusted R-squared:  0.1008 
F-statistic:  1250 on 17 and 189342 DF,  p-value: < 2.2e-16
```

Figure 5.9: Adjusted R-squared value and p value of linear model

token, followed by singleton tokens and every token, with the baseline having the highest error. Configurations with cosine similarity also performed better than euclidean distance and the GI strategy was better than the universum strategy. The kruskal-wallis test showed that there is significant difference in the values between different variables.



```
                Cosine Euclidean   minmax
A_Baseline    0.4637966 0.4801407 0.4412087
Every Tokens  0.3749734 0.4295509        NA
Singleton     0.3688855 0.4089734        NA
Top 30 Tokens 0.3248926 0.3688222        NA
```

Figure 5.10: Mean Absolute Errors for different configurations and similarities

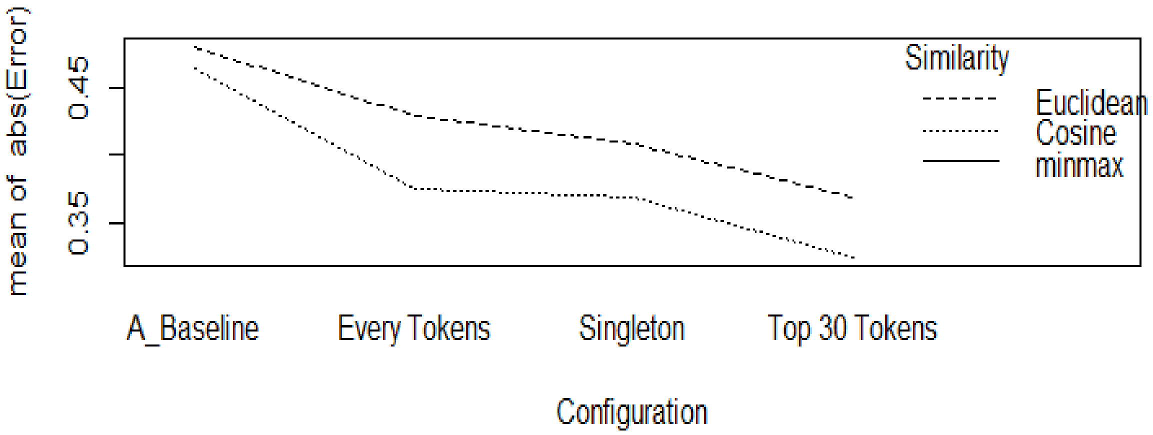
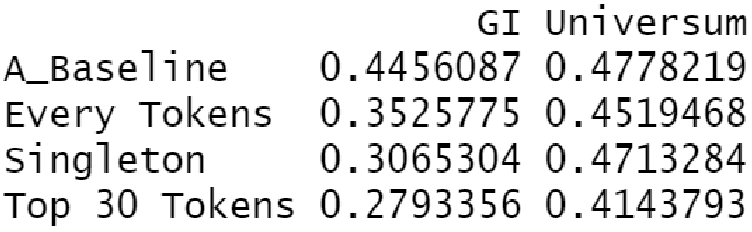


Figure 5.11: Graph between configurations and mean absolute errors for different similarities



```
                     GI  Universum
A_Baseline    0.4456087 0.4778219
Every Tokens  0.3525775 0.4519468
Singleton     0.3065304 0.4713284
Top 30 Tokens 0.2793356 0.4143793
```

Figure 5.12: Mean Absolute Errors for different configurations and strategies

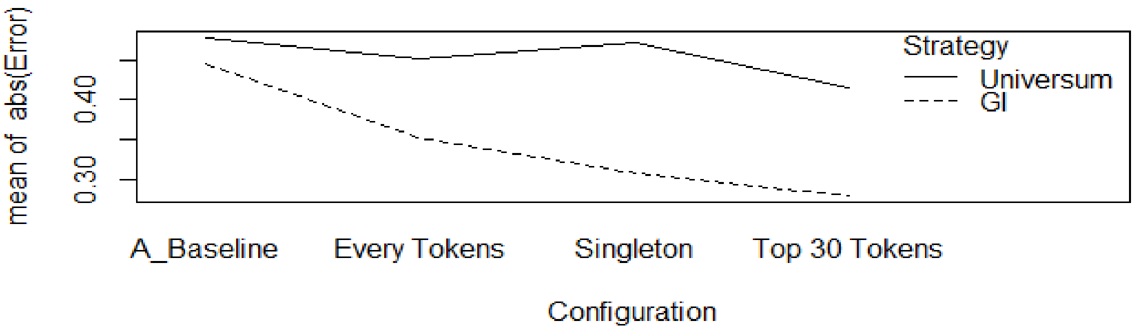
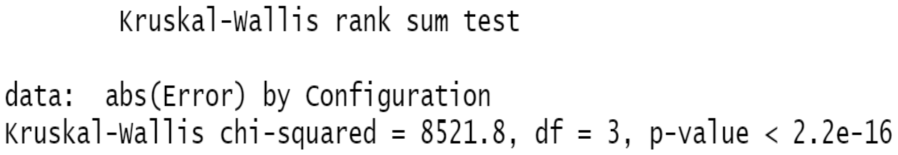


Figure 5.13: Graph between configurations and mean absolute errors for different strategies



```
                Kruskal-Wallis rank sum test

data:  abs(Error) by Configuration
Kruskal-Wallis chi-squared = 8521.8, df = 3, p-value < 2.2e-16
```

Figure 5.14: Results of the Kruskal-wallis test

Chapter 6

Conclusion and Future works

6.1 Conclusion

This thesis aims to determine the effect of context vectors on the task of authorship verification. Context vectors were used as feature in different configurations and strategies to find the relationship in which they act. Three ablation tests were performed namely, using every token, using top n tokens and using singleton tokens.

The PAN'20 dataset was used for the study and a data dump containing scraped data from `https://www.fanfiction.net` was used as the global corpus. Word embeddings were generated and were compared with word embeddings from a global reference corpus. Statistics from this similarity matrix generated after comparing the vectors are used to capture stylistic features. For e.g, the style of an author who uses a particular set of vocabulary or unique words in relation to each other can be captured by these statistics.

The generated features are then fed into a machine learning model, a decision tree in this case, to create a model and predict the similarity measures of pairs of unknown documents.

Statistical analysis on the results showed that the mean absolute error of the predicted measures is high in the baseline models, which do not use context vectors as a feature. The mean absolute error decreases after using context vectors with the top n tokens having the least mean error, followed by singleton tokens and every token respectively. It was also determined that Cosine similarity has a lower mean error than Euclidean distance measure across all the configurations and the Imposter strategy had lower mean absolute error than the Universum strategy. The kruskal-wallis test also gave a p-value less than 0.05, rejecting the null hypothesis and saying that there is significant difference between the different configurations of context vectors.

These analysis discussed in chapter 5, show the various ways different context vectors configurations affect the authorship verification task. Overall, context vectors help predict whether the document pairs are written by the same author or not positively, with the top n frequent words used by the author having the most significance and effect followed by singleton tokens (unique words) and every token (whole vocabulary) of an author.

6.2 Future Work

The context vector algorithm implemented in the research could be made more efficient in both space and time complexity. This is mostly due to the time taken to process the global corpus which is a few gigabytes in size. Each configuration run took at least 6 hours to run, so implementing a better and efficient algorithm would help greatly in the time taken. Determining a way to store the tokens in the global corpus with respect to their context rather than reading the global corpus every single time, would be hugely beneficial.

Genetic algorithm could also be used to obtain the best possible set of parameters to be used for context vectors. This would lead to a huge increase in the prediction accuracy in the AV task.

Bibliography

Fanfiction.net. URL https://www.fanfiction.net/. Accessed: 02-08-22.

Neeraj Agarwal. The ultimate guide to different word embedding techniques in nlp. https://www.kdnuggets.com/2021/11/guide-word-embedding-techniques-nlp.html, 2021. Accessed: 27-07-2022.

Jayesh Bapu Ahire. Introduction to word vectors. https://dzone.com/articles/introduction-to-word-vectors, 2021. Accessed: 27-07-2022.

Emir Araujo-Pino, Helena Gómez-Adorno, and Gibran Fuentes Pineda. Siamese network applied to authorship verification. In Linda Cappellato, Carsten Eickhoff, Nicola Ferro, and Aurélie Névéol, editors, *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*, volume 2696 of *CEUR Workshop Proceedings*. CEUR-WS, CEUR-WS.org, 2020. URL http://ceur-ws.org/Vol-2696/paper_222.pdf.

Douglas Bagnall. Authorship clustering using multi-headed recurrent neural networks. In Krisztian Balog, Linda Cappellato, Nicola Ferro, and Craig Macdonald, editors, *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016*, volume 1609 of *CEUR Workshop Proceedings*, pages 791–804. CEUR-WS, CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1609/16090791.pdf.

Georgios Barlas and Efstathios Stamatatos. Cross-domain authorship attribution using pre-trained language models. In Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 255–266, Cham, 2020. Springer International Publishing. ISBN 978-3-030-49161-1, organization=.

Ankit Bhakkad, Shweta Dharmadhikari, Emmanuel Mark, and Parag Kulkarni. E-vsm: Novel text representation model to capture contex-based closeness between two text documents booktitle=2013 7th International Conference on Intelligent Systems and Control (ISCO), organization=IEEE, isbn = 978-1-4673-4359-6, doi = 10.1109/ISCO.2013.6481176. pages 345–348, 01 2013.

Sebastian Bischoff, Niklas Deckers, Marcel Schliebs, Ben Thies, Matthias Hagen, Efstathios Stamatatos, Benno Stein, and Martin Potthast. The importance of suppressing domain style in authorship analysis. *ArXiv*, abs/2005.14714(1):1–14, 05 2020.

Benedikt T. Boenninghoff, Julian Rupp, Robert M. Nickel, and Dorothea Kolossa. Deep bayes factor scoring for authorship verification. *CoRR*, abs/2008.10105(1):1–14, 08 2020. URL `https://arxiv.org/abs/2008.10105`.

Benedikt T. Bönninghoff, Robert M. Nickel, and Dorothea Kolossa. O2D2: out-of-distribution detector to capture undecidable trials in authorship verification. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of *CEUR Workshop Proceedings*, pages 1846–1857. CEUR-WS, CEUR-WS.org, 2021. URL `http://ceur-ws.org/Vol-2936/paper-158.pdf`.

Daniel Castro-Castro, Yaritza Adame Arcia, María Peláez Brioso, and Rafael Muñoz. Authorship verification, combining linguistic features and different similarity functions. In Linda Cappellato, Nicola Ferro, Gareth J. F. Jones, and Eric SanJuan, editors, *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015*, volume 1391 of *CEUR Workshop Proceedings*, pages 1–7. CEUR-WS, CEUR-WS.org, 2015. URL `http://ceur-ws.org/Vol-1391/83-CR.pdf`.

Melesio Crespo-Sanchez, Helena Gómez-Adorno, Ivan Lopez-Arevalo, Edwin Aldana-Bobadilla, Karla Salas-Jimenez, and Jorge Cortes-Lopez. A content spectral-based analysis for authorship verification. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *CLEF 2022 Labs and Workshops, Notebook Papers*, pages 2416–2425. CEUR-WS, CEUR-WS.org, 2022. URL `http://ceur-ws.org/Vol-3180/paper-193.pdf`.

Pratap Dangeti. Cosine similarity. `https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/eb9cd609-e44a-40a2-9c3a-f16fc4f5289a.xhtml`, 2021. Accessed: 02-08-22.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

Kundan A. Dhande, Jayant S. Umale, and Parag A. Kulkarni. Context based text document sharing system using association rule mining. In *2014 Annual IEEE India Conference (INDICON)*, pages 1–6. IEEE, 2014. doi: 10.1109/INDICON.2014.7030458.

Daniel Embarcadero-Ruiz, Helena Gómez-Adorno, Alberto Embarcadero-Ruiz, and Gerardo Sierra. Graph-Based Siamese Network for Authorship Verification. *Mathematics*, 10(2):1–24, January 2022. doi: 10.3390/math10020277. URL `https://ideas.repec.org/a/gam/jmathe/v10y2022i2p277-d726262.html`.

Maël Fabien, Esau Villatoro-Tello, Petr Motlicek, and Shantipriya Parida. BertAA : BERT fine-tuning for authorship attribution. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 127–137, Indian Institute of Technology Patna, Patna, India, December 2020. NLPAI, NLP Association of India (NLPAI). URL `https://aclanthology.org/2020.icon-main.16`.

Jordan Frery, Christine Largeron, and Mihaela Juganaru-Mathieu. UJM at CLEF in Author Verification based on optimized classification trees. In *CLEF 2014*, page 7p., Sheffield, United Kingdom, September 2014. CEUR-WS. URL `https://hal-emse.ccsd.cnrs.fr/emse-01065829`. http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-FreryEt2014.pdf.

Lukasz Gagala. Authorship verification with prediction by partial matching and context-free grammar. In Linda Cappellato, Carsten Eickhoff, Nicola Ferro, and Aurélie Névéol, editors, *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*, volume 2696 of *CEUR Workshop Proceedings*. CEUR-WS, CEUR-WS.org, 2020. URL `http://ceur-ws.org/Vol-2696/paper_240.pdf`.

Jorge Alfonso Martinez Galicia, Daniel Embarcadero Ruiz, Alejandro Ríos Orduña, and Helena Gómez Adorno. Graph-based siamese network for authorship verification. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *CLEF 2022 Labs and Workshops, Notebook Papers*, pages 2594–2606. CEUR-WS, CEUR-WS.org, 2022. URL `http://ceur-ws.org/Vol-3180/paper-214.pdf`.

Stephen I. Gallant. Context vectors: A step toward a "grand unified representation". In *Hybrid Neural Systems, Revised Papers from a Workshop*, volume 1778, page 204–210, Berlin, Heidelberg, 1998. Springer, Springer-Verlag. ISBN 3540673059.

Antonio Menta Garuz and Ana García-Serrano. Authorship verification with neural networks via stylometric feature concatenation. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of *CEUR Workshop Proceedings*, pages 2064–2068. CEUR-WS, CEUR-WS.org, 2021. URL `http://ceur-ws.org/Vol-2936/paper-181.pdf`.

Tim Gollub, Martin Potthast, Anna Beyer, Matthias Busse, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. Recent trends in digital text forensics and its evaluation. In Pamela Forner, Henning Müller, Roberto Paredes, Paolo Rosso, and Benno Stein, editors, *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 282–302, Berlin, Heidelberg, 2013. Springer, Springer Berlin Heidelberg. ISBN 978-3-642-40802-1.

Oren Halvani, Lukas Graner, and Roey Regev. Cross-domain authorship verification based on topic agnostic features. In Linda Cappellato, Carsten Eickhoff, Nicola Ferro, and Aurélie Névéol, editors, *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*, volume 2696 of *CEUR Workshop Proceedings*. CEUR-WS, CEUR-WS.org, 2020. URL `http://ceur-ws.org/Vol-2696/paper_114.pdf`.

Josué Gerardo Gutiérrez Hernández, José Casillas, Paola Ledesma, Gibran Fuentes Pineda, and Iván Vladimir Meza Ruíz. Homotopy based classification for author verification task: Notebook for PAN at CLEF 2015. In Linda Cappellato, Nicola Ferro, Gareth J. F. Jones, and Eric SanJuan, editors, *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015*, volume 1391 of *CEUR Workshop Proceedings*, pages 138–143. CEUR-WS.org, 2015.

Mingjie Huang, Leilei Kong, Zeyang Peng, Yihui Ye, Zengyao Li, Xinyin Jiang, and Zhongyuan Han. Authorship verification based on fully interacted text segments. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *CLEF 2022 Labs and Workshops, Notebook Papers*, pages 2491–2495. CEUR-WS, CEUR-WS.org, 2022. URL `http://ceur-ws.org/Vol-3180/paper-201.pdf`.

Catherine Ikae. Unine at PAN-CLEF 2021: Authorship verification. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of

CEUR Workshop Proceedings, pages 1995–2003. CEUR-WS, CEUR-WS.org, 2021. URL `http://ceur-ws.org/Vol-2936/paper-173.pdf`.

Magdalena Jankowska, Vlado Keselj, and Evangelos E. Milios. Proximity based one-class classification with common n-gram dissimilarity for authorship verification task notebook for PAN at CLEF 2013. In Pamela Forner, Roberto Navigli, Dan Tufis, and Nicola Ferro, editors, *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013*, volume 1179 of *CEUR Workshop Proceedings*, pages 1–4. CEUR-WS, CEUR-WS.org, 2013. URL `http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-JankowskaEt2013.pdf`.

Patrick Juola. Authorship attribution. *Found. Trends Inf. Retr.*, 1(3):233–334, dec 2006. ISSN 1554-0669. doi: 10.1561/1500000005. URL `https://doi.org/10.1561/1500000005`.

Patrick Juola. An overview of the traditional authorship attribution subtask. In Pamela Forner, Jussi Karlgren, and Christa Womser-Hacker, editors, *CLEF (Online Working Notes/Labs/Workshop)*, volume 1178 of *CEUR Workshop Proceedings*, page 1. CEUR-WS, CEUR-WS.org, 2012. ISBN 978-88-904810-3-1. URL `http://dblp.uni-trier.de/db/conf/clef/clef2012w.html#Juola12`.

M. Kestemont, E. Manjavacas, I. Markov, J. Bevendorff, M. Wiegmann, E. Stamatatos, B. Stein, and M. Potthast. Overview of the cross-domain authorship verification task at pan 2021. In G. Faggioli, N. Ferro, A. Joly, M. Maistro, and F. Piroi, editors, *CLEF-WN 2021 - Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum*, volume 2936 of *CEUR Workshop Proceedings*, pages 1743–1759. CEUR-WS, CEUR-WS, 2021a.

Mike Kestemont, Enrique Manjavacas, Ilia Markov, Janek Bevendorff, Matti Wiegmann, Efstathios Stamatatos, Benno Stein, and Martin Potthast. Overview of the cross-domain authorship verification task at pan 2021. In *Working Notes of CLEF 2021 – Conference and Labs of the Evaluation Forum*, pages 372–383. CEUR-WS, 09 2021b.

Mahmoud Khonji and Youssef Iraqi. A slightly-modified gi-based author-verifier with lots of features (ASGALF). In Linda Cappellato, Nicola Ferro, Martin Halvey, and Wessel Kraaij, editors, *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014*, volume 1180 of *CEUR Workshop Proceedings*, pages 977–983. CEUR-WS, CEUR-WS.org, 2014. URL `http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-KonijEt2014.pdf`.

Alon Kipnis. Higher criticism as an unsupervised authorship discriminator. In Linda Cappellato, Carsten Eickhoff, Nicola Ferro, and Aurélie Névéol, editors, *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*, volume 2696 of *CEUR Workshop Proceedings*. CEUR-WS, CEUR-WS.org, 2020. URL `http://ceur-ws.org/Vol-2696/paper_228.pdf`.

Stefanos Konstantinou, Li, and Angelos Zinonos. Different encoding approaches for authorship verification. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *CLEF 2022 Labs and Workshops, Notebook Papers*, pages 2532–2540. CEUR-WS, CEUR-WS.org, 2022. URL `http://ceur-ws.org/Vol-3180/paper-206.pdf`.

Moshe Koppel and Yaron Winter. Determining if two documents are written by the same author. *J. Assoc. Inf. Sci. Technol.*, 65(1):178–187, jan 2014a. ISSN 2330-1635.

Moshe Koppel and Yaron Winter. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology*, 65 (1):178–187, 01 2014b. doi: 10.1002/asi.22954.

Moshe Koppel, Jonathan Schler, and Elisheva Bonchek-Dokow. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8(45): 1261–1276, 2007. URL `http://jmlr.org/papers/v8/koppel07a.html`.

Jakub Kruczek, Paulina Kruczek, and Marcin Kuta. Are n-gram categories helpful in text classification? In Valeria V. Krzhizhanovskaya, Gábor Závodszky, Michael H. Lees, Jack J. Dongarra, Peter M. A. Sloot, Sérgio Brissos, and João Teixeira, editors, *Computational Science – ICCS 2020*, pages 524–537, Cham, 2020. Springer, Springer International Publishing. ISBN 978-3-030-50417-5.

Anagha Kulkarni, Vrinda Tokekar, and Parag Kulkarni. Identifying context of text documents using naïve bayes classification and apriori association rule mining. In *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, pages 1–4. IEEE, 09 2012. ISBN 978-1-4673-2174-7. doi: 10.1109/CONSEG.2012.6349477.

Robert Layton, Paul A. Watters, and Richard Dazeley. Local n-grams for author identification notebook for PAN at CLEF 2013. In Pamela Forner, Roberto Navigli, Dan Tufis, and Nicola Ferro, editors, *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013*, volume 1179 of *CEUR Workshop Proceedings*, pages 1–4. CEUR-WS, CEUR-WS.org, 2013. URL `http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-LaytonEt2013.pdf`.

Li-Wei Lee and Shyi-Ming Chen. New methods for text categorization based on a new feature selection method and a new similarity measure between documents. In Moonis Ali and Richard Dapoigny, editors, *Advances in Applied Artificial Intelligence*, pages 1280–1289, Berlin, Heidelberg, 2006. Springer, Springer Berlin Heidelberg. ISBN 978-3-540-35454-3.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, pages 1–12. ICLR, 2013a. URL `http://arxiv.org/abs/1301.3781`.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781(1), 2013b. URL `http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781`.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013c. Microsoft, Association for Computational Linguistics. URL `https://aclanthology.org/N13-1090`.

Pashutan Modaresi and Philipp Gross. A language independent author verifier using fuzzy c-means clustering. In Linda Cappellato, Nicola Ferro, Martin Halvey, and Wessel Kraaij, editors, *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014*, volume 1180 of *CEUR Workshop Proceedings*, pages 1084–1091. CEUR-WS, CEUR-WS.org, 2014. URL `http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-ModaresiEt2014.pdf`.

Erwan Moreau and Carl Vogel. Style-based distance features for author verification notebook for PAN at CLEF 2013. In Pamela Forner, Roberto Navigli, Dan Tufis, and Nicola Ferro, editors, *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013*, volume 1179 of *CEUR Workshop Proceedings*, pages 1–5. CEUR-WS, CEUR-WS.org, 2013. URL `http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-MoreauEt2013.pdf`.

Erwan Moreau, Arun Jayapal, and Carl Vogel. Author Verification: Exploring a Large set of Parameters using a Genetic Algorithm - Notebook for PAN at CLEF 2014. In Linda Cappellato, Nicola Ferro, Martin Halvey, and Wessel Kraaij, editors, *Working Notes for CLEF 2014 Conference*, volume 1180, page 12, Sheffiled, United Kingdom, September 2014. CEUR-WS, CEUR Workshop Proceedings. URL `https://hal.archives-ouvertes.fr/hal-01076359`.

Erwan Moreau, Arun Jayapal, Gerard Lynch, and Carl Vogel. Author Verification: Basic Stacked Generalization Applied To Predictions from a Set of Heterogeneous Learners - Notebook for PAN at CLEF 2015. In Linda Cappellato, Nicola Ferro, Gareth J. F. Jones, and Eric SanJuan, editors, *CLEF 2015 - Conference and Labs of the Evaluation forum*, CEUR Workshop Proceedings, pages 1–12, Toulouse, France, September 2015. CEUR-WS, CEUR. URL `https://hal.archives-ouvertes.fr/hal-01226030`.

Maryam Najafi and Ehsan Tavan. Text-to-text transformer in authorship verification via stylistic and semantical analysis. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *CLEF 2022 Labs and Workshops, Notebook Papers*, pages 2607–2616. CEUR-WS, CEUR-WS.org, 2022. URL `http://ceur-ws.org/Vol-3180/paper-215.pdf`.

Zeyang Peng, Leilei Kong, Zhijie Zhang, Zhongyuan Han, and Xu Sun. Encoding text information by pre-trained model for authorship verification. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of *CEUR Workshop Proceedings*, pages 2103–2107. CEUR-WS, CEUR-WS.org, 2021. URL `http://ceur-ws.org/Vol-2936/paper-186.pdf`.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`.

Marina Pinzhakova, Tom Yagel, and Jakov Rabinovits. Feature similarity-based regression models for authorship verification. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of *CEUR Workshop Proceedings*, pages 2108–2117. CEUR-WS, CEUR-WS.org, 2021. URL `http://ceur-ws.org/Vol-2936/paper-187.pdf`.

Asef Poormasoomi, Mohsen Kahani, Saeed Varasteh Yazdi, and Hossein Kamyar. Context-based persian multi-document summarization (global view). In *Proceedings of the 2011 International Conference on Asian Language Processing*, IALP '11, page 145–149, USA, 2011. IEEE Computer Society. ISBN 9780769545547. doi: 10.1109/IALP.2011.53. URL `https://doi.org/10.1109/IALP.2011.53`.

Nektaria Potha and Efstathios Stamatatos. An improved impostors method for authorship verification. In Gareth J.F. Jones, Séamus Lawless, Julio Gonzalo, Liadh Kelly, Lorraine Goeuriot, Thomas Mandl, Linda Cappellato, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 138–144, Cham, 2017a. CEUR-WS, Springer International Publishing. ISBN 978-3-319-65813-1.

Nektaria Potha and Efstathios Stamatatos. An improved impostors method for authorship verification. pages 138–144. CEUR-WS booktitle= CLEF 2013 Evaluation Labs and Workshop - Online Working Notes, 08 2017b. ISBN 978-3-319-65812-4. doi: 10.1007/978-3-319-65813-1_14.

Francisco Rangel, Paolo Rosso, Moshe Koppel, Efstathios Stamatatos, and Giacomo Inches. Overview of the author profiling task at pan 2013. In *CLEF Conference on Multilingual and Multimodal Information Access Evaluation*, pages 352–365. CELCT, 2013.

Abhay B. Rathod, Sanjay M. Gulhane, and Shailesh R. Padalwar. A comparative study on distance measuring approches for permutation representations. In *2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT)*, pages 251–255. IEEE, 2016. doi: 10.1109/ICAECCT.2016.7942593.

Xin Rong. word2vec Parameter Learning Explained. *arXiv e-prints*, (1):arXiv:1411.2738, November 2014.

saed sayad. Decision tree - regression. `https://www.saedsayad.com/decision_tree_reg.htm`, 2020. Accessed: 04-08-22.

Sujan Kumar Saha, Satyam, Anand, and A.K. Dawn. *A Statistical Analysis Approach to Author Identification Using Latent Semantic Analysis*, pages 1143–1147. 09 2014. ISBN 1613-0073.

Tobias Schlagenhauf. Decision trees in python. `https://python-course.eu/machine-learning/decision-trees-in-python.php`, 2022. Accessed: 04-08-22.

Shachar Seidman. Authorship Verification Using the Impostors Method—Notebook for PAN at CLEF 2013. In Pamela Forner, Roberto Navigli, and Dan Tufis, editors, *CLEF*

2013 Evaluation Labs and Workshop – Working Notes Papers, 23-26 September, Valencia, Spain, pages 1–4. CEUR-WS, CEUR-WS.org, September 2013. ISBN 978-88-904810-3-1. URL `http://ceur-ws.org/Vol-1179`.

Dharmendra Sharma and Suresh Jain. Context-based weighting for vector space model to evaluate the relation between concept and context in information storage and retrieval system. In *2015 International Conference on Computer, Communication and Control (IC4)*, pages 1–5. IEEE, 2015. doi: 10.1109/IC4.2015.7375682.

Pulkit Sharma. Euclidean distance. `https://www.analyticsvidhya.com/blog/2020/02/4-types-of-distance-metrics-in-machine-learning/`, 2020. Accessed: 02-08-22.

Imran Sheikh, Irina Ulina, Dominique Fohr, and Georges Linarès. Document level semantic context for retrieving oov proper names. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6050–6054. ICASSP, 2016. doi: 10.1109/ICASSP.2016.7472839.

J.S. Sowmiya and S. Chandrakala. Joint sentiment/topic extraction from text. In *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pages 611–615, 2014. doi: 10.1109/ICACCCT.2014.7019160.

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Martin Potthast, B. Stein, Patrick Juola, Miguel Sanchez-Perez, and A. Barrón-Cedeño. Overview of the author identification task at pan 2014. *CEUR Workshop Proceedings*, 1180(1):877–897, 01 2014.

Efstathios Stamatatos, Martin Potthast, Francisco Rangel, Paolo Rosso, and Benno Stein. Overview of the pan/clef 2015 evaluation lab. In Josanne Mothe, Jacques Savoy, Jaap Kamps, Karen Pinel-Sauvagnat, Gareth Jones, Eric San Juan, Linda Capellato, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 518–538, Cham, 2015. CEUR-WS, Springer International Publishing. ISBN 978-3-319-24027-5.

Benno Stein, Nedim Lipka, and Peter Prettenhofer. Intrinsic plagiarism analysis. *Lang. Resour. Eval.*, 45(1):63–82, mar 2011. ISSN 1574-020X. doi: 10.1007/s10579-010-9115-y. URL `https://doi.org/10.1007/s10579-010-9115-y`.

u/nerdguy1138. Fan fiction data dump. `https://www.reddit.com/r/datasets/comments/b39dal/fanfic_just_all_of_it/`, 2019. Accessed: 09-08-2022.

Cor Veenman and David Tax. Less: A model-based classifier for sparse subspaces. *IEEE transactions on pattern analysis and machine intelligence*, 27(1):1496–500, 10 2005. doi: 10.1109/TPAMI.2005.182.

Cor J. Veenman and Zhenshi Li. Authorship verification with compression features. In Pamela Forner, Roberto Navigli, Dan Tufis, and Nicola Ferro, editors, *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013*, volume 1179 of *CEUR Workshop Proceedings*, pages 1–6. CEUR-WS, CEUR-WS.org, 2013. URL `http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-VeenmanEt2013.pdf`.

Carl Vogel, Gerard Lynch, and Jerom Janssen. *Universum Inference and Corpus Homogeneity*, pages 367–372. 01 2009. ISBN 978-1-84882-170-5. doi: 10.1007/978-1-84882-171-2_29.

Janith Weerasinghe, Rhia Singh, and Rachel Greenstadt. Feature vector difference based authorship verification for open-world settings. *CEUR Workshop Proceedings*, 2936: 2201–2207, 2021a. ISSN 1613-0073.

Janith Weerasinghe, Rhia Singh, and Rachel Greenstadt. Feature vector difference based authorship verification for open-world settings. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of *CEUR Workshop Proceedings*, pages 2201–2207. CEUR-WS, CEUR-WS.org, 2021b. URL `http://ceur-ws.org/Vol-2936/paper-197.pdf`.

Yihui Ye, Han Y, Zeyang Peng, Mingjie Huang, Leilei Kong, and Zhongyuan Han. Authorship verification using convolutional neural network. In Guglielmo Faggioli, Nicola Ferro, Alexis Joly, Maria Maistro, and Florina Piroi, editors, *CLEF 2022 Labs and Workshops, Notebook Papers*, pages 2607–2616. CEUR-WS, CEUR-WS.org, 2022. URL `http://ceur-ws.org/Vol-3180/paper-228.pdf`.

Hamed Zamani, Hossein Nasr Esfahani, Pariya Babaie, Samira Abnar, Mostafa Dehghani, and Azadeh Shakery. Authorship identification using dynamic selection of features from probabilistic feature set. In Evangelos Kanoulas, Mihai Lupu, Paul D. Clough, Mark Sanderson, Mark M. Hall, Allan Hanbury, and Elaine G. Toms, editors, *Information Access Evaluation. Multilinguality, Multimodality, and Interaction - 5th International Conference of the CLEF Initiative, CLEF 2014, Sheffield, UK, September 15-18, 2014. Proceedings*, volume 8685 of *Lecture Notes in Computer Science*, pages 128–140. CEUR-

WS, Springer, 2014. doi: 10.1007/978-3-319-11382-1_13. URL `https://doi.org/10.1007/978-3-319-11382-1_13`.

Appendix A

Context Vectors Source Code

```
def sort_dict_by_value(d, reverse = False):
    return dict(sorted(d.items(), key = lambda x: x[1],
        reverse = reverse))

def getFeatures(text1, text2, n):

    # remove punctuation from the texts
    processedText1 = re.sub(r'[^\w\s]', '', text1)
    processedText2 = re.sub(r'[^\w\s]', '', text2)

    #clean strings
    pat = re.compile(r'[^a-zA-Z ]+')
    processedText1 = re.sub(pat, '', processedText1).lower()
    processedText2 = re.sub(pat, '', processedText2).lower()

    tokens = set()
    # dict for keeping track of occurrences. to be used later in ranking.
    # hashFrequency = {}
    hashFrequency = OrderedDict()
    hashTokenRanking = {}
    hashRankingToken = {}
    nbTokensInText1 = 0

    splitText1 = processedText1.split(' ')
    for word in splitText1:
        tokens.add(word)
        if word in hashFrequency:
            hashFrequency[word] = hashFrequency[word] + 1
        else:
            hashFrequency[word] = 1

    nbTokensInText1 = len(tokens)

    # for adding missing words which are present in text
    # in cvectors of text 1. add word in token list
    splitText2 = processedText2.split(' ')
    for word in splitText2:
        tokens.add(word)
        if word not in hashFrequency:
            hashFrequency[word] = 0

    i = 1
    # sort hash frequency
    sortedHashFreq = sort_dict_by_value(hashFrequency, True)

    for token in sortedHashFreq:
        hashTokenRanking[token] = i
        hashRankingToken[i] = token
        i = i + 1

    #create context vectors
    after = {}
    before = {}

    for index, word in enumerate(splitText1):
        # word = splitText1[index]
        targetWordIndex = hashTokenRanking[word]



        # iterate from 1 to n (inclusive)
        for position in range(1, n+1):
            # for after
            if (index + position < len(splitText1)):
                contextWord = splitText1[index + position]
                contextWordIndex = hashTokenRanking[contextWord]

                if (targetWordIndex in after):
                    positionDict = after[targetWordIndex]
                    if (position in positionDict):
                        relFreqAtPositionJ =
                            after[targetWordIndex][position]
                        relFreqAtPositionJ[contextWordIndex - 1] =
                            relFreqAtPositionJ[contextWordIndex - 1] + 1
                    else:
                        #create an array
                        relFreqAtPositionJ = [0] * len(tokens)
                        relFreqAtPositionJ[contextWordIndex - 1] = 1
                        positionDict[position] = relFreqAtPositionJ

                else:
                    relFreqAtPositionJ = [0] * len(tokens)
                    relFreqAtPositionJ[contextWordIndex - 1] = 1

                    positionDict = {}
                    positionDict[position] = relFreqAtPositionJ

                    after[targetWordIndex] = positionDict

            # for before
            if (index - position >= 0):
                contextWord = splitText1[index - position]
                contextWordIndex = hashTokenRanking[contextWord]

                if (targetWordIndex in before):
                    positionDict = before[targetWordIndex]
                    if (position in positionDict):
                        relFreqAtPositionJ =
                            before[targetWordIndex][position]
                        relFreqAtPositionJ[contextWordIndex - 1] =
                            relFreqAtPositionJ[contextWordIndex - 1] + 1
                    else:
                        #create an array
                        relFreqAtPositionJ = [0] * len(tokens)
                        relFreqAtPositionJ[contextWordIndex - 1] = 1
                        positionDict[position] = relFreqAtPositionJ

                else:
                    relFreqAtPositionJ = [0] * len(tokens)
                    relFreqAtPositionJ[contextWordIndex - 1] = 1

                    positionDict = {}
                    positionDict[position] = relFreqAtPositionJ

                    before[targetWordIndex] = positionDict

    # process before and after for missing position vectors
    contextVectorHash = {}
    for rank in hashTokenRanking.values():
        contextVectorHash[rank] = []

        # start adding freq from before
        # n to 1 in before (so negative)
        for negativePos in range(n, 0, -1):
            # print(negativePos)
            if (rank not in before):
                contextVectorHash[rank] += [0] * len(tokens)
            else:
                if (negativePos not in before[rank]):
                    contextVectorHash[rank] += [0] * len(tokens)
                else:
                    contextVectorHash[rank] += before[rank][negativePos]

        for positivePos in range(1, n+1):
            if (rank not in after):
                contextVectorHash[rank] += [0] * len(tokens)
            else:
                if (positivePos not in after[rank]):
                    contextVectorHash[rank] += [0] * len(tokens)
                else:
                    contextVectorHash[rank] += after[rank][positivePos]

    for rank in hashTokenRanking.values():
        arrVector = np.asarray(contextVectorHash[rank])
        contextVectorHash[rank] = arrVector / float(nbTokensInText1)

    return contextVectorHash, hashFrequency, hashTokenRanking
```

Appendix B

Datasets

The Datasets used in the research can be found at,

PAN'20 Fanfiction Dataset, https://zenodo.org/record/5106099

The Data dump used for constructing the global corpus, https://archive.org/details/FanficRepack_Redux. This data was originally found at, https://www.reddit.com/r/datasets/comments/b39dal/fanfic_just_all_of_it/