Integrating State-of-the-art NLP Tools into Existing Methods to Address Current Challenges in Plagiarism Detection.

Victor Ughakepen Thompson

A thesis submitted in partial fulfilment of the requirements of the University of Sunderland for the degree of Doctor of Philosophy

June, 2023

Dedication

This thesis is dedicated to my beloved mother.

Acknowledgement

Special thanks to my family for all the support and encouragement through the ups and downs, to my supervisors Kenneth McGarry and Sardar Jafar for their excellent supervisory support and patience, and to God be the glory.

Abstract

Paraphrase plagiarism occurs when text is deliberately obfuscated to evade detection, deliberate alteration increases the complexity of plagiarism and the difficulty in detecting paraphrase plagiarism. In paraphrase plagiarism, copied texts often contain little or no matching words, and conventional plagiarism detectors, most of which are designed to detect matching stings are ineffective under such condition. The problem of plagiarism detection has been widely researched in recent years with significant progress made particularly in the platform of Pan@Clef competition on plagiarism detection. However further research is required specifically in the area of paraphrase and translation (obfuscation) plagiarism detection as studies show that the state-of-the-art is unsatisfactory. A rational solution to the problem is to apply models that detect plagiarism using semantic features in texts, rather than matching strings. Deep contextualised learning models (DCLMs) have the ability to learn deep textual features that can be used to compare text for semantic similarity. They have been remarkably effective in many natural language processing (NLP) tasks, but have not yet been tested in paraphrase plagiarism detection. The second problem facing conventional plagiarism detection is translation plagiarism, which occurs when copied text is translated to a different language and sometimes paraphrased and used without acknowledging the original sources. The most common method used for detecting cross-lingual plagiarism (CLP) require internet translation services, which is limiting to the detection process in many ways. A rational solution to the problem is to use detection models that do not utilise internet translation services. In this thesis we addressed these ongoing challenges facing conventional plagiarism detection by applying some of the most advanced methods in NLP, which includes contextualised and non-contextualised deep learning models. To address the problem of paraphrased plagiarism, we proposed a novel paraphrase plagiarism detector that integrates deep contextualised learning (DCL) into a generic plagiarism detection framework. Evaluation results revealed that our proposed paraphrase detector outperformed a state-of-art model, and a number of standard baselines in the task of paraphrase plagiarism detection. With respect to CLP detection, we propose a novel multilingual translation model (MTM) based on the Word2Vec (word embedding) model that can effectively translate text across a number of languages, it is independent of the internet and performs comparably, and in many cases better than a common cross-lingual plagiarism detection model that rely on online machine translator. The MTM does not require parallel or comparable corpora, it is therefore designed to resolve the problem of CLPD in low resource languages. The solutions provided in this research advance the state-of-the-art and contribute to the existing body of knowledge in plagiarism detection, and would also have a positive impact on academic integrity that has been under threat for a while by plagiarism.

Declaration of Authorship

I declare that this thesis is entirely my own work, and that it has not been submitted elsewhere for an award. All external information sources used to support this thesis were properly acknowledged.

TABLE OF CONTENTS

List of tables

List of figures

List of equations

List of software

1. Gensim
2. MySQL database
3. NLTK toolkits
4. NumPy
5. PyLucene
6. Python-programing language
7. Pytorch
8. R-programming language
9. SciPy
10. spaCy
11. Sentence transformer library
12. Weka-data mining

Abbreviations

AUC-ROC	Area under the receiver operating characteristics
BERT	Bidirectional encoder representations from transformers
Bhat	Bhattacharrya coefficient
CBOW	Continuous bag of words
CL	Cross-lingual
CL-ASA	Cross-lingual alignment-based similarity analysis
CL-CNGs	Cross-lingual Character N-Grams
CL-ESA	Cross-lingual explicit semantic analysis
CLM	Contextualised Learning Model
CLP	Cross-lingual plagiarism
CLPD	Cross lingual plagiarism detection
CoVe	Contextualised vectors
DCLM	Deep contextualised learning model
DL	Deep learning
DNN	Deep neural network
Doc2Vec	Document to vector
DPPD	Deep paraphrase plagiarism detector
ED	Euclidean distance
ELMo	Embeddings from language model
ESA	Explicit semantic analysis
GloVe	Global vectors
GPT	Generative pre-training
GST	Greedy string tiling
HSPD	Hybrid paraphrase plagiarism
HSSMM	Hybrid surface similarity measurement model
IC	Information content
IDF	Inverse document frequency
IDF	Inverse document frequency

IR	Information retrieval
JSD	Jensen-Shannon divergence
KLD	Kullback-Leibler divergence
KRGST	Karp-Rabin greedy string tiling
LCS	Longest common subsequence
LCSS	Longest common subsequence
LM	Language model
LSA	Latent semantic analysis
LSI	Latent semantic indexing
LSTM	Long short term memory
MLM	Mask language modelling
MSN	Multilingual semantic
MT	Machine translation
MT-LSTM	Machine translation LSTM
MTM	Multilingual translation model
NLP	Natural language processing
OOV	Out of vocabulary
P4P	Paraphrase for plagiarism
PCC	Pearson correlation coefficient
POS	Part of speech
PPD	Paraphrase plagiarism detection
PPDM	Paraphrase plagiarism detection model
RNN	Recurrent neural network
RoBERTa	Robustly optimised BERT pre-training approach
SI	Semantic interpreter
SOTA	State-of-the-art
SVD	Singular value decomposition
T + MA	Translation plus monolingual analysis
TF	Term frequency
TF-IDF	Term frequency inverse document frequency
TFISF	Term frequency inverse sentence frequency

VSM	Vector space model
Word2Vec	Word to Vector

1 Introduction

1.1 Background

This thesis presents new methods for external (extrinsic) plagiarism detection that combine state-of-the-art models used in natural language processing (NLP) with existing plagiarism detection methods to address important challenges facing conventional plagiarism detectors. External plagiarism detection involves searching for plagiarism in documents by making comparison with other (reference) documents (Potthast et al., 2013; Stamatatos et al., 2015; Foltýnek et al., 2019). Plagiarism is the act of using other people's work for one's own benefits without acknowledging the original authors (Meuschke and Gipp, 2013; Foltýnek et al., 2019). Plagiarism is a type of text reuse, and as with all text reuse there is always a clear similarity between plagiarised (copied) text and their sources (Gienapp et al., 2023). However it is worth pointing out that not all text reuse is plagiarism. The main difference lies in the intent of the user, plagiarism is often carried out with an intention to evade detection, while text reuse such as Journalistic text reuse (Clough et al., 2001) is done without any intentions to evade detection, but for plain journalistic purposes. Just because a pair of text are similar does not necessarily mean that plagiarism has taken place, the actual intent must be proven independently. Plagiarism can also occur when an author recycles (reuses) his previous work (self-plagiarism) without proper referencing (Krokoscz, 2021) or when someone's idea is used (idea plagiarism) without proper acknowledgement (Clough and Stevenson, 2011; Vani and Gupta, 2017).

Plagiarism is one of the most common types of academic misconducts, other types of misconducts related to plagiarism include contract writing (Draper et al., 2021), falsification and fabrication (Vaux, 2016). Contract writing involves using a third party to complete an academic work, it could take the form of ghost-writing or essay mill (Lancaster, 2020; Draper et al., 2021). Ghost writing involves outsourcing the writing of an academic work to someone else that is not one of the authors, this could be paid for or not paid for, while essay mills/banks are collection of already written essays that could be bought and used for ones' own benefit. Falsification and fabrication are more common in scientific research and involve presenting false (fabricated or manipulated) information (i.e. results) from a research work (Dal-Ré et al., 2020). Falsified and fabricated information are not reproducible and result in waste of resources (time, money and man-power). Falsification, fabrication and plagiarism are regarded as the worst types of academic misconduct (dishonesty) (Bülow and Helgesson, 2019), some have advocated criminalising plagiarism given the damage it causes to academic integrity (and to research domains such as in the field of medicine where human lives are involved), while others argue that criminalising one type of academic misconduct will only encourage other types of academic dishonesty that are not criminalised (Bülow and Helgesson, 2019). In general, some form of retribution should be put in place to deter all types of academic misconducts, and the degree of punishment should vary depending on the type of misconduct.

Plagiarism has been in existence for a long time, but the situation has been made worse by recent advancement in internet technology that allows for easy access and movement of large amount of information over the internet. Plagiarism is a problem because it discourages innovation and creativity, it also misrepresents the originality (efforts) of a submission (Perkins et al., 2020), and infringes on the ownership of an author (Clough et al., 2003; Mosco, 2021). Plagiarism is prevalent in academic environments (Hopp and Speil, 2021), and has a negative impact on academic integrity. Experts on plagiarism recommends a comprehensive approach to addressing the problem of plagiarism that includes prevention and detection. Plagiarism prevention involves using deterrent measures such as educating students on proper referencing and the consequences of being caught, using detection software to check assignments before submission, and other preventive measures to discourage students from plagiarizing (Halak and El-Hajjar, 2018; Foltýnek et al., 2019). This thesis focuses on plagiarism detection which could be carried out manually or automatically. Detecting plagiarism often require large amount of document comparison, and processing such amount of documents manually is not practical or suitable for real time plagiarism detection. Hence the use of automated systems that can quickly search large collections of documents to detect plagiarism is required. The main challenge facing conventional plagiarism detection is different obfuscation strategies used by plagiarist to evade detection (Potthast et al., 2013; Clough et al., 2015). Obfuscation is a technique used to disguise the act of plagiarism, it can take the form of lexical, semantic or syntactic changes to texts, such as replacing words with their synonyms (lexical substitutions), reordering a text passage, replacing phrases with their semantic equivalent, rewriting passages using other words (Clough et al., 2003; Mozgovoy et al., 2010; Clough and Stevenson, 2011; Alvi et al., 2021). Another challenging, but less common obfuscation strategy is technical disguise, which can occur when text is altered by substituting characters with visually identical characters (homoglyphs or foreign characters) or white space characters (Meuschke and Gipp, 2013, Alvi et al, 2017). Substituted characters are interpreted differently by computers and can alter an entire sequence of text leading to evasion of plagiarism. Technical disguise plagiarism could also take the form of replacing text with images/photos (of the replaced text), conventional detectors are rendered ineffective in such situations.

Studies reveal that paraphrase (disguise) plagiarism in the form of summary and translation obfuscation are ongoing challenges facing conventional plagiarism detectors (Meuschke and Gipp, 2013; Potthast et al., 2013, Meuschke et al., 2018). Summary obfuscation is characterised by semantic, syntactic and lexical alterations, while translation (or cross-lingual) obfuscation occurs when text expressed in one language is translated to another language and used without proper acknowledgement, translating text across languages often result in lexical and syntactic changes (Potthast et al., 2013; Clough et al., 2015; Dougherty, 2020). These changes are all features of paraphrase expressions. What makes paraphrase plagiarism difficult to detect is that in most cases, alterations carried out on copied texts often result in little or no overlapping features (e.g. lexicons) to be used as measurement parameters for similarity estimation. This research therefore proposed new plagiarism detection methods that addressed these challenging types of obfuscation plagiarism.

The methods proposed in this thesis are designed for monolingual and cross-lingual plagiarism detection; for monolingual plagiarism detection which involves searching for plagiarism by comparing documents of the same language, this thesis proposes a plagiarism detector that combines deep contextual learning with existing plagiarism detection tools to enhance the detection of paraphrased (obfuscated) plagiarism. In terms of cross-lingual plagiarism detection (CLPD) which involves searching for plagiarism by comparing documents of different languages, this research addresses a problem that is more likely to limit CLPD in underdeveloped nations, which is their reliance on internet translation services (such as Google and Microsoft translates) that is limited in many ways, notably: frequent disconnection and slow speed due to high traffic on servers, internet connections (network) not always available, not suitable (feasible) for large scale plagiarism detection given the large amount of translation required over the web etc. Most underdeveloped nations do surfer from poor internet and (or) electricity/power supply to establish steady and reliable connection to the cloud for plagiarism detection. To address this problem, this research proposes a multilingual translation model (MTM) based on a state-of-the-art neural word embedding model known as the Word2Vec model (Mikolov et al., 2013a; 2013b; 2017). The propose MTM is capable of translating words across languages with similar accuracy to an online machine translator, but without connecting to the internet, and given that the rise of cross lingual plagiarism is mainly due to easy access to common internet translation tools (Ehsan, 2016), the MTM is designed to reproduce the output of a common internet translation tool with good accuracy so as to improve precision in CLP detection.

While there are many standalone machine translation (MT) models, most of them are built for bilingual translation (Wu et al., 2017; Matusov, 2019) and with parallel corpora which are difficult to obtain or create in sufficient amount, especially for low resource languages. Hence MT (or neural machine translation (NMT)) models tend to perform poorly on low resource languages (Koehn and Knowles, 2017) and cannot be applied in multilingual settings. The proposed MTM is designed to be used for multilingual translation, and does not require parallel corpora, hence it can be used to detect CLP in multiple languages, and also on low resource languages which is still a challenge because of their lack of representation on the internet. Most NMT models also struggle with long sentences, and also on short sentences that lack context (Koehn and Knowles, 2017; Wan et al., 2022).

1.2 Research Questions and Objectives

Here are the questions addressed in this research;

1. What are the best performing combination of surface similarity measurement tools/techniques (as measured by precision, recall and F1-score) from those described in the literature for detecting similar and near similar text?
    - To determine the best combination of surface similarity measurement tools/techniques for detecting plagiarised text that have been obfuscated to varying degrees of complexity.

      Hypothesis: Since plagiarized text often contain fragments of unaltered texts that could link them to their sources, it is therefore possible that *with the right combination of tools used for detecting surface level similarity which includes similarity measures, ngram textual features and tem weight methods, different formation of plagiarized text could easily be detected, including cases with high degree of obfuscation.*

2. Can deep contextual learning models be used to enhance the detection of paraphrase plagiarism with performances comparable to, or even better than the current state-of-the-art (SOTA)?
   - To determine whether the application of a DCLM in paraphrased plagiarism detection could result in performances comparable to, or even better than the current state-of-the-art.

3. Can a multilingual translation model that is independent of internet translation services be built using a Word2Vec (word embedding) model and applied to effectively detect cross- lingual plagiarism (CLP) with performances comparable to a state-of-the-art CLPD model?
   - To determine whether a multilingual translation model can be built by leveraging the predictive power of word embedding (particularly the Word2Vec model) and applied in CLPD to achieve performances comparable to a state-of-the-art CLPD model (based on the T +MA model).
   - To determine whether a Word2Vec model could be trained to reproduce the output of an online machine translator and used in CLPD to produce similar performance to a commonly used CLPD model (T+ MA) which is dependent (and limited) by its reliance on internet translation services.

1.3 Research motivations

Plagiarism detection is a problem (challenge) that has received considerable attention particularly in the field of NLP and IR. While significant progress has been made in recent years (in plagiarism detection) more still need to be done in terms of improving performance on high obfuscation plagiarism detection.

Common methods used in detecting plagiarism are based on string (or lexical) matching and semantic similarity measurement methods (Foltýnek et al., 2019; 2020), in combination with text alignment. The string/lexical matching approach searches for overlapping strings in a pair of texts and merges nearby overlaps at close proximity (not more than certain distance apart) into a potential plagiarised passage (a process known as text alignment). This approach can be implemented using common surface tools such as surface similarity measures, ngrams (sequence of words or characters) and term weighing methods (Thompson et al., 2015; Sánchez-Vega et al., 2019), and has shown to be effective in detecting verbatim (cut and paste) and moderately altered plagiarism cases, but not effective in detecting heavy obfuscation plagiarism (with little or no overlapping features for alignment), such as in summary obfuscation plagiarism (Vani and Gupta, 2017). However, it is important to note that different surface similarity measurement tools capture different dimensions of surface textual features that can influence intertextual similarity in different ways, it would therefore be interesting to know how combining different textual features along these dimensions would perform in detecting plagiarised text with different formation of intertextual similarity ranging from verbatim to heavily altered cases.

In view of the above limitations of string matching methods, several attempts have been made to improve their performance by addressing a common strategy used in obfuscating texts called lexical substitution (Barrón-Cedeño et al., 2013; Sun and Yang, 2015; Alvi et al., 2021). These attempts basically involved the application of semantic methods, including the use of semantic relationships in a lexical database (i.e. WordNet) to detect words that were replaced with their synonyms in the act of plagiarism, the idea is to increase the amount of matching words for better alignment of texts and detection of plagiarism. While this approach seems promising in theory, in practice, it does not bring about significant improvement in performance (Ceska and Fox, 2011; Nawab et al, 2012), likely due to reliance on lexical databases (i.e. WordNet) which are limited in vocabulary size (Achananuparp and Shen, 2008; Manning, 2011; Álvarez-Carmona et al., 2018), and to words of only certain part-of-speech (POS: nouns, pronouns, verbs and adverbs). Semantic relationships such as path-length in a semantic network (i.e. WordNet) has also been applied in plagiarism detection to compute word level semantic similarity (Bär et al., 2012; Álvarez-Carmona et al., 2018; Sánchez-Vega et al., 2019). Vocabulary size limitation, as well as absence of contextualised information are limiting to this approach. Word embedding models such as the Word2Vec have also been used to detect plagiarism with promising results (Álvarez-Carmona et al., 2018; Alvi et al., 2021), however absence of contextualised information limits the effectiveness of such non-contextualised models.

High obfuscation plagiarism such as summary and paraphrase plagiarism are characterised by semantic, syntactic and lexical changes such as replacing phrases with single words (semantic equivalents), changing the order of a text passage, deleting and inserting words in a passage. These alterations are difficult to detect by conventional plagiarism detectors, and would require models that are designed to learn deep characteristics in texts, including semantic and syntactic patterns.

Recent advancement in deep learning (DL), particularly in contextualised (sequence) learning (McCanan et al., 2017; Ethayarajh, 2019) resulted in state-of-the-art performances in many NLP tasks including text classification, sentiment analysis, entailment, text summarisation etc. (Devlin et al., 2018; Reimers and Gurevych, 2019; Miller, 2019). Different deep contextual learning models (DCLM) such as ELMo (Embeddings from language model; Peters et al., 2017), Bert (Bidirectional encoder representation from transformers; Devlin et al., 2018), Generative pre-training (GPT; Radford et al., 2018; 2019) have been proposed in the literature. These models have different abilities in learning the context of a word (relationship between words in context), and can even capture long distance dependencies between words and sequences, which is particularly useful in comparing pairs of texts that are similar but appear lexically and syntactically different; a typical characteristics seen in paraphrase and summary obfuscation plagiarism. CLM are pre-trained on large datasets with a language model objective (next word prediction), and are designed to be fine-tuned to specific NLP tasks (for transfer learning) (Ethayarajh, 2019) with minimal architectural modification. DCLMs have the ability to learn deep semantic and syntactic features in texts, which can be used to estimate the semantic similarity between text sequences. The task of plagiarism detection can be reduced to searching for semantically similar texts in pairs of documents, a common practice is to make comparison at sentence level (Burrows et al., 2013; Farouk, 2019). Hence applying context learning models (seq2seq) to an existing plagiarism detection model would not require much modifications, given that a sentence is a sequence of words. The ease with which contextualised learning models fits into the plagiarism detection process, and their ability to extract deep semantic and syntactic features in texts make them promising for detecting obfuscation plagiarism. In addition, most of the NLP tasks which DCLMs have demonstrated remarkable effectiveness are similar to plagiarism detection as they all require the computation of semantic text similarity in order to achieve their individual objectives (Eneko et al., 2016; 2017). Given the differences in abilities of these models in context learning, it is therefore reasonable to investigate how well deep contextual learners from these two architectures would perform in terms of detecting high obfuscation plagiarism, their relative performances, and also their comparison to existing plagiarism detection tools (to the state-of-the-art).

The other challenge addressed in this research is the reliance of common CLP detectors on internet translation services that is limited in many ways as stated in section 1.1. The main challenge in CLPD arises from the fact that comparison is made between languages that do not have the same semantic, syntactic and lexical features for establishing similarity (via overlaps). Hence translating text across languages is necessary for many CLP detectors, although models based on statistical machine translation (MT) do not require such translations, but their reliance on parallel corpora which are often insufficient or not available for languages that are not well represented on the internet limits their use, MT methods are not well developed for efficient CLPD. The most common method used in detecting CLP is the translation monolingual analysis (T+MA) method (Barrón-Cedeño et al., 2013) which involves translating texts to a uniform language and applying a monolingual plagiarism detection method to detect plagiarism. Most implementation of the T+MA method require the use of internet translation tools, which are limited in many ways as stated earlier. To address this problem, this research experimented on building a language translation model (that is independent of the internet) by leveraging the learning and predictive capability of a Word2Vec (Mikolov et al., 2013) neural embedding model.

The Word2Vec model is a neural word representation model (word embedding model) that learns the context of a word with the objective of generating similar vector representations for words that occur frequently in the same context. Comparing vector representations using a similarity function such as the cosine measure usually results in similarity values that decreases progressively (from the maximum score:1, to the minimum score:0) with decrease in similarity of vector representations (1 represents perfect similarity, while 0-means completely dissimilar).

Considering the way the Word2Vec model works, it may be possible to modify the model to compute similar word representations to semantically similar words in different languages as opposed to non-semantically similar words, and apply the modified model in the T+MA model for translating texts across languages. How the resultant model will perform in comparison to the state-of-the-art in CLPD remains to be seen.

The next section outlines the contributions of this thesis to knowledge.

1.4 Research contributions

1. We proposed a paraphrase plagiarism detection model that integrates a deep contextualised learning model into an existing framework used in plagiarism detection to enhance the detection of paraphrase plagiarism. The proposed model is novel and original; unlike existing methods that either rely on surface features (i.e. string matching) or context independent semantic features (i.e. WordNet, Word2Vec), the proposed model advances the-state-of-the-art by using features deeply embedded in the context of texts to measure semantic similarity between text sequences, which is at the core of paraphrase plagiarism detection. The proposed model goes a long way to address the problem of paraphrase plagiarism which is one of the major challenges facing conventional plagiarism detection, and encourages future research in the application of DCLMs in plagiarism detection.

2. We proposed a cross lingual plagiarism detection (CLPD) model that uses a novel yet simple and effective multilingual translation model (MTM) for translating text across multiple languages in CLPD. The MTM is novel; it uses a Word2Vec model trained on a multilingual embedding space that maps semantically similar words in different languages into the same context, and can be directly used in candidate selection and text alignment in CLPD, which are two of the most important stages in plagiarism detection. The application of the MTM in the proposed CLPD model makes it novel and original. Furthermore, the MTM does not require parallel or comparable corpora to be trained, which as mentioned earlier, are difficult to obtain especially for low resource languages that are underrepresented in the internet, the proposed CLPD model can therefore be used for CLPD in low resource languages which is an ongoing challenge. The proposed model could also be used as standalone for detecting CLP in

underdeveloped countries where either internet connection and (or) electricity supply are unreliable to establish stable connection to the cloud for plagiarism detection.

3. Findings from our experiments revealed important details regarding specific paraphrased types (or expressions) embedded in plagiarised texts that could be detected by the application of DCLMs, and a few others that are challenging to detect; this finding is novel and could inform developers of where to focus their attention so as to build DCLMs with good quality representations for a wider range of downstream NLP tasks.

4. We proposed the best combinations of common surface similarity measurement techniques/tools for detecting plagiarised texts that have been obfuscated to varying degrees of complexity, in addition, findings from our experiments revealed specific characteristics of surface tools with respect to different complexity (features) in obfuscated text. The proposed combinations and findings are novel, and provide users with vital information about when best to use surface similarity measurement tools, what tools to use and the best combinations to accomplish specific textual similarity measurement tasks with optimal performance.

The next section contains overview of the remaining chapters in the thesis.

1.5 Overview of the Remaining Chapters

In order to answer the research questions (and address the above stated problems), several steps were taken which are broken down into chapters. Each chapter is a concise description of a step taken and how each step contributes to the end goal of this research. The rest of this thesis is organised as follows:

Chapter 2: reviews the relevant literature on plagiarism and plagiarism detection, including current challenges facing plagiarism detection such as paraphrase and translation obfuscation. The chapter discusses information retrieval (IR), natural language processing (NLP) and machine learning tools relevant to this research, including common IR and NLP techniques such as data pre-processing techniques, term weighting methods and ngram document models, intertextual similarity measurement techniques and tools used in the literature for computing lexical, semantic and syntactic similarity, and state-of-the-art deep learning models used for contextualised and non-contextualised learning, such as transformers, LSTMs and Word2Vec. The chapter also reviews current methods proposed in the literature for detecting cross lingual plagiarism, and relevant evaluation metrics used for plagiarism detection systems, and concludes with description of the research problems, and a number of tools relevant to the problems.

Chapter 3: covers specific tools, techniques and methods used in this research to address the problems identified in the literature. This chapter discusses the relevance of the chosen methods to the research problems, the specific way they were implemented in this research, and the rationale for the choices made compared to other relevant methods. The chapter describes the implementation of selected NLP tools and techniques such as text pre-processing techniques, n-gram models, and deep learning models, including contextualised and non-contextualised learning models. Also described are IR term weighting methods, candidate selection and document retrieval methods, and surface similarity measures. The chapter also describes the corpora and evaluation methods used in this research for building plagiarism detection systems and evaluating the performance.

Chapter 4: addresses the first research question, and contains experiments carried out to determine the best performing combinations of surface similarity measurement tools/techniques for detecting plagiarised text that have been obfuscated to different levels of complexity. The chapter describes evaluation of the performance of different combinations of common surface similarity measurement tools/techniques including term weighting methods, ngram document representation models and selected surface similarity measures.

Chapter 5: addresses research question two, which investigates whether DCLMs could be used to detect paraphrase plagiarism with performance comparable to, or better than a state-of-the-art model. The chapter describes experiments carried out to evaluate the performance of two common state-of-the-art DCLMs in the task of paraphrase plagiarism detection using corpora that contains paraphrase plagiarism and standard evaluation metrics. The chapter also discusses results obtained from comparative evaluation of the performance of the models against standard baselines and a state-of-the-art paraphrase plagiarism detection model.

Chapter 6: addresses the third research question, which investigates whether a language translation model that performs comparably to an online machine translator could be built and used in CLPD without connecting to internet translation services. The chapter contains evaluation of a proposed multilingual translation model (MTM) that uses a Word2Vec model that is trained to predict the semantic equivalent of a word in other languages with similar accuracy to an online machine translator. The chapter describes evaluation of the performance of a CLPD model (that uses the MTM) for CLPD without connecting to internet, using datasets that contain CLP and standard evaluation methods that involves performance comparison against baselines and previous studies. The chapter also describes experiments carried out to evaluate the performance of the proposed CLPD model on low resource languages.

Chapter 7: discusses the main findings of this research and relates the findings to the research questions and objectives, and to previous (related) research work on plagiarism detection. The chapter also discusses the relevance of the contributions of this research, the limitations, and proposes areas of future work.

1.6 Summary

This chapter introduces a research on external plagiarism detection that addressed existing problems on mono and cross lingual plagiarism detection. In regards to monolingual plagiarism, the problem addressed includes the poor performance of conventional plagiarism detector with respect to paraphrase plagiarism detection, which is an ongoing challenge. In terms of cross lingual plagiarism, the problem addressed is the reliance of common CLPD models on online machine translation services which is limiting in many ways (as stated in section 1.3). The solutions proposed involve using state-of-the-art methods in NLP such as contextualised and non-contextualised deep learning models relevant to the research problems, and several contributions were made to the body of knowledge in plagiarism detection as a result, see section 1.4 for details of the contributions made. This chapter concludes with an overview of the remaining chapters in the thesis.

The next chapter reviews the relevant literature on plagiarism detection.

2 Literature Review

2.1 Introduction

In the last chapter, the background of this research was briefly discussed, which includes the research questions, aims/objectives, contributions and motivations. This chapter reviews the relevant literature on plagiarism and plagiarism detection; the aim here is to review state-of-the-art information retrieval (IR) and natural language processing (NLP) methods and tools that have been proposed in the literature for addressing the problem of plagiarism detection, and related textual similarity measurement problems so as to identify the most suitable set of tools to build models for addressing the problems raised in this research.

The scope of this literature is limited to plagiarism detection in natural language (plain text), and does not include plagiarism in programming language. The literature covers plagiarism and common types/forms of plagiarism, plagiarism detection and common methods used in the literature to detect plagiarism (including newer methods based on word embeddngs), surface similarity measures and term weighting methods that have been successfully used in the literature to detect plagiarism and related textual similarity measurement tasks, cross-lingual plagiarism and common methods used in the literature to detect cross-lingual plagiarism (including newer methods based on graphs and word embeddings). The literature also covers challenges facing conventional plagiarism detection, and common methods used in the literature to evaluate plagiarism detection and classification systems.

The next section reviews the literature on common types/form of plagiarism.

2.2 Types/Forms of Plagiarism

Plagiarism is the use of other people's work without referencing the sources. There are different ways in which a text passage can be plagiarised, common forms of plagiarism includes; verbatim copy (copy and paste), paraphrase plagiarism and plagiarism of ideas (Vani and Gupta, 2017), technical disguise (Eisa et al., 2015; Velásquez et al., 2016; Gupta, 2016; Alvi et al., 2017; Foltýnek et al., 2019) and self-plagiarism (recycling) (Foltýnek et al., 2019; Gregory and Leeman, 2021; Krokoscz, 2021).

2.2.1 Verbatim Plagiarism (cut and paste)

In copy and paste plagiarism, large chunks of texts from one or more reference documents are lifted and used without any alterations to the copied text. This form of plagiarism does not require much effort from a plagiarist and can easily be detected using simple text matching techniques such as longest common subsequence and n-gram overlap (Clough and Stevenson, 2011; Oberteran, 2013). Verbatim plagiarism is the easiest to detect because they contain no alterations.

2.2.2 Paraphrase Plagiarism

In paraphrase plagiarism, plagiarist may alter one or more passages taking from one or more source documents before using them. The aim of the plagiarist in this case is to disguise the plagiarised passages in ways that would mislead a reader to believe that the suspicious document was solely written by the plagiarist. Popular alteration techniques used by plagiarist includes shuffling of words and sentences, complete removal of words and phrases, swapping of words with their synonyms etc. Paraphrase plagiarism is the most widely used by plagiarists, and varies in complexity. Clough and Stevenson described two types of paraphrases, they include word and sentence paraphrase. In word paraphrase, one or more words in a passage are replaced or re-written using other words, while sentence paraphrase involves re-writing one or more sentences in a passage in one’s own words before using the passage. Paraphrase plagiarism is an ongoing challenge (Foltýnek et al., 2019; Meuschke et al., 2019) to plagiarism detection, current detection systems are not well equipped to deal with the variety of paraphrase expressions found in texts, and hence their performance has been unsatisfactory.

A comprehensive study by Barrón-Cedeñoet al., (2013) revealed about 20 types of paraphrase expressions found in plagiarized text grouped into the following categories:

- Morphological changes: include all changes that affect the form in which a lexical unit appear in text, this include inflectional, derivational and modal verb changes.
- Lexical changes: include all changes that involves substituting one lexical unit with another, or altering the structure of a unit, changes under this category include spelling and format changes, converse, opposite-polarity, synthetic/analytic and same-polarity substitutions.
- Semantic changes: all changes that involves rewriting portion of text without semantic changes to the text.

- Syntax based changes: include all changes that affect the structure of texts such as moving lexical units around, they include coordination changes, subordination and nesting changes, ellipsis, negation switching and diathesis.
- Discourse changes: all changes that affect the style, format, or mode in which text is presented, they include punctuation and format changes, direct/indirect style alterations, sentence modality changes and syntax discourse structure changes.
- Miscellaneous changes: includes all change to text that involves reordering, insertion or deletion of one or more lexical units.

It is worth noting that paraphrase plagiarism is similar to paraphrasing in general as similar alteration techniques are typically employed, and they both retain semantic similarity between altered texts and their sources. However studies reveal that lexical substitution is by far the most common paraphrase technique found in plagiarized text (Barrón-Cedeñoet al., 2013), it is therefore important for this to be reflected in a paraphrase plagiarism corpus. In addition, unlike in general paraphrasing where a pair of text has to be semantically similar, in paraphrase plagiarism a pair of documents may not be semantically (or thematically) similar, but may contain one or more semantically similar passages.. Taking the above into account, it is reasonable to say that in the absence of paraphrase plagiarism corpus, a corpus created for the task of paraphrase identification could be used for training and evaluating paraphrase plagiarism detection models, especially when the only task is to compare a pair of text passages for semantic similarity, and when other important stages in plagiarism detection such as candidate document selection and post processing (that require details of actual plagiarized text and their sources (offset and length of copied passages)) are not required.

Given that paraphrase plagiarism is an ongoing challenge facing plagiarism detection, this thesis therefore explore the possibility of addressing paraphrase plagiarism using state-of-the-art NLP tools.

2.2.3 Idea Plagiarism

Plagiarism of idea involves representing other people’s ideas in one’s own words, contrary to the types of plagiarism described above, this form of plagiarism cannot be detected by matching texts in two documents as an idea could be expressed in entirely different words from their sources rendering conventional detection algorithms ineffective. Idea plagiarism is common in the academia, ideas from unpublished work, as well as unpublished methodologies could be plagiarised (Zimba and Gasparyan, 2021). Summary plagiarism to some extent is regarded as idea plagiarism (Vani and Gupta, 2017) as it involves high level of paraphrasing that render a plagiarised passage completely different from it source.

2.2.4 Technical Disguise Plagiarism

Technical disguise are obfuscation strategies carried out to render a plagiarised text unrecognizable to computational machines, common ones include the use of foreign characters, inserting white spaces in between the characters of a word, and submitting images of text in place of the actual text. A plagiarism detector designed for a particular language is unlikely to recognize foreign characters of other languages, for example a system designed to detect plagiarism in English text will be unable to recognize non-ascii characters. In addition, inserting white spaces in between characters in a word renders the word lexically unmatchable (undetectable). Research on plain text and computer code plagiarism detection have been well establish, image plagiarism detection is still in its infancy. Using images of text in place of the actual text renders most conventional plagiarism detectors ineffective.

Table 2.1: Types of Plagiarism, their Features and Examples

Plagiarism type	**Attribute/features**	**Example**	
		Source (original)text	*Suspect text*
Copy and paste (verbatim) plagiarism	Characterized by unaltered text sequences; cut from a source document and paste on suspect document without any form of alteration.	In object-oriented programming, inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined. The inheritance concept was invented in 1967 for Simula.	In object-oriented programming, inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined. The inheritance concept was invented in 1967 for Simula.
Paraphrase plagiarism	Characterised by lexical, semantic, syntactic or miscellaneous alterations to copied text. Form by removing text from a source document and altering it using different paraphrase techniques, followed by pasting the altered text in a suspect document without referencing the original source.	These classes have some of the behaviour and attributes which are in existent in the classes that it inherited from. The purpose of inheritance in object oriented programming is to minimize the reuse of existing code without modification.	The new classes, known as derived classes, take over (or inherit) attributes and behavior of the pre-existing classes, which are referred to as base classes (or ancestor classes). It is intended to help reuse existing code with little or no modification.
Plagiarism of ideas	Formed by rewriting ideas taken from a source passage using mostly different words and syntax so that the altered passage has no surface resemblance to its original but retain semantic similarity i.e. such as in summary plagiarism. Matching words are more likely to be name entities or technical terminologies that cannot be altered.	There are different types of paraphrase plagiarism, one common but not well known type is idea plagiarism which involves using other people's ideas without referencing them.	Applying concepts derived from external sources without acknowledge is an academic misconduct that's not often talked about.
Technical disguise	Characterized by foreign words and white characters that are not recognized by detection algorithm. Formed by inserting white or foreign characters in source text, and inserting the altered text into a suspect document.	The purpose of inheritance in object oriented programming is to minimize the reuse of existing code without modification.	The puŕpose of inheŕitance in óbject oriented βrogramming is to minimise the reūse of èxisting code without modification.

Table 2.1 contains summary of common plagiarism types identified in the literature, including brief description of their main features/characteristics and examples of how they may appear as passages in text documents.

The next section reviews plagiarism detection and common methods used in the literature to detect plagiarism.

2.3 Plagiarism Detection

Plagiarism detection is the task of searching for reused texts in a suspect document, in simpler terms, plagiarism detection involves searching for passages of texts that are similar in two documents using similarity measurement techniques (Meuschke and Gipp, 2013; Foltýnek et al., 2020). Given a suspicious document d_{SP} and a collection of source (reference) documents $D_{SC} = \{d_{SC1}, d_{SC2}.,.,...d_{SCn}\}$, the task of plagiarism detection is to find portions of plagiarised texts in the suspicious document d_{SP} that have been removed from one or more source documents D_{SC} by comparing d_{SP} with each source document in D_{SC}. Plagiarism detection can be carried out in natural language and programming language/source code (Devore-McDonald and Berger, 2020). This review is about plagiarism detection in natural language. There are basically two types of plagiarism analysis, they include external (extrinsic) and intrinsic plagiarism analysis (Oberreuter et al., 2011; Hagen et al., 2015; Potthast et al., 2019). External plagiarism analysis involves searching for plagiarism in a suspect document by comparing the suspect document with a reference document (Gupta, 2016; Foltýnek et al., 2019), while intrinsic plagiarism analysis involves searching for plagiarism in a suspect document using stylometric techniques to find variations or inconsistencies in writing style, it is usually carried out in the absence of reference documents, and very similar to authorship attribution as they both involve searching for inconsistencies in writing in a document (Stamatatos et al., 2016; Potthast et al., 2019). analysis is Intrinsic plagiarism was first introduced in Zu Eissen and Stein (2006) and has since become a widely accepted type of plagiarism analysis, and of the two types of plagiarism analysis, the external one is more common; it has been in existence much longer and has been well researched upon. Since this research is focused on external plagiarism detection, the discussion will mainly be focused on external plagiarism analysis. Much of the earlier research on external plagiarism detection involves searching for plagiarism in documents of the same language (monolingual), in more recent studies, the scope have been expanded to include searching for plagiarism in documents of different languages, also known as cross-lingual (translation) plagiarism detection (Barrón-Cedeno and Rosso, 2009; Potthast et al., 2011; Franco-Salvador et al., 2016; Stegmüller et al., 2020). Cross lingual plagiarism (CLP) occurs when text original expressed in a given language is translated into a different language and used without properly referencing the original source. Searching for CLP usually involves translating texts to their original sources using a language translation tool such as Google translate and applying a standard monolingual detection method to detect plagiarized passages. This research focuses on improving both mono and cross lingual plagiarism detection.

2.4 Monolingual Plagiarism Detection

Monolingual plagiarism (MLP) detection involves searching for plagiarism by comparing texts of the same language. The task of detecting plagiarism in text documents can be divided into three subtasks, namely data pre-processing, candidate source retrieval and exhaustive document comparison and post-processing [text alignment] (Stamatatos et al., 2015; Gupta, 2016; Hourrane and Benlahmar, 2017; Potthast et al., 2019). Figure 2.1 is schematic of a traditional plagiarism detection system showing the main stages involved.

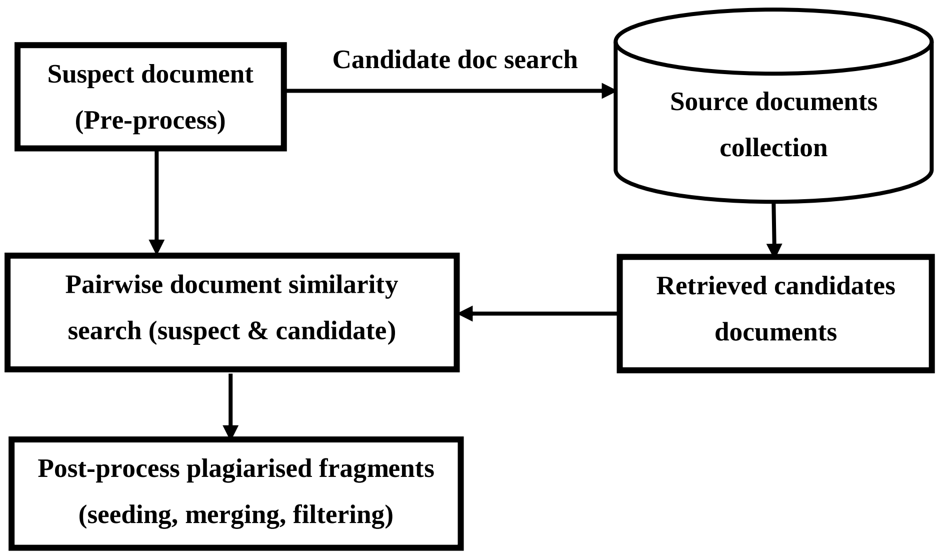

Suspect document (Pre-process)
Candidate doc search
Source documents collection
Pairwise document similarity search (suspect & candidate)
Retrieved candidates documents
Post-process plagiarised fragments (seeding, merging, filtering)

Figure 2.1: Schematics of a generic plagiarism detection system showing the main processes involved in plagiarism detection.

The diagram in fig 2.1 shows the stages involved in plagiarism detection following the introduction of suspect document. The first stage is data pre-processing which involves using NLP techniques to transform a suspect and a collection of source documents into uniform comparable format, and to remove noisy unwanted elements that may interfere with the detection process. The second stage is a typical information retrieval (IR) stage known as candidate selection, and involves (transforming the pre-processed source document) using the pre-processed suspect documents as query to retrieve a few number of source documents as candidates from the source document repository; these are usually documents that contain significant amount of query terms. A vital part of candidate selection is query formulation; queries have to be formulated so as to increase the chance of retrieving potential plagiarised candidates with high accuracy and at a low computational cost, a common technique is to use sequence of text (ie. words/characters/ngrams) as queries, the rationale is that significant amount of matching sequence between a suspect and a source document is more indicative of plagiarism than single words, see the section on candidate selection for more on query formulation. The next stage is pairwise document comparison between the suspect document and each candidate, this process involves searching for plagiarised text fragments by comparing fragments of text in the pair for similarity using a similarity function (i.e. string (e.g. cosine measure) or semantic based). The final stage is post-processing (seeding, merging and filtering) which involves retrieving matching fragments (seeding), merging nearby fragments at close proximity into plagiarised passages, and filtering off all passages that do not satisfy certain conditions, such as size (length). These stages are explained in more details in the literature below:

2.4.1 Data Preprocessing

Data pre-processing removes noisy and unwanted features from texts that may interfere with a comparison algorithm, it also presents documents in uniform comparable format. Typical pre-processing steps used in NLP and IR include tokenization, stop-word removal, case normalization (all characters are nomalised to the same case) and stemming (Chong et al., 2010; Gupta, 2016; Foltýnek et al., 2019). Tokenization parses documents into individual words; in effect a document is broken down into bag-of-words (index terms) enabling detailed document comparison to be carried using individual terms as features. The word '*term*' is used to describe a word or sequence of words (or characters) in a document. Stop-words are words with low discriminating power usually present in most documents (e.g. the, she, who etc.); they are misleading to a comparison algorithm and increase chances of false matches, hence they are often removed prior to document comparison. There are arguments about whether stop-words should be removed or not before document comparison, such as when stop-words are the primary discriminators and removing them will result in error or false positives (Turney, 2010). Hence some experts are against the removal of stop-words. One notable effect of stop-word removal is that it speeds up comparison (Saiyed and Sajja, 2022) because it results in fewer words to be compared. Case normalization (case folding) is often carried to normalize all characters to the same alphabetic case; upper or lower case. Case folding (normalization) helps in eliminating discrepancies (Alvi, 2020) that may arise when a term co-occur in a pair of text in different cases (such as 'Fan' and 'fan'), a term that occurs in different cases is considered different by a comparison algorithm, however there situations where it is best not to case normalize, especially when dealing with abbreviations as it could result in false matches, for example the abbreviation ‘USA’ and a person’s name; ‘Usa’ would result in a false match if they are both nomalised to the same alphabetic case. Stemming reduces words to their root form and increases chances of overlaps (for example: ‘friendly’, ‘friendship’, ‘friend’ are all reduced to ‘friend’). Stemming can increase efficiency and recall of a comparison algorithm (Ceska and Fox, 2011). However, there are times when it is best to leave words in the original form in which they occur in texts, as they may be more discriminatory that way.

2.4.2 Candidate Source Document Retrieval

Candidate source documents are potential documents from a collection of source/reference documents from which one or more text passages have been removed and used for plagiarism. Candidate retrieval in plagiarism detection is a typical IR task similar to web search (Potthast et al., 2013; Vani and Gupta, 2016; Foltýnek et al., 2019). To ensure efficiency when dealing with large document collections, a search engine (such as Apache Lucene) may be required to, otherwise a simple database is fine for small document collections. Candidate retrieval involves querying a database of source documents and retrieving a handful of potential candidate documents using IR techniques such as keyword search and document ranking. The aim of candidate retrieval is to reduce the search space of the subsequent plagiarism detection phase, which is computationally intensive (Potthast et al., 2014; Vani and Gupta, 2016; Meuschke et al., 2018; Foltýnek et al., 2019). Candidate retrieval is not necessary on small source document collections, applying candidate selection in such case would not result in any significant performance gain. The main component of a candidate retrieval system includes a query and a search engine (an inverted index data structure and a ranking method). In the context of plagiarism detection, a query is a suspect document, a source document collection is stored in an inverted index structure in a database/file, and ranking ensures only the most relevant documents to a query are retrieved as candidates.

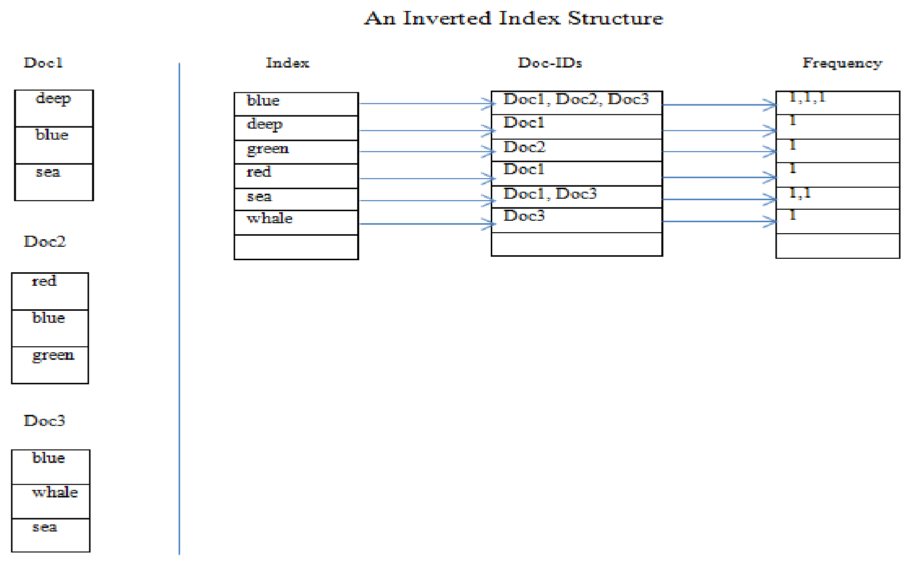


Figure 2.2: A Simple Inverted Index Data Storage Structure

Fig 2.2 is an inverted index table built with terms in *doc1, doc2 and doc3*. The table itself comprises of three columns namely; index, Document-ID and term frequency. To retrieve all documents that contain the word *sea*, simply query the table with the word *sea* and all documents that contain the word sea will be retrieved, in this case, Doc1 and Doc3 will be retrieved along with their term frequencies.

The main objective of a candidate retrieval system in plagiarism detection is to retrieve plagiarised sources at a low retrieval cost (Potthast et al., 2013; 2014; Kong et al., 2019). Keyword/phrase selection and query formulation is quite important to ensure good performance, using all keywords/phrases as queries for candidate retrieval is quite expensive and may defeat the objective of candidate retrieval. A proper query selection methods is necessary to ensure that the most relevant plagiarised sources are retrieved at minimal computational cost. A common method is to select keywords using term frequency inverse document frequency (TFIDF: Robertson, 2004) relevance weighting scheme. Kong et al., (2012) used only terms with high TFIDF as queries to retrieve candidate documents. Ravi and Gupta (2015) combined POS tagging and TFIDF to select keywords for candidate retrieval. While using IR methods such as TFIDF could result in the retrieval of relevant documents, it is important to note that relevance in IR is defined in terms of topic similarity, but a pair of documents (query and candidate) of similar topic (with similar terms) may not necessary be plagiarized. To increase the chances of retrieving potential plagiarized documents, candidate selection in plagiarism detection focusses more on query formulation, which is the creation of queries from suspect text to maximize the retrieval of plagiarized documents. One of the best indicators of plagiarism is matching sequence of text, and the longer such matching sequences (ie n-grams), the higher the chances of plagiarism (Thompson et al., 2015). This is clear from previous studies, for example Grozea and Popescu (2012) applied character sequence (16-grams) to detect plagiarism and proposed encoplot (Grozea and Popescu 2012; Amzuloiu et al., 2021) which emerged as the best performing system in the Pan@clef 2012 (Potthast et al., 2012) on plagiarism, similarly Stamatatos (2011) used stop-word ngram (size=8) as query to retrieve candidate source documents. In a more recent study, Kong et al., (2019) reiterated a fact about the lack of guarantee that candidate documents retrieved from a search engine would be true plagiarised sources given the heuristic nature of query formulation and aggregation of query results, and proposed a method that uses logistic regression to ensure queries from a suspect document closely correlate for better aggregation of query results. Candidate retrieval is one of the most important stages in plagiarism detection, poor execution of this stage could result in potential plagiarised source being left from the subsequent stages. High recall is therefore essential, but at a low cost (Hagen et al., 2015). When potential source documents are retrieved, the next stage is to search and extract plagiarized text passages and their respective sources from suspect and source documents using text alignment techniques.

2.4.3 Exhaustive Document Comparison and Plagiarism Detection

This subsection is divided into two, the first part reviews state-of-the-art methods used in the literature to detect plagiarised texts, and the second part reviews methods that are used for post-processing (seeding, merging and filtering). Common methods used in the literature to detect plagiarised texts are: string (lexical) matching, syntactic and semantic similarity detection methods (Eisa et al., 2015; Gupta, 2016; Foltýnek et al., 2019). Each method comprises of a number of techniques designed to compare texts for similarity, details of these methods and their respective techniques are described below.

2.4.3.1 Lexical (String) Matching Method

This method comprises of all plagiarism detection approaches that use string overlaps to determine intertextual similarity. Approaches under this category are ngram overlap, Vector space method, fingerprinting (Gupta, 2016; HaCohen-Kerner and Tayeb, 2017; Foltýnek et al., 2019) and Karp Rabin Greedy Sting Tilling (Clough et al., 2002; Jayapal, 2012; Alvi et al., 2021). Details of these approaches are described below.

2.4.3.2 The Vector Space Document Ranking Approach

A vector space document ranking model is any information retrieval model that represents documents as vectors and ranks them based on their similarity to a query vector. The vector space model (VSM) is the most common IR model used for ranking documents based on relevance to a user's information need (Turney, 2010; Marcos-Pablos and García-Peñalvo, 2020). In the VSM approach, documents and queries are represented as vectors in a multi-dimensional space, the queries are then compared with the document vectors (pairwise) using a similarity function (cosine similarity in many cases) and the documents are ranked by decreasing order of similarity based on their similarity scores. The intuition here in terms of plagiarism detection is that, higher ranking documents are more similar to a query, and are considered likely plagiarised sources. However, unlike in IR where relevance is based on matching terms between queries and documents, in plagiarism detection relevance is based on matching features (e.g. substrings), and not limited to terms alone.

When implementing the VSM, text must first be transformed to vectors by tokenization to either characters, words or sentences, and weights are assigned to tokens/terms. Term weighting is commonly done by assigning weights to terms either by their term frequency (TF) count, term frequency inverse document frequency (TFIDF), which is a relevant weighting scheme that assigns higher weights to relevant terms, where relevance depends on the rarity of a word in a document as opposed to a collection. Binary weighing is also common and involves assigning equal weights to all terms present in a document, and a value of zero to terms not present. Details of these common term weighing methods are described in section 2.5.3. In NLP, newer methods have been proposed for vectorising texts by transforming words in text sequences to vectors of real numbers, and aggregating word vectors into a single vector representation for a text sequence. The two most common NLP methods used in vectorising words are one hot vector encoding and word embeddings (Salim and Mustafa, 2022; Lauriola et al., 2022). In one hot vector encodings, words are transformed to vectors using a binary system that assigns a value of 1 to represent the position of a word in the vocabulary of a model, and zero to all other dimension of the vector. The problem with one hot vector encoding is that it can easily result in extremely large vectors with length equal to the size of the vocabulary of a model, it also ignores vital contextualised information that captures the real meaning of words in context, and results in orthogonal vectors (perpendicular with dot product of zero) for different terms (Lauriola et al., 2022). Below is an example of how a sequence of text could be encoded using a one hot vector encoding;

The text sequence: *'one hot vector encoding'* is encoded as follows:

Vocabulary size

one	[0, 0,1,0,….0]
hot	[0,1,0,0,….0]
vector	[0,0,0,0,….1]
encoding	[1,0,0,0,….0]

one **hot** **vector** **encoding**

[[0,0,1,0,….0], [0,1,0,0,….0], [0,0,0,0,….1], [1,0,0,0,….0]]

Word embeddings on the other hand addressed most of the problems with one hot vector encoding by taking into account contextualized information using a fixed size window to capture the context of words, and specifying fixed size length for all vectors irrespective of vocabulary size.

For a window size of four word sequence, the above text sequence could be represented using word embeddings as follows:

window size capturing context

one	[1,0,0,0]
hot	[0,1,0,0]
vector	[0,0,1,0]
encoding	[0,0.0,1]

One hot vector encoding

[[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,0,1]]

The above example shows word representations using word embeddings to capture the context of the text sequence in a window size of four word sequence.

The VSM in combination with term weighing and similarity function are often used for both candidate retrieval and detection of plagiarised texts (exhaustive comparison stage). Zechner et al., (2009), Ekbal et al., (2012), Kong et al., (2013) and Vani and Gupta (2017 applied the VSM and TFIDF weighting to rank and retrieve candidate source documents. Sánchez-Vega-Perez et al., (2014) applied the VSM and TFISF (term frequency inverse sentence frequency) with a combination of Cosine and Dice similarity measures for candidate retrieval and exhaustive comparison for plagiarism detection at sentence level. Ehsan and Shakery (2016) applied the VSM and binary weighting at the exhaustive comparison stage to detect plagiarised passages.

The VSM is very efficient and captures some degree of semantic similarity, hence its wide application for candidate document retrieval in plagiarism detection.

2.4.3.2.1 The N-gram Overlap Approach

One other technique that has been successfully used for document similarity analysis and plagiarism detection is n-gram overlap. The idea is to use the proportion of overlapping n-grams (sequence of words or characters) in a pair of texts as a measure of similarity. Similarity measures are usually used to transform the proportion of overlaps into similarity scores, commonly used ones include Dice coefficient, Jaccard index and containment (overlap) measure. Some previous research where this approach was used for document similarity analysis and plagiarism detection are (Clough, 2002; Clough and Stevenson, 2011; Sánchez-Vega et al., 2019; Bensalem et al., 2019).

Lyon et al., (2004) built a moderate scale plagiarism detection system based on shared word-trigrams and Jaccard index and obtained promising results that became standard baselines for plagiarism detection systems. Lyon et al., argue that trigrams are best for measuring document similarity, and that ngrams longer than three word sequence are likely to bypass shorter overlaps (plagiarised sections), while ngrams shorter than 3-grams are unlikely to capture long plagiarized fragments of varying sizes. Containment measure is based on set theory, it measures how much of a document is contained in another. The implementation of n-gram overlap method using containment is carried out by transforming texts to set of n-grams (i.e. set A and set B), and computing similarity as the intersection of sets *A*, *B*, normalized by the length of *A*, where *A* is the suspect document. The output is in the range of 0 and 1.

$$cont.(A,B) = \frac{(A \cap B)}{len(A)}, \qquad \text{Equation 2.1}$$

Where *A* and *B* are two documents represented as set of unique ngrams

Barrón-Cedeño and Rose (2009) proposed an approach for detecting plagiarised texts that have been altered by reordering using n-gram overlap and the containment measure, and sentence level comparison, experimented results from Barrón-Cedeño and Rose study revealed that 2-3-grams are the best ngram sizes for detecting reordered plagiarised texts.

Clough et al., (2002) used n-gram overlap method to measure journalistic text reuse and obtained encouraging results. Clough and Stevenson (2011) also applied n-gram overlap to detect plagiarized texts that have been altered to different levels (degree) of obfuscation. In more recent study, Sánchez-Vega et al., (2019) applied character ngrams to detect similarity in writing style and content in paraphrased plagiarised text and outperformed a number of baselines including a knowledge based model (based on the application of WordNet). In addition, Sánchez-Vega et al., proposed ngrams of size 3-4 as best for detecting paraphrase plagiarism.

Both word and character ngrams have been used to detect plagiarism, word ngrams are more efficient and are very effective for detecting verbatim plagiarism. Character ngrams are able detect similarity in writing style and content, spelling errors and morphological changes via substring matching (Liao et al., 2017; Sánchez-Vega et al., 2019). N-gram overlap method is quite effective for verbatim (word for word) similarity analysis (Eisa et al., 2015; Bensalem et al., 2019), easy to implement and quite efficient for comparing documents. However, one drawback of the n-gram overlap method is that it is ineffective for uncovering high obfuscation plagiarism when there is little or no overlapping strings.

2.4.3.2.2 The Fingerprinting Approach

Fingerprinting was first used to address the challenge of identifying similar files in a large file system. Fingerprinting is one of the most common approaches used for plagiarism detection, the idea is to represent documents as fingerprints, and use the amount of overlaps in their fingerprints as a measure of similarity. Overlapping sections of fingerprints indicate areas of copy (or plagiarism). In practice, the fingerprinting approach could be implemented in the following steps;

- Document pre-processing: this step removes unwanted noisy elements and transforms documents into comparable formats; see section 2.5.1 for details about common NLP data pre-processing steps.
- Transformation to substrings: this step transforms the pre-processed documents into substrings such as sequence of characters or words of a specific length, or even whole sentences. The specific size or granularity of substrings has a direct bearing on the effectiveness of fingerprinting; the larger the granularity, the more efficient and less effective the comparison process will be, and the smaller the granularity, the less efficient and more effective the process will be. Hence finding the right substring size (granularity) is crucial. Experimental results on fingerprinting strategies by Hoard and Zobel (2003) revealed that grain sizes between 3 and 5-word sequences are the best substring sizes for an effective implementation of fingerprinting.
- Substring selection: to ensure efficiency in document fingerprint comparison, it is vital to retain only the most informative substrings for fingerprinting generation. However, the fingerprinting approach is most effective when all substrings are retained (full resolution) and used for document comparison, as all information will be used in document comparison. Full substring selection comes with the cost of large storage requirement, and low efficiency during comparison.
- Substrings encoding (Hashing): this step involves using a hash function (such as MD5) to assign hash values/codes (minutia) to the selected substrings. The most important thing to note here is that each unique substring in a document collection must be assigned a unique hash code to avoid collision, which can occurs when two or more substrings share the same hash code.
- Indexing of fingerprints: this step involves indexing the collection’s fingerprints in a database for quick querying and retrieval of similar texts and their respective document IDs.

Many studies on plagiarism, duplicates and near duplicates detection (Manku et al; 2003; Henzinger, 2006) are based on fingerprinting; this is because of its relative high efficiency on large data collections. However, it is worth noting that fingerprinting itself does not necessarily bring about improvement in accuracy of detection, its main benefit is efficiency in plagiarism detection. One major drawback of the fingerprinting technique is that in practice not all chunks (substrings) are used in fingerprint generation as this can result in reduction in efficiency and ultimately defeat the very essence of using fingerprints for plagiarism detection. To completely avoid information loss and achieve the best outcome, full document fingerprints would have to be used. However, this will require large storage space and memory to store documents' fingerprints, and to compare documents fingerprints efficiently in-memory. Hence the question of which chunk to retain and which to discard remains an issue as important chunks that may overlap with other documents may easily be discarded which can result in inaccuracy in documents comparison (Metzler et al., 2005).

The application of fingerprinting is not very common in the current literature of plagiarism detection, however in a recent study, HaCohen-Kerner and Tayeb (2017) proposed a fingerprinting model that uses random parameters to rapidly detect similar scientific documents in a collection with encouraging results.

2.4.3.2.3 Karp-Rabin Greedy String Tiling (KRGST)

The Karp-Rabin Greedy String Tiling (KRGST) is a matching algorithm that searches for the longest common substring between two strings. The KRGST is commonly used in DNA sequence alignment (Wise, 1993) and for source code plagiarism detection (Agrawal and Sharma, 2016 Foltýnek et al., 2020), and more recently in plagiarism detection in natural language (Clough et al., 2002; Jayapal, 2012; Alvi et al., 2021). Unlike similar algorithms such as the Levenshtein distance or the longest common subsequence, the KRGST addresses the problem of transposition in string alignment. Transposition occurs when the original order of a string or sentence is changed or when one or more strings or tokens are out of place. The basic idea of the greedy string tiling is to capture matched substrings (known as tiles) between two strings, and extend the matched substrings to maximum length using the Karp-Rabin algorithm.

The Karp-Rabin algorithm takes a string of a specific length, and searches in a document for strings of similar length, when it finds one, it compares the two strings character by character and continues until there is a character mismatch. The algorithm then marks and isolates the matched string from subsequent comparison. One major drawback of the KRGST is that it ignores short matching substrings, and the implication of this in plagiarism detection is that highly altered cases of plagiarism with short plagiarised fragments will evade the KRGST algorithm. In addition, KRGST rely on matching stings to function, and in the absence of such strings, such as in high obfuscation plagiarism where a plagiarised text is altered so that there is little or no matching strings between the text and its source (original), the application of GST will likely fail in such situation because of the absence of matching strings (or tiles).

Jayapal (2012) applied the KRGST with varying tile length in aligning plagiarised texts in Pan@Clef 2012 text alignment competition. However the results obtained was not impressive, after having had the best performance in the source retrieval task. GST was also used in (Clough et al., 2002) for detecting Journalistic text reuse; however the use of GST in plagiarism detection has mostly been focused on the detection of plagiarism in programming code. In more recent study Alvi et al., (2021) applied GST to similar text fragments in a pair of sentences for paraphrase type detection.

2.4.3.2.4 Semantic Similarity Measurement Methods

One of the most difficult types of plagiarism to detect is obfuscation plagiarism, this is largely due to semantic, syntactic and lexical changes in obfuscated text that evade detection. Obfuscation as mentioned earlier is used to disguise plagiarism and evade detection, studies have characterized plagiarised texts by intensity or degree of obfuscation (alterations), this include no-obfuscation, light and heavy obfuscation (Cloughs and Stevenson, 2013; Potthast et al., 2012; 2013); studies show the difficulty in plagiarism detection increases with increase in obfuscation from no-obfuscation to heavy obfuscation plagiarism (Clough and Stevenson, 2013; Thompson et al., 2015). Plagiarism have also been characterized by different obfuscation strategies, which are alteration techniques employed by plagiarist to disguise plagiarism, common obfuscation plagiarism described in the literature include translation obfuscation, cyclic translation obfuscation, summary obfuscation (Potthast et al., 2013; Foltýnek et al., 2020). Cyclic translation occurs when text written in one language is translated to another language and then back to the original language. Difference in syntax and slight variations in the meaning of words between languages introduce alterations in translated text so that they differ from their original, but remain semantically similar. Summary obfuscation was described under idea plagiarism in section 2.3, it occurs when an original idea is rewritten in different wordings and syntax so that the summarized texts appear different lexically and syntactically, but retain the semantic of the original text. To enhance the detection of obfuscated text, semantic similarity measurement methods developed in the field of NLP and computational linguistic were proposed. Virtually all semantic similarity measurement methods rely on external resources to function. Common semantic similarity measures are grouped into two namely, corpus and knowledge based semantic similarity measures (Eisa et al., 2015; Foltýnek et al., 2019; Salloum et al., 2020).

2.4.3.2.5 Corpus Based Semantic Measures

Corpus based semantic measures obtain sematic information from large corpus such as Wikipedia dump. Common corpus based semantic measures include word embeddings, latent semantic analaysis (LSA) and explicit semantic analysis (ESA). ESA is based on Wikipedia concept (Gabrilovich and Markovitch, 2007; Eisa et al., 2015; Salloum et al., 2020), the basic idea is that a pair of text that map to similar Wikipedia concepts are semantically related, however not much is said about LSA in the literature of monolingual plagiarism detection. The review on semantic measurement methods will be focused on word embeddings because they have been recently used in both mono and cross lingual plagiarism detection, and represent the state-the-art in many NLP applications.

2.4.3.2.5.1 Word Embeddings

Word embedding is the mapping of words to vectors of real numbers (vector representation of words). The application of word embeddings for measuring degree of semantic similarity between texts follows the concept of distributional hypothesis where similarity between words is a measure of the number of contexts they share in common. Words that occur more frequently in similar contexts are considered similar. This idea could be justified by the fact that words that occur frequently in similar contexts can be easily substituted without changing the meaning of the contexts, for example the word *smart* and *intelligent* share similar contexts and can be substituted without changing the meaning of the context in which they occur. Word embedding models are trained to learn the semantics of words from a large corpus so that semantically related words are assigned similar vector representations; training of models is usually carried out using both shallow and deep neural networks (Kumar and Garg, 2019). Word embedding models are either contextualised or non-contextualised learners, non-contextualised learning models (NCLMs) are trained to learn the semantic (meaning) of a word in context, and do not take contextualised relationship into account, which means the actual sense of a word in context is disregarded when training, hence disambiguating the true sense of a word in context is not possible with such models. Non-contextualised or static embeddings do not change irrespective of the context they occur, for example the word *bats* in sentence1 below will have the same representation as the word *bats* in sentence2, even though they are not semantically similar; the word *bats* mean different things in both sentences (contexts).

Sentence 1: '*baseball bats are easy to swing'*

Sentence 2: *'bats are scary looking birds'*

Contextualised learning models (CLMs) on the other hand are trained to learn the relationships between words in context, which includes semantic, syntactic, lexical and miscellaneous relationships, they learn actual meaning or sense of words in context, and therefore generates different representations for a given word based on its context (Ethayarajh, 2019). Contextualised embeddings change when the context they occurs change, from the above example, the word *bats* will have different representations for both sentences. Unlike static embeddings, contextualised embeddings attempt to capture the true sense of a word in context, and are therefore more likely to perform better on sequence level semantic similarity measurement tasks. Words with more than one meaning (polysemy) are better handled by CLMs than static embedding models. Contextualised learning models were proposed to address the limitation of static embeddings including their inability to disambiguate the real meaning of words in context. A recent study on detecting machine generated paraphrase plagiarism revealed that CLMs significantly outperformed non-contextualised learners (Wahle et al., 2022).

Non-Contextualised Learning (Context Independent) Models

Common non-contextualised word embedding models such as the Word2Vec (Mikolov et al., 2013a; 2013b; 2017) and GloVe (global vectors: Pennington et al., 2014) have been successfully used in many NLP tasks to estimate the degree of semantic similarity between texts. GloVe was proposed shortly after the Word2Vec model based on the argument that the Word2Vec model does not take into account global count statistics when building word vectors, in particular global co-occurrence statistics. However in practice, there is little or no difference in the performance between the Word2Vec and GloVe. The Word2Vec model is more common than GloVe, probably because it was first proposed and well-studied, and there are many off the shelf tools that allow for easy implementation of the Word2Vec model. Hence much of this literature is focused on the Word2Vec model.

Word2Vec (Word to Vector) Model

There are two architectures associated with the Word2Vec model, they are the skip-gram and the continuous bag-of-word (CBOW) architecture. The skip-gram model is however more common in many studies, partly because it does not require huge amount of training data to achieve reasonable performance, as opposed to the continuous CBOW architecture that requires huge amount of training data to perform optimally. A Word2Vec model is trained with the objective of maximising the log probability of predicting a context word (*Cw*) given an input word (*Iw*) i.e. log*P*(*Cw* I *Iw*)[1]

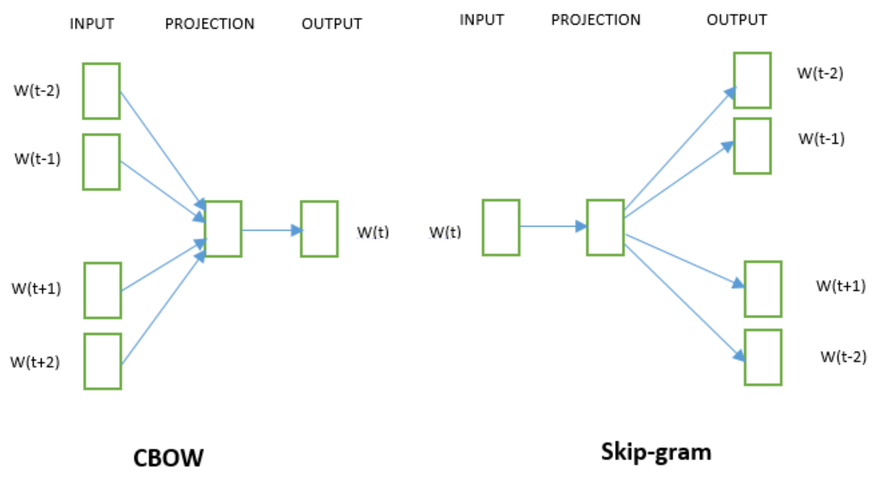


Figure 2.3: The two Architectures of the Word2Vec model.

Training a Word2Vec model begins with the creation of contexts for words (of a specific window size using the CBOW or Skip-gram model) and mapping contextual words to vectors of real numbers (one hot encoding), followed by learning word representations (word vectors) using a feed forward deep neural network with one hidden layer. The Word2Vec model was originally designed for word level similarity computation, many variants now exist for computing sequence level similarity. The basic idea is to average the embeddings of words in a sentence, paragraphs or document embeddings and train a feed forward neural network to generate fixed length representation for the input sequence (sentences, paragraphs or documents). The argument for sentence embeddings is that sequence level representations retains word order and semantics which are not captured in word representations (Le and Mikolov, 2014). A Common variant include Doc2Vec which is trained to generate fixed length representation for an entire document. Fasttext (Joulin et al., 2016) is another variant that is based on character ngrams, it handles sub-words and out of vocabulary words better, and it is trained much faster than the other models.

[1] In figure 3, the CBOW model learns to predict a word w(t) given a context, while the Skip-gram model learns to predict contextualised words given a word w(t).

Word2Vec models have been used in the literature to detect both cross-lingual (CL) and paraphrase plagiarism. Ferror et al., (2017a) applied the Word2Vec to detect CL plagiarism with remarkable performance, Álvarez-Carmona et al., (2018) combined a Word2Vec model with string similarity measures to detect paraphrase plagiarism. In a similar study Alvi et al., (2021) combined the Word2Vec model with Smith-Waterman distance to detect paraphrase plagiarism. Many of the participants in the STS competition (Cer et al., 2017) built systems around the Word2Vec model.

Contextualised Learning Models (Context Dependent)

Contextualised learning models (CLMs) can be subdivided into two basic architecture, namely the LSTM and the transformer architectures (Han et al., 2021). An LSTM is a type of recurrent neural network (RNN) that is designed to capture long term dependencies much better than a traditional (or vanilla) RNN. A RNN learns the context of a word (sequence) by learning to predict the next word in a text sequence using information from previously seen words. Contextualized learning models based on the LSTM architecture include CoVe (McCanan et al., 2017) and ELMo (embeddings from language model) (Peters et al., (2017). On the other hand, models based on the transformer architecture do not follow the sequential learning structure of the recurrent model, but applies what is known as attention mechanism (Vaswani et al., 2017) to learn different aspects (context) of an input sequence concurrently, and can even access all previous states of the network to capture long distance dependencies, this feature allows for parallelization and training on large datasets much faster than a recurrent network. Transformers comprise of an encoder and decoder attention mechanisms. Common contextualised learning models based on the transformer architecture include BERT (Devlin et al., 2018), GPT (Radford et al., 2018), XLNet (Yang et al., 2019) etc. From the two common architecture, ELMo (LSTM) and BERT (Transformers) are the two most common CLMs, and both models were implemented in this research in the task of paraphrase plagiarism detection. Relevant research and details of ELMo and BERT are described below.

Relevant work on ELMo

In an attempt to generate vector representations for word sequences where the sense of a word is context dependent, McCanan et al., (2017) proposed contextualized vectors (CoVe), which are representations from the encoder of a trained MT-LSTM (machine translation-LSTM), evaluation of CoVe on a number of NLP tasks results in state-of-the-art performances. Peters et al., (2017) argue that a contextualized learning models should be able to learn complex relationship between words (semantic and syntactic), including the actual sense of a word in context (polysemy) and proposed ELMo (Embeddings form language model), a bidirectional LSTM trained with a language model objective (next word prediction). Analysis of the hidden state of an ELMo LSTM model revealed that the uppermost layer captures semantic relationship between words and the lower layers capture syntactic relationships. Unlike CoVe (McCanan et al., 2017) that uses only the last hidden layer representation, ELMo learns a linear combination of all hidden units to generate deep contextualised representations for text sequences. Evaluation of ELMo on a number of NLP tasks revealed state-of-the art performances.

Architecture of ELMo

ELMo is a bidirectional LSTM trained with a language model objective (for next word prediction). An LSTM is a RNN that is designed to better handle long term dependencies using gates to control what inputs to accept and which ones to reject during training. Similar to an RNN, inputs to an LSTM are sequences (i.e. sentences), an LSTM learns to predict a token (in an input sequence) using information from previous tokens (see Fig 2.4 below), a bidirectional LSTM learns contextualised representations of an input sequence using forward and backward passes (bidirectional), and concatenate the outputs from both passes.

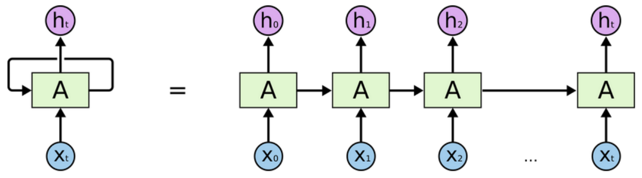


Figure 2.4: Folded and Unfolded/unrolled RNN showing the interaction between input and output tokens

The RNN in fig 2.4 shows input (x_0, x_1..x_n) and output (h_0, h_1..h_n) sequences, and how an input token (i.e. x_1) relies on the hidden state [A] of the previous token (i.e. x_1).

ELMo convert input text to characters, this ensures that representations are generated for all types of input text, and addresses the problem of out of vocabulary (OOV) words that limits earlier word embedding models such as the Word2Vec and GloVe. An ELMo model learns representations for tokens from an input sequence using a linear combination of the hidden states of an LSTM to generate deep contextualised representations. Representations from an ELMo model are context dependent; for example, a word that occurs in two different sentences would have different representations based on its context.

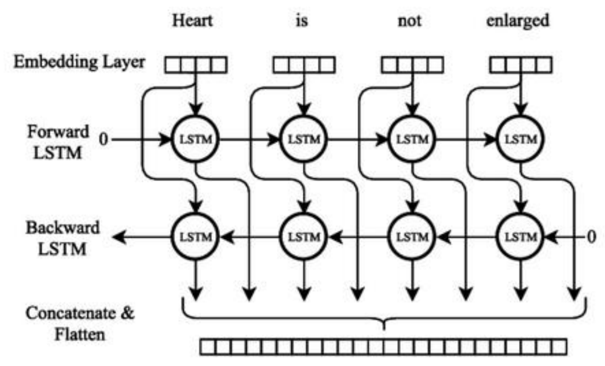


Figure 2.5: An ELMo model showing bidirectional context learning

The diagram in figure 5 shows the flow of input in an ELMo embedding model, the network includes an input embedding layer, forward and backward LSTMs and a concatenated output layer.

The input embedding layer is usually one hot encoding for tokens in an input text, while the output contains contextualised representations of input tokens that can be used for word level semantic similarity measurement or averaged into a contextualised sentence embeddings for comparing sentences. In this way, sentence embeddings from an ELMo model could be compared for semantic similarity and applied in plagiarism detection to detect obfuscated cases.

Relevant Work on BERT

Devlin et al., (2017) propose BERT (Bidirectional encoder from transformers), a stack of encoder only transformers built upon existing sequential models (such as ELMo and GPT), with an objective to improve transfer learning. Similar to the GPT and ELMo, BERT is designed to learn bidirectional contexts so as to capture deep syntactic and semantic relationship, and also to produce context dependent representations. However unlike the previous models that are trained using LM objective, BERT is trained using mask language modelling (MLM) and next sentence prediction. Experimental results revealed state-of-the-art performances on a number of NLP tasks, including semantic text similarity, question and answer, text entailment etc. Several variants of the BERT model have been proposed in the literature, two common variants that are relevant to this work are RoBERTa and SBERT. Liu et al., (2019) argues that the BERT model could perform better by adjusting certain parameters when training, and therefore proposed RoBERTa, which is a BERT model retrained with much larger iterations (from 100 to about 500) and dataset, dynamically changing masking pattern, with the exclusion of the next sentence prediction objective. RoBERTa significantly outperformed BERT on a number of NLP tasks. While BERT is currently one of the best performing DCLM, applying BERT for tasks such as semantic text similarity comes with huge overhead as requires sentences are required to be fed in pairs followed by a similarity computation between every possible combination which is extremely time consuming. To address this problem,

Reimers et al., (2019) proposed sentence BERT (SBERT), which is a RoBERTa model fine-tuned to generate fixed size embeddings for sentences that can be compared for similarity using a measure such as cosine similarity. Sentence BERT comprises of a Siamese network (which are two identical BERT heads) and an average pooling layer, the BERT heads generate embeddings for a pair of sentences, and pass them to a pooling layer, with an object to minimize a cosine loss (mean squared error loss). Results from evaluation revealed that it takes significantly less time to fine-tune SBERT than RoBERTa, while maintaining the same level of performance. SBERT also outperformed two common sentence embedding methods namely inferSent (Conneau et al., 2017) and the universal sentence encoder (Cer et al., 2018) on a number of semantic text similarity tasks, and produced state-of-the-art on the STS benchmark dataset (Cer et al., 2017).

Yang et al., (2019) argues that masking in BERT could result in discrepancy between training and fine tuning representations and therefore propose XLNet, an encoder only transformer that learns bidirectional contexts just like BERT, but uses random permutation for mask language modeling (MLM). To further enhance performance, XLNet adopted the recurrence feature (and position encoding) of transformerXL (Dai et al., 2019) to capture long distance dependencies. Evaluation results revealed state-of-the-art performance on a number of NLP tasks. However evaluation on semantic text similarity revealed that XLNet performed even lower than GloVe (Reimers et al., 2019).

Architecture of BERT

BERT is a stack of encoder only transformers with multi-head attention mechanisms. A transformer is a deep learning model that comprises of an encoder and a decoder, and an attention mechanism. The encoder mechanism encodes input text sequences by transforming input tokens into contextualised representations, and at the other end, the decoder transforms the encoded sequence element wise into the original input sequence, but in a different format, for example the encoded sequence could be decoded in a different language (e.g. English to French) or from text to speech (speech recognition). The attention mechanism allows the model to focus on different parts of an input sequence without losing contextualised information.

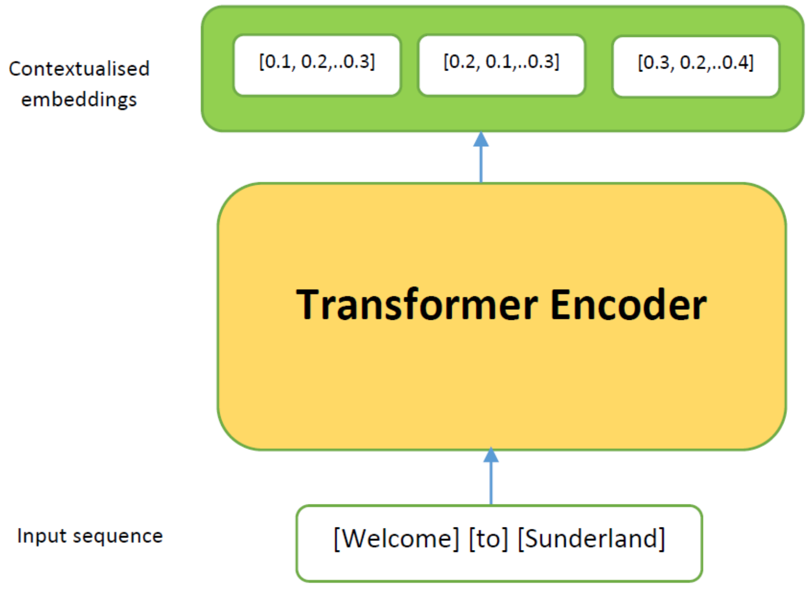


Figure 2.6: A transformer encoder showing input text sequence and encoded output

Mask language modelling (MLM) is a training objective that involves predicting mask words; usually 15% of an input sequence is masked and the model is trained to predict the mask words. Similar to ELMo, training in BERT is bidirectional and contextualised representations from forward and backward passes are averaged for input tokens. See the diagram below for an example of MLM.

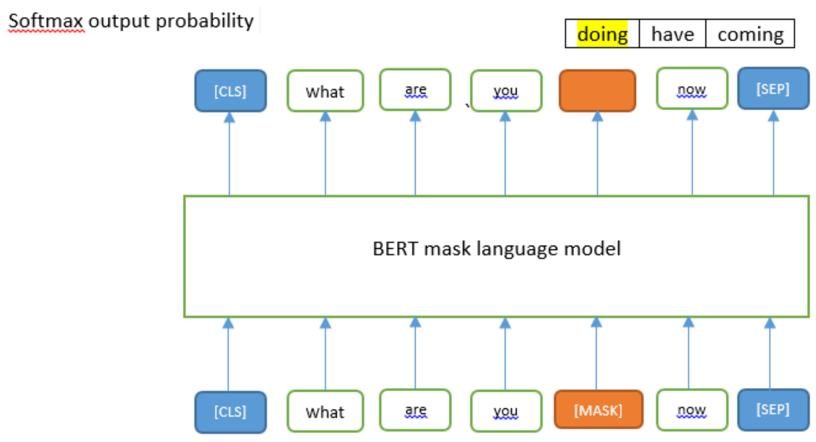


Figure 2.7: A BERT model showing mask language modeling objective

CLS is a special token that does not only act as an identifier, but summarises the token embeddings for an input sequence (Devlin et al., 2019; Zang et al., 2020). SEP is also a special token that separates input sequences (i.e. marks the end of an input sequence).

Next sentence prediction on the other hand involves training a model to predict whether a pair of sentences are (next to each other) in sequence in a corpus, this objective helps to determine how semantically related a pair of sentences is, which can be effective theoretically for addressing NLP tasks such as question and answer. The model of interest in this research is RoBERTa because it is currently the most effective BERT model trained with only MLM objective. BERT comes in two different sizes, they are BERT base which is a stack of 12 encoders, and BERT large which is a stack of 24 encoders and trained with much larger corpus than BERT base. The maximum input size of a BERT model is 512 tokens, and output representations for input tokens is a fixed length vector of 768 dimensions for one token.

Word embedding models come pre-trained, and are designed to be fine-tuned to specific downstream NLP tasks (i.e. sentiment analysis, text classification, entailment etc.) with minimal architectural modifications. In terms of application in plagiarism detection, contextualised word embeddings could be used to detect semantic, syntactic, lexical and miscellaneous changes in plagiarised texts that cannot be detected using string matching methods.

2.4.3.2.5.1.1 Handling Out-of-Vocabulary (OOV) words

One of the challenges facing word embeddings is dealing with out-of-vocabulary words, which are words that do not occurs in the vocabulary of a model. Earlier word embedding models such as the Word2Vec and GloVe typically output error messages when presented with an OOV. In more recent studies, several methods have proposed for handling OOV words, Joulin et al., (2016) proposed fasttext, a variant of the Word2Vec model based on character embeddnigs. Fasttext is able to deal with OOV by tokenising words into characters and generating embeddings for every character in a word which can be aggregated into a single embedding for the word. Similar to fasttext, the ELMo model as described in section ….. is based on character embeddings. A different approach was proposed by Devlin et al., (2018) for the BERT model, when presented with an OOV word, the BERT model tokenises the word into sub-words and attempt to generate embeddings for the sub-words, when a sub-word is an OOV, the model tokenises the word further and repeats the process until sub-words becomes characters and embeddings are generated for each character. The BERT approach does not only dealing with OOV, but can also detect morphological changes and generate embeddings for sub-words, which is important in semantic similarity measurement.

2.4.3.2.5.2 Explicit Semantic Analysis

Explicit semantic analysis (ESA) is a corpus based method proposed by (Gabrilovich and Markovitch, 2007), it is based on the idea that texts that map to similar Wikipedia concepts are semantically related. ESA assumes that each article in Wikipedia is a concept, and transforms concepts into tf-idf (term frequency inverse document frequency) weighted vectors in an *n*-dimensional vector space and indexed in an inverted index data structure that links indexed terms to their Wikipedia concepts. The number of dimensions in the vector space is equal to the number of unique terms in Wikipedia knowledge base (N-dimensional vector space), and the entire index structure is referred to as semantic interpreter (*SI*). See the diagram in figure 2.8 for how *SI* is built by indexing Wikipedia concepts, and transformed to Wikipedia concept vectors to compute semantic relatedness.

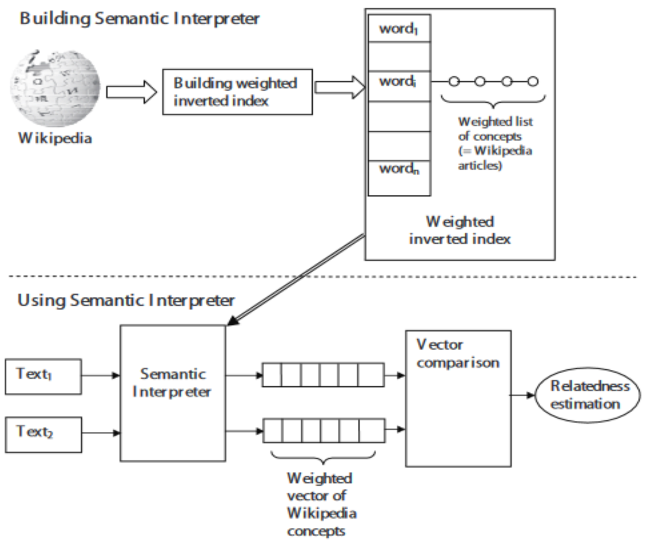

Building Semantic Interpreter
Wikipedia
Building weighted inverted index
word$_1$
word$_i$
word$_n$
Weighted list of concepts (= Wikipedia articles)
Weighted inverted index
Using Semantic Interpreter
Text$_1$
Text$_2$
Semantic Interpreter
Weighted vector of Wikipedia concepts
Vector comparison
Relatedness estimation

Figure 2.8: Semantic interpreter built from Wikipedia concepts (Gabrilovich and Markovitch, 2007)

The diagram in figure 2.8 shows how a Semantic interpreter is used to generate weighted semantic vectors for texts that can be compared for semantic relatedness using the cosine measure.

Comparing texts for semantic similarity is carried as follows: input texts are converted to semantic concept vectors by retrieving concept weights for input texts, this is done by querying the semantic interpreter (inverted index) with each term in an input text. Semantic similarity between a pair of texts is computed by comparing their concept semantic vectors using cosine similarity measure.

Meuschke et al., (2017) integrated ESA with citation detection method to detect semantic relatedness and structural similarity in obfuscated text. The proposed method uses ESA to detect semantic relatedness between texts, and apply in-text citation proximity and patterns to detect structural similarity, evaluation results presented from Meuschke et al., study revealed that the proposed integrated method outperformed established plagiarism detectors that are primarily based on text matching.

2.4.3.2.5.3 Latent Semantic Analysis/Indexing (LSA/LSI)

LSA (LSI in IR) (Deerwester et al., 1990; Landauer et al., 1998) is a dimensionality reduction technique that uses a mathematical technique called singular value decomposition (SVD) to reduce a high dimensional vector space into a lower one (Deerwester et al., 1990; Wiemer-Hastings, 2004; Chandrasekaran and Mago, 2021). LSA basically extracts core information contents in a collection of documents by decomposing a high dimension document matrix into a lower and more informative document matrix structure. In terms of implementation, LSA begins by constructing a term document matrix with an entire collection of document vectors (where unique terms occupy rows and each column is a document vector), followed by decomposing the document matrix into three smaller matrices. Among the three matrices, the most similar to the original term document matrix is used as an approximation of the original matrix. In terms of dimensionality reduction capability, LSI could be used to reduce a vector space with tens of thousands of dimensions down to a few hundred dimensions (Wiemer-Hastings, 2004). LSI can also be used to smooth a VSM by eliminating unfilled dimensions that create sparseness during the dimensionality reduction. The major drawback of the LSA (or LSI) is that dimensionality reduction can result in information loss.

In plagiarism detection and semantic similarity measurement in general, LSA is used to reduce a large VSM to allow for vector comparison between a suspect document vector and source documents in the reduced semantic space using a similarity measure. Ceska (2008) applied SVD to detect paraphrases in plagiarised texts. Most studies on the use of LSA in plagiarism detection are focused on cross lingual and source code plagiarism detection (Ratna et al., 2017).

2.4.3.2.6 Knowledge Based Semantic Measurement Methods

Knowledge based methods obtain semantic information from external knowledge resource such as WordNet, thesaurus and dictionaries (Chandrasekaran and Mago, 2021). The most common knowledge based resource used in plagiarism detection is WordNet; a semantic network and lexical database. WordNet is a network of words and their semantic relationships in a hierarchical structure, and includes relationships such a synonyms, antonyms hyponyms and homonyms (Barbouch et al., 2021). There are two approaches in which WordNet has been used in the literature to detect plagiarism, they are query expansion and path length of synsets (synonyms) or information content similarity. Details of the two approaches are described below.

2.4.3.2.6.1 Query Expansion Using WordNet to Detect Synonym Replacements

Query expansion is used in IR to expand queries to increase matches and improve the quality of information retrieval. The application of query expansion in plagiarism detection is to address the problem of synonym replacements (lexical substitution), which is known as the most common paraphrase phenomenon or technique used by plagiarist to obfuscate plagiarism (Baron-Cerdeno et al., 2013). In the context of plagiarism detection, a suspect document is the query that requires expansion, a suspect document is expanded by retrieving synsets (a list of synonyms) for each word from a lexical resource (i.e. WordNet). Through synset generation, words that have been replaced with their synonyms in the act of plagiarism are reintroduced back into a suspect document to be detected by a matching algorithm when the expanded query (suspect document) is compared with potential source documents. The challenge in this approach is that too many synsets could easily be generated which increases the chances of false matches and erroneous detection. Hence previous studies in this area have been focused on reducing the amount of synonyms generated during expansion so as to minimize false matches, but without bypassing relevant synonyms required for plagiarism detection.

This approach was used by Ceska et al., (2012) and Abdi et al., (2012) to detect obfuscation plagiarism. In an attempt to retrieve accurate synonyms from WN for query expansion and plagiarism detection, Ceska et al., experimented with the first sense (the first synonym in a set of synonyms retrieved from WN) and all sense (all synonyms in a set). Results from Ceska et al., experiments show very little improvement in performance in both cases. In a similar way, Nawab et al., assigned different weights (based on probability) to every first 3 synonyms generated by WN and obtained results that were better than a baseline method (Kullback–

Leibler divergence method) used for candidate selection. Other studies based on query expansion using WN for plagiarism detection and in document similarity search are discussed below.

In an attempt to detect altered cases of plagiarism, Chong and Specia (2011) used WordNet lexical resources to generate synsets as basic units for measuring similarity between pairs of documents. The proposed approach uses the normalised sum of common synsets in a pair of documents (under comparison) as a measure of similarity. The normalization was carried out by dividing the total common synsets in a pair of document by the total unique synsets (similar to the computation of Jaccard-index). Results obtained from this study show improvement in the detection of paraphrase (obfuscated) cases of plagiarism over a baseline method that uses 5-gram overlap for similarity measurement. The problem with Chongs' et al., (2011) study is that comparison against such baseline would not give a realistic view of how effective the proposed model is in detecting paraphrase plagiarism. This is because the baseline model uses 5-gram overlap to detect plagiarism, which can rarely be found in highly altered plagiarised texts. If the baseline was of lower order n-grams such as 1, 2 or 3 grams, it would have been better, as such n-gram sizes are able to detect altered plagiarized fragments.

A slightly different approach was used by Chen et al., (2010) to detect plagiarised passages. The approach involves comparing each synset in a suspected document with the entire collection of synsets in a source document, word for word, and any comparison that results in the highest similarity score is noted. The recorded similarity scores obtained for all the synsets in a suspect documents are added up and normalised by the total unique synsets in the document pair, which is then use as a measure of similarity between the pair. Chen et al., (2010) incorporated this method into existing methods used in ROGUE (a text summarization algorithm) to detect multiple forms of plagiarism. The methods in ROGUE include the longest common sub-sequence (LCS), skip-bigram and unigram. Experimental results revealed that the WordNet method performed best for detecting plagiarism cases that involves substituting words with their synonyms.

2.4.3.2.6.2 Word Pair Similarity Using Path Length and Information Content of a Semantic Network

This approach uses the taxonomy of a semantic network (i.e. WordNet) to measure wordpair similarity, and has not had much application in plagiarism detection. The depth of two synsets (i.e. Wu Palmer similarity) or information contents (IC) of two concepts (Resnik IC) relative to their common subsumer (common ancestors in an ontology) in WordNet taxonomy are two common methods used to measure the similarity between words (Mehachie et al., 2006; Chandrasekaran and Mago, 2021). This approach was used in Sánchez-Vega et al., (2019) as baseline and in Lovepreet et al., (2021) to detect paraphrase and other complex cases of plagiarism respectively.

Given a pair of terms (t1 and t2), their similarity based on Wu Palmer is computed as follows:

$$sim_{wup}(w1, w2) = \frac{2depth(w_{lcs})}{depth(w1) + depth(w2)}$$ ……………………………………………Equation 2.2

Where *lcs* = longest common subsume of both terms (common ancestor)

Resnic information content of two concepts (c1 and c2) is expressed mathematically as follows.

$$res(c1, c2) = IC(LCS(c1, c2))$$ …………………………………………………...Equation 2.3

Where information content $IC(c) = -\log P(c)$,

and LCS is the longest common subsumer of the two concepts

The application of external lexical resources such as WordNet is limited by vocabulary size, for instance WordNet only contains words of certain part-of-speech i.e. nouns, verbs, adverbs and adjectives (Barbouch et al., 2021). This limitation is evident in Álvarez-Carmona et al., (2018) study where the performance of a plagiarism detection system designed to use either WorldNet knowledge base or the Word2Vec model were compared, and the one built with WordNet was clearly outperformed.

2.4.3.2.7 Syntactic Measurement Technique

Syntax are grammatical rules that specify how words should be arranged in phrases and sentences to make grammatical sense. Each word in a sentence is of a particular part-of-speech (POS) and certain parts of speech should not be placed next to each other in a sentence. Common parts of speech in English include Nouns, verbs, adjective, adverb, preposition, and many more.

The application of syntactic measures in plagiarism detection is to detect obfuscated plagiarism by comparing texts for similarity based on their syntactic structure. The most common method used to measure syntactic similarity in plagiarised texts is POS comparison. According to Chong et al, (2010), when a plagiarised word is replaced with a new word, the new word always certainly retain the same POS as the old word, and by comparing documents with their POS features, modified cases of plagiarism could be uncovered. Another notion is that certain parts of speech are more informative that others (which includes verbs, nouns, adjectives, adverbs), for example, WordNet lexical database is made of only nouns, verbs, adjectives and adverbs (Banerjee and Pederson, 2002; Liu et al., 2007; Jurafsky and Martin, 2015), words of other POS are regarded as stop-words (common words) and are not in WordNet. Hence by limiting document comparison to only certain POS, the accuracy of document similarity measurement could be improved, and in particular, cases of highly altered plagiarism could be uncovered.

Bar et al., (2012) combined POS ngrams and Stop-word ngrams to detect syntactic structure in plagiarised texts that have been paraphrased; this method was implemented by representing texts with their POS tags, and transforming the represented text into ngrams which in effect captures the syntactic structure of the text that can be compared with other texts for structural similarity. Vani and Gupta (2015) demonstrated that applying POS tagging in plagiarism detection significantly improves precision by selecting only nouns, verbs and adjectives as keywords. Kong et al., (2015) selected only nouns and verbs as keywords with the help of the Stanford POS tagger for source retrieval in plagiarism detection (Hagen et al., 2015). Gupta et al., (2016) combined stop-word ngram and POS tagging to detect obfuscation plagiarism and presented results that outperforms a baseline model that uses stop-word-ngrams to detect plagiarism by comparing patterns of stop-word ngrams in a pair of documents proposed in (Stamatatos, 2011).

The application of syntactic method, such as POS comparison in plagiarism detection is limited to cases where words are replaced with their synonyms, as in such cases, the new and replaced words are likely to be of the same POS. In complex cases where text are rewritten without synonym replacements, comparison based on POS tagging will not be effect.

Table 2.2: Summary of plagiarism detection methods

Methods	Sub-class	Function	Pros	Cons
Lexical (string)	Ngram overlap, Vector space model, Fingerprinting.	Compute proportion of overlapping strings	Suitable for detecting verbatim and light obfuscation plagiarism. Easy to implement and very efficient.	Not suitable for detecting heavy obfuscation plagiarism.
Semantic	Knowledge based: WordNet, dictionaries, thesaurus Corpus based: word embeddings, SRL, LSA, ESA	Compute word level and sentence level semantic similarity using external resources such as lexical databases (i.e. Wordnet) and word embeddings (i.e. Word2Vec and BERT)	Suitable for detecting obfuscation (paraphrase) plagiarism, such as in synonym replacements	No suitable for large scale, real time plagiarism detection. Rely on external resource which may be limited in vocabulary.
Syntactic and structural	POS patterns, stop-word ngram pattern	Compare POS and stop word patterns.	Suitable for detecting heavy obfuscation plagiarism, and best when used to complement established methods.	Significant pattern of similarity must exist. Not effective for detecting reordering.
Others	Citation based method, Machine learning	Compare patterns in citation.	Suitable when combined with established content based method.	Limited to academic papers that contains citations, and requires a minimum number of citations.

This table contains methods that have been used in the literature to detect plagiarism, the mechanism in which the function and their pros and cons.

One current but not so popular technique used in detecting plagiarism is the citation plagiarism detection technique. Gipp (2014) argue that the results obtained from traditional plagiarism detection systems (such as those that rely on lexical overlaps) on highly paraphrased and translational cases of plagiarism are unsatisfactory, and propose the use of citation matching as a way of complementing existing techniques used in detecting plagiarised documents. The idea behind this technique is that plagiarised documents are likely to have similar citations, even when they have been seriously paraphrased or translated. Hence a pair of documents that have similar in-text citation patterns, in terms of proximity and sequence may have been plagiarised. One limitation of this approach as pointed out by Gip and Beel (2010) is that text passages must be large enough to hold at least two citations for this method to work. Hence the citation approach to plagiarism detection should be used to complement existing techniques and not to replace them, especially in academic documents that often contain citations.

2.4.3.3 Post-Processing

When potential plagiarised text fragments have been detected, the next stage is post-processing where detected fragments in both source and suspect documents are merged into plagiarised passages, and passages that are not well formed are discarded. Post processing is usually carried out using a combination of techniques, and commonly used ones include seeding, merging and filtering (Potthast et al., 2012; 2013; Foltýnek et al., 2019). Details of these techniques are described below.

2.4.3.3.1 Seeding, Merging and Filtering

As stated above, seeding, merging and filtering are common text alignment techniques used in the post plagiarism detection stage. Seeding is the processes of identifying overlapping text fragments in a document pair. Seeds are overlapping units which include words, phrases, n-grams or even passages. Merging or extension is the process of joining nearby seeds into maximum overlapping sequences (phrases, or sentences), while filtering involves removing merged sequences that do not co-occur or satisfy certain criteria, i.e. phrases less than a specific length are removed, and isolated phrases that do not fall into any identified plagiarised passage are filtered off.

Table 2.3: Example of Seeding, Merging and Filtering

	Plagiarised text	Source (original) text
Seeding	Inheritance in object oriented programming is where a new class is formed using classes which have already been defined. These classes have some of the behavior and attributes which where existent in the classes that it inherited from.	In object-oriented programming, inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined. The inheritance concept was invented in 1967 for Simula.
Merging	Inheritance in object oriented programming is where a new class is formed using classes which have already been defined. These classes have some of the behavior and attributes which where existent in the classes that it inherited from.	In object-oriented programming, inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined. The inheritance concept was invented in 1967 for Simula.
Filtering	Inheritance in object oriented programming is where a new class is formed using classes which have already been defined	In object-oriented programming, inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined.

This table contains an example of how seeding, merging and filtering could be performed on a pair of potential plagiarised and source text.

The first row in table 2.3 displays seeds in the pair of text, which are short overlapping substrings, the second row display how nearby seeds are merged into long contiguous text sequences, and the third row displays potential plagiarised and source passage after potions of texts that contain little or no overlaps have been filtered off.

Leilei et al., (2012, 2013) proposed a text alignment algorithm that is based on seeding, merging and filtering. The algorithm uses overlapping sentences as seeds; in particular pairs of sentences (in a suspicious and source document) with overlaps that exceed a specific threshold were used as seed. The algorithm merges neighboring seeds into a plagiarised passage using a sorting algorithm that alternates between seed searching and merging, although details of the merging algorithm were not disclosed. Passages with Jaccard coefficient less than a specific similarity threshold were discarded.

Sánchez-Vega-Parez et al., (2014) proposed a text alignment algorithm that uses seeding, merging and filtering techniques to align plagiarised document and their sources, and won the

Pan@Clef 2014 text alignment competition. The proposed algorithm uses sentences for seeding, and compares each sentence in a suspicious document with sentences in a source document using the VSM with TFIDF, and Cosine similarity and Dice-coefficient for similarity measurement. The algorithm stores overlapping sentences with similarity scores that exceed certain threshold, and merges sentences within certain distance apart. In terms of filtering, the algorithm filters off duplicate fragments, and fragments less than certain length.

The next section reviews common surface similarity measures that have been used in the literature to detect plagiarism and in other related textual similarity measurement tasks.

2.5 Similarity Measures

This review is to identify surface similarity measures that have been dominant in the literature in one similarity measurement task or the other, and use them for subsequent analysis. Similarity measures are needed in many IR and NLP tasks such as plagiarism detection (Barrón-Cedeño et al., 2009; Alvi et al., 2017; Álvarez-Carmona et al., 2018), document clustering and categorization (Huang, 2008; Gali et al., 2019) and in collaborative filtering for recommendation systems (Magara et al., 2018; Amer and Abdalla, 2021), and duplicate and near duplicate detection (Charika, 2002). Similarity measures are functional tools used for measuring the similarity between objects (where objects means text documents in this research). A similarity measure takes in two objects and outputs a numerical value that reflects the degree of their similarity. The numerical value is known as similarity score and it is usually in the range of 0 and 1 where 0 means exact dissimilarity and 1 means exact similarity, values between 0 and 1 are intermediate levels of similarity. According to the literature, there are three major groups of similarity measures, they include string-based, corpus-based and knowledge-based (Mihalcea et al., 2006; Gomaa and Fahmy, 2013; Gali et al., 2019). Knowledge-based and corpus-based similarity measures apply semantic and syntactic similarity measurement techniques such as Latent semantic analysis (LSA or LSI in IR) (Deerwester et al., 1990), pointwise mutual information (Turney et al., 2010; Amigó et al., 2020) or lexical databases such as WordNet for measuring semantic and syntactic similarity between texts. String similarity measures are divided into two groups namely character and term (token) based (Elmagarmid et al., 2007; Gomaa and Fahmy, 2013; Gali et al., 2019).

2.5.1 String Similarity Measures

String similarity measures are divided into character based and term based metrics. Below is a discussion of these metrics;

2.5.1.1 Character Based Similarity Measures

Character-based similarity measures compares texts by sequence alignment using edit operations such as insert, delete, replace and transpose to transform one string to another. Standard edit operations include insert, delete, replace and transpose. The main idea is that the more similar two strings are, the less edit operations is required to transform one into the other, and vice versa. Character based similarity measures are commonly used in DNA sequence alignment and duplicate record detection (Elmagarmid et al., 2007), and are based on dynamic programming and hence not efficient for comparing large texts (documents). Common character based similarity measures include Euclidean distance, Jaro-distance, Jaro-Winkler distance, Smith-Waterman distance etc. Elmagarmid et al., 2007; Gomaa and Fahmy, 2013; Gali et al., 2019). One character based similarity measure commonly used in plagiarism detection (sometimes as baseline) and in document similarity search is the longest common subsequence (Clough and Stevenson, 2011; Bär et al., 2012; Sánchez-Vega et al., 2019)

2.5.1.1.1 The Longest Common Subsequence

The longest common subsequence (LCSS) is a modified version of the longest common sequence (LCS). The longest common sequence is a variant of the Euclidean distance, but differs from the Euclidean distance in that it uses less edit operations to transform one string into another. Unlike the Euclidean distance that transforms one string to the other by character insertion, deletion and replacement, the LCS uses only insert and delete edit operations.

The longest common sequence is very effective in aligning strings, but like most character based similarity metrics, the LCS is based on dynamic programming (hence it is not an efficient algorithm. In order to use the LCS on large texts (such as in plagiarism detection), it has to be made efficient, and the only way to do this is to trade off some of its effectiveness for efficiency. The LCSS is a variant of the LCS that has undergone such trade-off; it is more efficient than the LCS on large texts, but less effective. The longest common subsequence has been successfully used in many studies to detect plagiarism (Chong et al., 2010; Clough and Stevenson, 2011; Baba et al., 2017); however its effectiveness is limited to the alignment of verbatim plagiarism and sometimes lightly altered texts, and will fail when used to detect plagiarised texts that have been reordered.

2.5.1.2 Term (Token) Based Similarity Measures

Term based similarity measures compute similarity by comparing words (or terms/tokens) in documents**;** they use terms as the basic units for comparing texts, and not characters. Term based similarity measures are relatively more efficiency (than character based metrics) on high dimensional data such as documents, and for the most part they are standard in IR for addressing many document similarity measurement problems such as plagiarism detection, document clustering, near duplicate detection etc. Hence term based similarity measures are considered in this research for text similarity measurement. Term based similarity measures are sometimes referred to as surface or lexical similarity measures, this is because they compare texts by matching surface features (terms/strings/characters) without taking into account the meaning (or semantics) of words when measuring similarity between texts (Metzler, 2007; Zesch and Gurevych, 2012; Gali et al., 2019). A term may have more than one meaning (e.g. bat→an animal and a sport equipment), for example the homonym '*address*' in the two phrases below matches lexically, but mean different things semantically in context in both phrases.

- 'my address' and
- 'address the issues'

There are several basic properties that define a proper similarity measure**s**, they include**d**:

(1) The symmetrical property: the output of a similarity measure must be consistent even when the objects of measurement are swapped, for example given two text objects A and B*, sim(A,B)* must be equal to *sim(B,A).*

(2) The triangular inequality property: this property could be described in terms of the equivalent relationship between the three sides of a triangle, and basically means the sum of the distance of any two sides of a triangle must be equal to or greater than the third; *AB + AC =>BC.*

(3) The similarity property: a similarity measure must satisfy the similarity property, which simply means the distance between two identical objects must be 0.

While there are other properties of a proper similarity measure**s**, these three properties are the most widely accepted in research communities. See Oakes (2014) for more.

A non-symmetrical measure can be made symmetrical by taking measurement twice (while swapping the objects of measurement) and using the average. (sim (A, B) + sim (B, A))/2.

Similarity measures are either distance based or similarity based. Distance based measures carry out measurements in distances; their outputs reflects dissimilarity, but can easily be transformed into similarity using one of the following methods;

$\text{Similarity} = 1 - \text{distance}$ or

$$\text{Similarity} = \frac{1}{(\text{distance} + 1)} \quad \text{.........Equation 2.4}$$

The outputs of similarity measures are a direct reflection of similarity between objects and are usually in the range of 0 and 1, there is usually no need for any form of transformation in the output of similarity measures when working on a text similarity measurement task.

2.5.2 Types of Similarity Measures

Different types of similarity measures have been proposed in the literature (Cha, 2007; Magara et al., 2018; Molle, P., Verbelen, 2021; Amer and Abdalla, 2021) text similarity analysis. Term based similarity measures can be grouped into geometric, probabilistic and set based theoretical measures. In practice, term based similarity measures are usually applied on document vectors to determine document similarity. Given two document vectors $\vec{U}$ and $\vec{V}$, their similarity could be computed using the following common term based similarity measures;

2.5.2.1 Geometric Similarity Measures

Geometric similarity measures apply geometrical principles to measure the similarity between text documents. If two objects (or vectors) have similar shape in a Euclidean space, they are likely similar. A Euclidean space comprises of points in a vector space, or coordinates scattered in two or more Euclidean planes. Popular geometric similarity measures include Euclidean distance (L2), Manhattan (L1) (both L1 and L2 are together known as generalised Minkowsky distance) and cosine similarity (Amer and Abdalla, 2020; Diallo et al., 2022).

2.5.2.1.1 Euclidean Distance (ED)

Euclidean distance is the ordinary distance between two points in a Euclidean space that can easily be measured with a ruler. ED is about the oldest and most widely used distance measure (Oakes, 2014), it is sometimes referred to as ‘as the crow flies’ which simply means ‘in a straight line'. ED measures distances based on differences in vector length or magnitude, and does not explicitly focus on content similarity. Hence ED measurements that are equal but of different content cannot be easily differentiated (Diallo et al., 2022). Mathematically, ED is the square root of the sum of square difference between any two vectors. Given two documents vectors *U and V*, their Euclidean distance can be calculated using the following mathematical expression:

$$\text{Euclidean}\ (\mathrm{U},\mathrm{V}) = \sqrt{\sum_{i=1}^{z}\left(\mathrm{u_i} - \mathrm{v_i}\right)^2}$$ ……………………………………………...Equation 2.5

Where *i to z* = Euclidean space over which distance is measured.

ED can easily be implemented and it is also efficient, however one major limitation of the ED is that it is skewed by outliers resulting in uneven scale of measurement; features with outstanding weights in document vectors contribute more to similarity (or distance) measurement than other features. Hence the actual similarity between documents based on relative distribution of common features is often skewed, and likely inaccurate. One way of overcoming this problem is by applying data standardization (or normalization), with common technique such as the z-score.

$$\mathrm{Z\text{-}score} = \frac{\mathrm{x_i}\ \text{- mean(i)}}{\mu(\mathrm{i})},$$
$$\textit{where } \mathrm{x_i} \textit{ is the weight of feature i}$$ ……………………………………………Equation 2.6
$$\textit{and } \mu(\mathrm{i}) \text{ is the } s\tan \textit{dard deviation} i$$

What z-score normalization does is to set the standard deviation to one and mean to zero for every feature. In so doing, the scale becomes normalized as every feature has equal standard deviation and measured from a common mean point. One other disadvantage of the ED as it relates to document similarity measurement is that ED performs best when comparing vectors with two or three dimensions and documents tend to have much higher dimensions (Sohangir and Wang, 2017). Hence the accuracy of ED decreases with increase in dimension, which makes it not quite suitable for distance measurement in high dimensional space such as what is usually encountered in document similarity measurements.

2.5.2.1.2 Manhattan Distance

The Manhattan distance (also known as city block distance or L1) between two vectors is the sum of the difference between the common components shared by the vectors. The difference between Manhattan distance and the ED is that when computing Manhattan distance, the difference between each corresponding vector components is not squared. The Manhattan distance and the ED are both regarded as special cases of Minkowsky distance because of their similarity.

$$\text{Manhattan } (U,V) = \sqrt{\sum_{i=1}^{z} \left(u_i - v_i\right)^1} \qquad \text{...Equation 2.7}$$

$$\text{Minkowsky } (U,V) = \sqrt{\sum_{i=1}^{z} \left(u_i - v_i\right)^p} \qquad \text{...Equation 2.8}$$

When the difference in the Minkowsky distance is raised to power (p) of one (1), the Minkowsky distance becomes Manhattan distance, and if p=2, then it becomes Euclidean distance.

2.5.2.1.3 Cosine Similarity

Cosine measure along with Euclidean distance are about the most widely used surface similarity measures (Sohangir and Wang, 2017). The Cosine similarity uses the cosine of the angle between two document vectors as a measure of similarity; if the angular difference between two vectors is 0^θ (zero degree), cosine 0=1, which means exact similarity, and if the angular difference is 90^θ (ninety degrees), cosine 90=0, which means exact dissimilarity. Cosine measure do not take into account the magnitude (length) of vectors (Diallo et al., 2022), but rather the content similarity (similarity in vector components). In terms of actual implementation, cosine similarity can simply be expressed as the inner product (dot product) of two vectors divided by their norm product.

$$\text{Cosine similarity}\left(\vec{U},\vec{V}\right) = \frac{\text{dot product}}{\text{norm product}} = \frac{\sum_{i=1}^{z} u_i v_i}{\sqrt{\sum_{i=1}^{z} u_i}\,\sqrt{\sum_{i=1}^{z} v_i}} \qquad \text{.......................Equation 2.9}$$

Cosine similarity has really been successful in many IR tasks, especially when combined with TFIDF weighting in the vector space model (Salton et al., 1975; Meuschke and Gipp, 2013, Gupta et al., 2016). However cosine similarity is insensitive to differences in document length; hence some experts argue that cosine similarity in its original form is not suitable for all textual similarity measurement tasks (Shivakuma and Monilar, 1996; Hoard and Zobel, 2003).

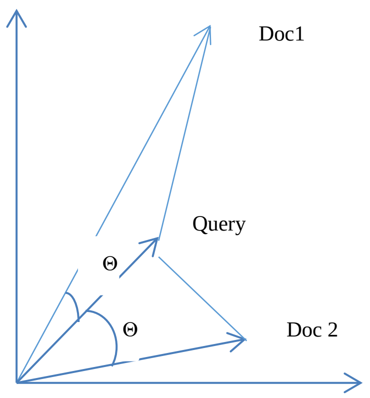


Figure 2.9: Document Comparison in a Vector Space.

Fig 2.2 shows document comparison in a vector space using angular distances between vectors and distances between vector lengths, where the length of a vector is indicated by the arrow sign at the peak.[2]

2.5.2.1.4 Pearson Correlation Coefficient (PCC)

Pearson correlation coefficient is a well-known statistical tool used for measuring the linear relationship between two random variables. Given any two random variables *U* and *V* taken from a population, PCC is mathematically expressed as the covariance of *U* and *V* divided by the product of their respective standard deviation. Where the covariance is the relative variance of variables *U* and *V*. PCC is usually expressed as a number between +1 and -1, where +1 represent perfect relationship.

$$\mathrm{PCC(U, V)} = \frac{\sum_{i=1}^{z}\left(\mathrm{Ui} - \mathrm{mean(U)}\right)\left(\mathrm{Vi} - \mathrm{mean(V)}\right)}{\sqrt{\sum_{i=1}^{z}\left(\mathrm{Ui} - \mathrm{mean(U)}\right)^2}\sqrt{\sum_{i=1}^{z}\left(\mathrm{Vi} - \mathrm{mean(V)}\right)^2}} \text{ ....Equation 2.10}$$

PCC can be computed very efficiently, and can be effective when the relationship between two vectors is linear. However, PCC always assumes linear relationships even when such relationships do not exist, it is also affected by outliers Baak et al., (2019); what this implies in

[2] Note that there are two measurement variables that determine the similarity between objects in a vector space, they include vector length (extent of similarity) and angular distance (content /topic similarity) (Zhang and Korfhage, 1999). A query and document a vector are exactly similar if they have equal length and zero angular distance between them, but are completely different if one is orthogonal to the other. However, for documents similarity measurement, angular distance matters most as it reflects content (or topic) similarity.

terms of document similarity measurement is that not linear relationships cannot be captured, and one or more extremely high and outstanding values could alter the accuracy of similarity measurement, which could potentially result in high similarity in a pair of documents even when they are not really similar or vice versa.

2.5.2.2 Similarity Measures Based On Event Probability

Similarity measures under this group use probability of similar events in two documents to determine how similar they are. Most probabilistic similarity measures are used in information theory; they compare documents using information content or features derived from documents. Commonly used ones include Kulback-Leibler divergence, Jenssen-Shannon divergence and Bayttacharyan coefficient (Amer and Abdalla, 2020; Levada and Haddad, 2021).

2.5.2.2.1 Kullback-Leibler Divergence (KLD)

Kullback-Leibler divergence also known as relative entropy is a one of the most widely used information theoretical measure for computing the divergence between two probability distributions (Barrón-Cedeno, 2009; Levada and Haddad, 2021). Given a pair of texts *U and V,* the *KLD* measure of (*U*||*V)* is the amount of uncertainty or information loss when *V* is used to approximate (Kullback-Leibler, 1951) or replace *U*. KLD is not symmetrical, hence it is not a proper distance measure, but there are many variants of KLD that are symmetrical. Mathematically, KLD can be expressed as follows:

$$\mathrm{KLD}(\mathrm{U} \parallel \mathrm{V}) = \sum_{\mathrm{i}} \mathrm{u_i} \cdot \log \frac{\mathrm{u_i}}{\mathrm{v_i}} \quad \text{…………………………………………………..Equation 2.11}$$

KLD requires that there is absolute continuity in comparison, if $U(i) = 0 \Rightarrow V(i) = 0$. This simply means, if *U* and *V* are two probability distributions, every event in *U* must be compared, and if such event is not present in *V*, KLD becomes undefined ($0 \log 0 = \infty$) due to sparse data. Data sparseness occurs when there is insufficient data to use to make accurate estimate of similarity between two texts. A common solution to the problem of data sparseness is smoothing where vector components with values equal to zero are replaced with a very small positive number (Alison et al., 2006; Sueno et al., 2020). More about smoothing is discussed in the section below under the limitations of the language model.

2.5.2.2.2 Jensen-Shannon Divergence (JSD)

JSD measures how similar two probability distributions taken from the same sample space are. It is a well-known measure in information theory because of its strong statistical grounding. JSD is based on Kullback Leibler divergence (KLD), it is generally referred to as normalized KLD, but differs in the sense that it is symmetrical and normalized, and it is not constrained by absolute continuity, hence its output is always finite. The JSD for any two text documents *U* and *V* can be mathematically represented as:

$$\text{JSD }(\text{U},\text{V}) = \text{dKLD}(\text{U} \parallel \frac{\text{U}+\text{V}}{2}) + \text{dKLD}(\text{V} \parallel \frac{\text{U}+\text{V}}{2}) \quad \text{.............................Equation 2.12}$$

2.5.2.2.3 Bhattacharyya Coefficient (Bhat)

Bhattacharrya coefficient measures the amount of overlaps (intersections) between two statistical distributions. Bhattachayya coefficient and KLD are increasing being used for measuring uncertainty in neural networks and collaborative filtering in recommendation systems (Patra et al., 2015; Singh et al., 2020; Pieter et al., 2021).

$$\text{Bhat}(\text{U},\text{V}) = \sum_{i=1}^{z} \sqrt{\left(\sum \text{u}_i \cdot \sum \text{v}_i\right)} \quad \text{...Equation 2.13}$$

Where *U and V* are two probability distributions, and the letter *i* represents an overlapping event taken from the even space *'i –to- z'*. In light of this research, *U and V* represent the two text documents that are being compared.

2.5.2.3 Similarity Measures Based on Set Theory

Similarity measures under this group represent documents as sets of features and apply basic set theories such as intersection, union etc. to measure how similar two documents are. Popular similarity measures under this group include Jaccard-index, Dice-coefficient and overlap (Egghe and Michel, 2003; Jimenez et al., 2018; Verma and Aggarwal, 2020).

2.5.2.3.1 Jaccard-Index

The Jaccard-index is simply the intersection of two objects divided by their union. It is one of the standard measures used in IR, and was originally designed for similarity measurements involving binary vectors, but can be modified to work with weighted vectors using a version called the extended Jaccard (Ghosh and Strehl, 2006). Jaccard index is computed as the intersection of two sets normalized by their union. Given two document vectors *U* and *V*, their Jaccard-index can be calculated using the formula below:

$$\text{Jaccard Similarity}(U, V) = \frac{U \cap V}{U \cup V} \quad \text{……………………………………………………..Equation 2.14}$$

$$\text{Extended Jaccard}\left(\vec{U}, \vec{V}\right) = \frac{\sum_{i=1}^{z} u_i v_i}{\sum_{i=1}^{z} (ui)^2 + \sum_{i=1}^{z} (vi)^2 - \sum_{i=1}^{z} u_i v_i} \quad \text{……………………………Equation 2.15}$$

2.5.2.3.2 Dice Coefficient

Dice coefficient is very similar to Jaccard index, and in many cases they tend to have similar outputs (many times the same). The main difference between Dice and Jaccard index is that Dice assigns twice as much weight to set intersection than Jaccard-index, and in many textual similarity measurement studies, there is little or no difference in their performance. Given two document vectors *U and V*, their Dice coefficient can be calculated using the function bellow:

$$\text{Dice Coefficient}(U, V) = \frac{2 \cdot |U \cap V|}{|U \cup V|} \quad \text{……………………………………………….Equation 2.16}$$

$$\text{Extended Dice}\left(\vec{U}, \vec{V}\right) = \frac{2\,X\left(\sum_{i=1}^{z} u_i v_i\right)}{\sum_{i=1}^{z} (ui)^2 + \sum_{i=1}^{z} (vi)^2} \quad \text{…………………………………………..Equation 2.17}$$

2.5.2.3.3 Overlap Measure

The overlap measure is an efficient similarity measure that can be very handy when a quick estimate of similarity between texts is needed. The overlap measure of two sets of objects is computed as their intersection divided by the minimum size of the smaller object. Give two sets of objects *U* and *V*, their overlap measures can be computed as;

$$\text{Overlap Measure}\left(U, V\right) = \frac{|U \cap V|}{\min\left(|U| \cup V|\right)} \quad \text{………………………………….Equation 2.18}$$

From the literature and description above, it is clear that similarity measures function differently; which means they have different areas of strengths and weaknesses; some may be effective on some problems, and may not be effective on other problems. It is therefore important to know the right similarity measures to use. Before designing an experiment model that will allow the best performing similarity measures to be determined, it is only reasonable to select from the literature those that have been dominant and successful from previous evaluation studies.

The following similarity measures have been successful in the literature; Bhattacharyya coefficient, Cosine similarity, Dice distance, Euclidean distance, Jaccard-index, Euclidean distance, Jensen-Shannon divergence, Kullback-Leibler divergence and Pearson correlation coefficient.

The literature revealed that these similarity measures are either dominant or successfully used in at least one area of textual similarity measurement tasks such as in text categorization, document classification and clustering etc. Euclidean distance is commonly used in clustering algorithms, especially for k-means clustering (Moradi et al., 2020; Aamir and Zaidi, 2021), it is efficient and can be used to measure distances between objects in two or more dimensional space. Results from Ljubesic et al; (2008) experiments show both Euclidean distance and Jensen-Shannon divergence outperformed many well-known similarity measures in the extraction of semantic similarity between texts. Cosine similarity, Jaccard-index and Dice coefficient are well known IR similarity measures, empirical studies revealed that Cosine similarity, Jaccard-index and Pearson correlation coefficient are some of the best similarity measures for text document clustering. Results from Huang (2008) experiments revealed that document clustering based on Pearson correlation coefficient and Kullback-Leibler divergence were more accurate than clustering based on most other similarity measures. Barrón-Cedeno et al., (2009) applied KLD to reduce search space in plagiarism detection with improved performance. Pearson correlation coefficient once again proved to be more effective than many well-known similarity measures in Forsyth and Sharoff (2014) study. Sanchez-Perez et al., (2014) applied Dice coefficient and Cosine measure to detect plagiarised sentences with outstanding results. In more recent studies, similarity measures such as Jaccard index, Cosine similarity, Bhattacharyya coefficient, KLD and PCC have found great applications in collaborative filtering (Patra et al., 2015; Singh et al., 2020; Ayub et al., 2020; Amer and Abdalla, 2021) for recommender systems.

These similarity measures have all shown to be remarkably effective in different areas of text similarity measurement tasks and are considered in this thesis for plagiarism detection. Another common tools used in surface similarity measurement are term weighting methods. The next subsection reviews common term weighting methods used in IR.

2.5.3 Term Weighting

The success of IR models such as the VSM, and text classification systems to a large extent depends on the type of term weighting method used (Polettini, 2004; Dogan and Uysal, 2020). Term weighting is important as it allows terms in documents to be well represented according to their relevance, which improves the quality of retrieval systems. When documents are pre-processed, they are usually transformed to vectors by assigning weights to indexed terms.

Several term weighting methods have been proposed in the literature, they are broadly classified into local and global term weighting methods (Polettini, 2004, Cummins and O'Riordan, 2006; Domeniconi et al., 2015). Local term weighting methods express the relative importance of terms in a document; common local term weighting methods include term frequency, binary, logarithm weighting and its derivatives etc. Global term weighting on the other hand is focused on expressing the importance of a term across all documents in a collection. It main goal is to project the discriminatory power of each term in a collection of documents; to scale down the weights of highly frequent terms, and scale up the weights of rare terms due to their high discriminating power. Examples of global term weighting methods include inverse document frequency (IDF), term frequency inverse document frequency (TF/IDF), probability inverse document frequency (PIDF) etc. In practice, common term weighing methods used by retrieval systems are binary, TF, IDF and TF/IDF (Domeniconi et al., 2015; Dogan and Uysal, 2020); these weighting methods are common because of their success in previous studies, hence are described in details below:

2.5.3.1 Binary Weighting

In binary weighting, a value of one is assigned to every term that occurs in a document irrespective of their frequency of occurrence, and terms which do not occur are assigned a weight of 0. Binary weighting is one of the oldest weighting methods, it is easy to implement and runs efficiently, but does have obvious limitations. Binary weighting does not take into consideration the relative importance of terms in documents, which is a disadvantage especially when comparing documents that have high dominance of common terms; such common terms (usually with low discriminating power) will contribute more to document similarity than rare terms that could bring about clear discrimination. Hence this weighting method is not suitable for all document similarity measurement tasks.

2.5.3.2 Term Frequency (TF)

Term frequency is the number of times a term occurs in a document.

$$\begin{aligned}&\mathrm{TF}\,(t,d) = f(t,d)\\&\text{where } t' \text{ is a term in document } d',\\&\text{and } f = \text{frequency}\end{aligned}$$

TF is easy to implement and can be computed efficiently, however with TF weighting, common words with high document frequency are assigned higher weights than rare words (with high discriminating power) in a document, as long as such common words occur more frequently. This will therefore result in false matches and inaccurate measurement of similarity as common words will contribute more to document comparison than highly discriminating words.

2.5.3.3 Inverse Document Frequency (IDF)

IDF measures the relative importance of terms in a document collection; it is mathematically expressed as the logarithm of the quotient of the total number of documents divided by the document frequency of a term. Document frequency itself is the number of documents in which a term occurs in a collection.

$$\begin{aligned}&\text{Inverse Document Frequency}\,(\mathrm{IDF}_t) = \log\frac{N}{\mathrm{DF}_t}\\&\text{where } N = \text{number of documents in a collection} \;\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{Equation 2.19}\\&\text{and } \mathrm{DF}_t = \text{document frequency of term } t\end{aligned}$$

2.5.3.4 Term Frequency Inverse Document Frequency (TF/IDF)

TFIDF is a global term weighting scheme and a relevance measure (Qaiser et al, 2018), it is simply the multiplication of term frequency (TF) and the inverse document frequency (IDF). Both TF and IDF can be derived as described above. The intuition behind the implementation of TF/IDF is to balance the weights of terms in a corpus by assigning more weights to rare terms and scale down the weights of common terms that have high document frequency. Hence rare terms, and terms with high TF and low DF are often weighted higher than common terms with high DF.

$$\text{TF/IDF} = f(t,d) \times \log \frac{\text{N}}{\text{DF}_t} \quad \text{...Equation 2.20}$$

TFIDF is one of the most successful term weighting methods used in IR. TFIDF is the main weighting method used in SMART information retrieval system (Salton, 1971). However, TFIDF often scale down the weights of common words, and when working with real life data, there are often cases where common words are better discriminators than rare words. Hence the use of TFIDF in such situation would most likely result in inaccurate similarity measurement.

The next section reviews the literature on cross-lingual plagiarism and common methods used in the literature to detect cross-lingual plagiarism.

2.6 Cross-lingual Plagiarism Detection

This section describes state-of-the-art methods used in detecting CLP, some of which have been in existence for a while relative to newer models based on word embeddings and knowledge graphs.

2.6.1 Cross-lingual Character N-Grams (CL-CNGs)

Cross-lingual character n-grams (CL-CNGs) use character n-grams to measure syntactic similarity between texts, it is suitable for languages that have high lexical similarity such as English and French (McNamee and Mayfield, 2004; Barrón-Cedeño et al., 2013). This model has the advantage of being language independent, as similarity is computed by matching strings (characters).

2.6.2 Cross-lingual Explicit Semantic Analysis (CL-ESA)

Cross-lingual explicit semantic analysis (CL-ESA) (Potthast et al., 2011) uses comparable (intermediary) corpora to capture topic similarity based on explicit semantic analysis (Gavrilovic et al., 2007). In CL-ESA, semantic similarity is computed using Wikepedia concept, the idea is, if a pair of texts in two different languages maps to the same Wikepedia concept, then they are semantically related. See section 2.4.3.3.1 for details on explicit semantic analysis.

2.6.3 Cross-lingual alignment-based similarity analysis (CL-ASA)

Cross-lingual alignment-based similarity analysis (CL-ASA) (Barrón-Cedeño et al., 2013) uses statistical machine translation (SMT) based on the IBM model to align parallel corpora. Statistical machine translation is based on probability distributions, the main idea is to maximize the probability of mapping a string S_s in one natural language (e.g. Spanish) into a string S_e in another natural language (e.g. English). An implementation SMT based on Bayes rule could be expressed mathematically as follows:

$$\overset{\neg}{e} = \frac{\arg\max\ p(e|s)}{e \in e^*} = \frac{\arg\max p(s|e)\ p(e)}{e \in e^*} \quad \text{...Equation 2.21}$$

The first and second languages are usually referred to as the source and target languages respectively. Once texts have been translated, plagiarism could be detected format a standard monolingual format. One drawback machine translation models is that they are computationally expensive (Stegmüller et al., 2021), and they require parallel corpora which takes a lot of time to create.

2.6.4 Knowledge Based Methods

Knowledge based methods rely on external knowledge resources such as Eurovoc and multilingual dictionaries to translate texts from one language to another before applying a monolingual detection method. Gupta et al., (2012) and (Pataki, 2012) applied knowledge based methods with encouraging results. One major limitation of this approach is that performance in plagiarism detection is often related to vocabulary sizes of knowledge resources used, which are often limited.

2.6.5 T + MA Model (Translation plus Monolingual Analysis)

The T+ MA model normalises texts into a common language using an online machine translator, before applying monolingual plagiarism detection methods. The T + MA method is the most common, having been used by most participants in the Pan competition on plagiarism detection, and in the SemEval (Agirre et al., 2016; Cer et al., 2017) competition on cross and multi lingual semantic similarity analysis, including the best performing systems (Tian et al., 2017; Wu et al., 2017). Evaluation of state-of-the-art CLPD models revealed that the T + MA model is the most effective model for detecting CLP (Barrón-Cedeño et al., 2013). However, the T+MA model is limited by the fact that internet translation tools are not always available, it is not effective in situations where texts are translated and then altered by replacing words with similar words. The T+MA model is also limited by the amount of translations required for large scale plagiarism detection over the internet, and can easily be disrupted by failed or slow internet connection or online traffic.

Newer Approaches for CLPD

In more recent studies, approaches based on word embeddings and semantic networks have been proposed for CLPD.

2.6.6 Cross Lingual Word Embedding

Word embedding models use distributed representation of words to predict semantically similar words; the basic idea is that words that appear frequently in the same contexts are considered similar. Common but efficient (and effective) word embedding architectures include the wor2vec CBOW and skip-gram models (Mikolov et al., 2013a), and Glove (Pennington et al., 2014; Ghannay et al., 2016). These models map words to vectors of real numbers, and follow the intuition that when words are represented in a common vector space, similar words should have similar vector representations. Word embeddings were originally proposed for monolingual similarity analysis, but have been extended to cross-lingual settings where a joint embedding space is used to learn cross-lingual representation of words (Ruder et al., 2017); typical implementation involves projecting the embeddings of a source language into the space of a target language. Common CL-WE models are the canonical correlation analysis (CCA) (Faruqui and Dyer, 2014; Lu et al., 2015; Ammar et al., 2016), alignment projection (Guo et al., 2015; 2016), and the linear transformation model proposed by Mikolov et al., (2013b). The CCA projects matrices from parallel corpora into lower dimensions of maximum correlation, and translate by projecting across both matrices. The alignment projection method aligns a parallel corpora (bilingual dictionary) and project words in a target language to the embedding space of a trained monolingual source. The linear projection method uses a linear transformation function to map the embeddings of one language into the space of another using a trained dictionary that learns the function. Duong et al., (2017) argued that most of the common CL-WE models were built for bilingual analysis, and proposed a method based on multilingual joint training that combines bilingual word embeddings from multiple languages in a common embedding space and obtained encouraging results. CL-WE are rarely used in CLPD, although they can be effective offline machine translators. In one application of CL-WE in CLPD, Ferrero et al., (2017a) used bilingual embeddings (from a parallel corpus) mapped to a common space, and trained using the Word2Vec CBOW model to detect CLP. The actual implementation was carried out using MultiVec (Berard et al., 2016); a toolkit designed for creating and managing a number of distributed representation models, and the specific model used was original proposed by Luong et al., (2015); a skip-gram extension of Word2Vec in a bilingual space. CLPD was carried out by comparing word vectors in pairs of suspect and source sentences using the cosine measure, pairs of sentences with similarity scores above a predefined threshold are considered potential plagiarised fragments. The skip-gram and CBOW models retain word order and capture the context of a word, which reveal a lot about the word. In a similar study Glavaš et al., (2017) applied the linear transformation method proposed by Mikolov et al., (2013b) in CLPD; the actual detection of plagiarism was based on word vector comparison on sentence level similar to Ferrero et al., (2017a).

2.6.7 Knowledge Graphs/Semantic Networks

Semantic networks link words with similar meanings in different languages to common concepts; words more closely linked to a concept are assigned higher weights than words further away. Examples of semantic networks are BabelNet and ConceptNet (Franco-Salvador et al., 2014, 2016; Speer. and Lowry-Duda, 2017). Franco-Salvador et al., (2014; 2016) proposed a CLPD model that uses knowledge graphs built form a multilingual semantic network (MSN) to compare documents in different languages for semantic similarity, and CLPD. Knowledge graphs capture relationships between concepts derived from text fragments in a document, while a MSN links semantically related words in different languages to a specific concept. The knowledge graph method was proposed because most of the existing CLPD models were designed to detect texts that have been translated to other languages using online machine translators, and struggle to detect translated texts that have been paraphrased. To address the problem, Franco-Salvador et al., applied BableNet, a MSN made from concepts derived from Wikipedia and WordNet, to build knowledge graphs for documents written in different languages and apply a similarity function to measure their similarity. Results from Franco-Salvador et al., study shows that the knowledge graph method outperformed state-of-the-art methods.

2.6.8 Hybrid CLPD Models

Hybrid models that combine word embeddings with other methods have also been proposed. A hybrid of word embeddings and knowledge graph was proposed in (Franco-Salvado, 2016) with the aim of determining whether both models complement each other in detecting CLP, evaluation results presented by the authors revealed that the models are complementary, and that when in combination, they outperformed several state-of-the-art CLPD models. Speer and Lowry-Duda (2017) used concept-net to combine pre-trained Word2Vec and Glove models (word embedding models) into a multilingual semantic similarity detection system, and emerged best in the SemEval 2017 competition. Concept-net is an open multilingual knowledge graph that generates concepts that relate meanings of words and phrases. The idea used in combining the concept graph to a word embedding model is retrofitting, where a pre-trained embedding model is built upon a concept graph. This process is carried out separately on the individual word embedding models, and then combined using a unified vocabulary. The redundant features/dimensions that result from the combination are then removed via truncated singular value decomposing (SVD), a technique used in latent semantic analysis to reduce the dimensions of a VSM to only the most relevant ones. España-Bonet and Barrón-Cedeño (2017) presented a language independent model that measures the semantic similarity between text snippets across multiple languages. The system uses a Support Vector Machine (SVM) to combine a number of intertextual features, which includes features derived from embeddings trained using the Word2Vec model and a multilingual corpora, from lexical similarity measurements, from the internal representation (hidden layer) of a neural network trained using multilingual parallel corpora and from CL-ESA. This approach is however best suited for low resource languages.

Evaluation of state-of-the-art CLPD models (Barrón-Cedeño et al., 2013) revealed that the T + MA model outperformed the others due to its precision in translating texts using online machine translators, and as mentioned in (Burrows et al., 2013; Barrón-Cedeño et al., 2013), precision is the single most important measure used in plagiarism detection as it reduces the time in deciding whether plagiarism is carried out or not. Hence one of the objectives of this thesis is to propose a CLPD model functions with similar precision and performance to a T + MA model, but without the limitations.

The next section briefly discusses some of the main challenges facing plagiarism detection.

2.7 Challenges in Plagiarism Detection

One of the main challenges to plagiarism detection is the complex tactics used by plagiarist in carrying out the act of plagiarism (Pertile et al., 2016). It is a fact that majority of conventional plagiarism detection software are ineffective in detecting paraphrased and translated plagiarism (Foltýnek et al., 2020; Kaur et al., 2021). Due to the absence of overlaps, many cases of plagiarism go undetected as traditional plagiarism detection tools rely on overlaps to perform. To make matters worse, plagiarism in the form of reuse of other people's ideas render most detection software ineffective. Such cases can only be detected using manual comparison; even semantic similarity techniques may not be really effective in such cases. In recent years, plagiarists have gone a step further to apply complex tactics such as the use of foreign characters to beat plagiarism software. Since most plagiarism software are designed to work with English characters, replacing some English characters in a plagiarised passage with foreign characters (i.e. German characters) renders most plagiarism software ineffective.

Another challenge facing conventional plagiarism detection is the lack of evaluation corpus for testing plagiarism detection software before deployment into production environments remains an issue (Foltýnek et al., 2020). Plagiarism detection, specifically in academic institutions of learning is sensitive and care should be taken to ensure that plagiarism detection software are accurate and reliable before deployment into a production. One way of ensuring accuracy and reliability is by testing on real life data and measuring performance and analyzing results under different conditions. The question then becomes how can real life data be acquired for testing plagiarism detection software? There is currently no real life plagiarism corpus to use for testing and evaluating plagiarism detection systems; this is because of the legal ramifications of using real life plagiarism cases (Clough and Stevenson, 2011; Foltýnek et al., 2020). Due to the right of privacy given to students, plagiarist and academic institutions have to agree on the use of plagiarised cases for system development purposes, which is something that is almost impossible as virtually all plagiarists would oppose the idea of making their information public.

The next section reviews common methods used in the literature to evaluate plagiarism detection and classification systems.

2.8 Metrics Used for Evaluating the Performance of Plagiarism Detection Systems

The performance of a plagiarism detection system is measured in the same way as most IR systems; the same performance metrics can be applied. Standard metrics for measuring the performance of plagiarism detection systems includes precision, recall (Zobel and Moffat, 2006; Manning et al., 2008) and F1-measure (Baeza-Yates and Ribeiro-Neto, 2011; Baron-Cerdeno et al., 2013). Efficiency is another performance measure that can be measured in terms of computational time or complexity.

Here are all the possible outcome from a detection model or classifier that define the above evaluation metrics.

True positive (TP): number of positive examples correctly classified as positive.

False positives (FP): number of negative examples wrongly classified a positive.

True negative (TN): number of negative examples correctly classified as negative.

False negative (FN): number of positive examples wrongly classified as negative.

The precision of a retrieval system is a measure of how precise the system is; it can be calculated as the number of relevant retrieval divided by the total number of items (documents) retrieved. Mathematically, this can be represented as;

$$\mathrm{Pr}\,ecision = \frac{Number\,of\ documents\ accurately\ retrieved}{Total\,number\,of\ documents\ retrieved} = \frac{TP}{TP+FP}$$

Recall is an expression of sensitivity; recall is a measure of how sensitive a detection model is with respect to positive (relevant) items the recall of a system is the number of relevant retrieval divided by the number of relevant items expected;

$$\mathrm{Re}\,call = \frac{True\,positives}{Total\,number\ of\,document\ \exp ected} = \frac{TP}{TP+FN} \quad \text{……………………..Equation 2.22}$$

The difference between the precision and recall of a system is not supposed to be large, but if it does happen, a common practice is to use F1-score as a measure of performance. The F1-measure (F1-score) is the harmonic mean of precision and recall, it is a measure that takes into account the precision and sensitive of a system.

$$F1\text{-}measure = 2 \times \frac{\Pr ecision \times \operatorname{Re} call}{\Pr ecision + \operatorname{Re} call}$$ ....Equation 2.23

In an event where time is equally as important as effectiveness, then the performance of a system may be measured in terms of efficiency. The efficiency of a system can be measured in terms of its response time, or the rate at which inputs are processed into outputs (throughput). Efficiency can be expressed in percentage, likewise precision, recall and F-score.

These evaluation metrics do not take into account details of plagiarised fragments (granularity) such as the extent to which a fragment is detected (complete or partial), and multiple detection of a single plagiarised fragment. To address this problem, a number of evaluation metrics were proposed in Pan (Potthast et al., 2014; 2015) for measuring performance at character level, they include precision, recall, granularity and plagdet score. These measures are applied using positional character alignment (character overlaps) between actual plagiarised passages *S= {s1,s2....sn}* and passages retrieved by a detector *R={r1,r2,...rn}* for a pair of suspect and source documents, and averaged across all plagiarised cases in the corpus. Precision: measures the proportion of retrieved passages that are relevant

$$prec(S,R) = \frac{1}{R} \sum_{r \in R} \frac{|\bigcup_{s \in S} (s \cap r)|}{|r|}$$ ....Equation 2.24

Recall: measures the proportion of relevant passages retrieved.

$$rec(S,R) = \frac{1}{S} \sum_{s \in S} \frac{|\bigcup_{r \in R} (s \cap r)|}{|s|}$$ ....Equation 2.25

Granularity: Penalises for multiple (or fragmented) retrieval of a single plagiarism case.

$$gran(S,R) = \frac{1}{|S_R|} \sum_{s \in S_R} |R_S|$$ ....Equation 2.26

Plagdet score: combines precision, recall and granularity into a single performance score for ranking plagiarism detectors.

$$plag\det(S,R) = \frac{F_1}{\log_2(1 + gran(S,R))}$$ ....Equation 2.27

F1=harmonic mean of precision and recall.

These evaluation metrics are only relevant when details of plagiarised passages are provided, such as length and offset (starting character) of plagiarised passages (included in ground truth), otherwise the IR metrics described above should be used.

One other tool that can be used to evaluate the performance of plagiarism detection system is area under the receiver operator characteristic (AUC-ROC) commonly used in NLP and machine learning for comparing classification systems. An ROC curve is a plot of true positive rates against false positive rate of a binary classifier, it gives a clear picture of the accuracy of a classification model with respect to a positive class.

$$True\ positive\ rate = \frac{TP}{TP + FN} \quad \text{...Equation 2.28}$$

$$False\ positive\ rate = recall = \frac{FP}{FP + FN} \quad \text{...Equation 2.29}$$

AUC is a measure of the area under the ROC curve, the larger the area, the better the classification accuracy of a model. AUC is recommended for binary classification problems (Davis and Goadrich, 2006), such as classifying a pair of source and suspect text as plagiarized or not. AUC values range from 0.5 to 1, where 1 is the maximum score assigns to a perfect classifier, and 0.5 is assign to a poor classifier that makes random guess (Szymura, 2022); that completely lacks the ability to discriminate between two classes. AUC is used in authorship attribution (Stamatatos et al., 2015) for comparing detection systems, authorship attribution is one of the closest task the plagiarism detection.

The next section this chapter, and highlights important topics covered in the literature relevant to the research problems.

2.9 Summary

This chapter reviewed the relevant literature on plagiarism and plagiarism detection, with emphasis on external plagiarism detection using automated systems in both mono and cross-lingual settings. The review covered state-of-the-art methods used in the literature to detect plagiarism which includes methods for data pre-processing, candidate retrieval and text alignment. For data pre-processing, NLP techniques such as tokenization, case-folding, stop-word removal, stemming were described. For candidate selection, information retrieval techniques used in document ranking and retrieval such as search engines, keyword selection were described. For text alignment which involves exhaustive similarity search and detailed comparison, methods used for detecting copied text which includes lexical matching, semantic and syntactic similarity measurement methods, and text alignment techniques (seeding merging, and filtering) used for joining detected fragments into plagiarized passages were all reviewed. Common IR term weighting methods for optimizing document similarity measures (TF, IDF and binary) weighting schemes were described. In terms of Cross-lingual plagiarism, the review covered old and new methods used in the literature for detecting cross lingual plagiarism, with much emphasis placed on new methods such as word cross lingual word embeddings and knowledge graphs. The review also covered methods proposed in the literature for evaluating plagiarism detection systems, mostly from the field of IR given that plagiarism detection is primarily an IR tasks, evaluation metrics such as precision, recall and F1-score were described, and newer evaluation metrics (proposed in the Pan@Clef competition on plagiarism) that take into account details of plagiarised fragments (i.e. granulaty and plagdet) were described. Also included are major challenges facing conventional plagiarism detection, and number of weaknesses in terms of performance of current plagiarism detectors were highlighted.

3 Methodology

3.1 Introduction

This thesis is about developing models for addressing some of the current challenges facing plagiarism detection which includes paraphrase and cross-lingual plagiarism. This thesis is also about improving the state-of-the-art in plagiarism detection using advance technology such as the deep neural network (DNN). In chapter two, a review of the relevant literature was carried out in order to study the problem of plagiarism detection, and to identify state-of-the-art approaches, methods and tools used in the literature to detect plagiarism. This chapter looks at a methodology designed to address the research problems, and this breaks down into three research questions, which are:

1. What are the best performing combination of surface similarity measures and textual features (as measured by precision, recall and f1-score) from those described in the literature for detecting similar and near similar texts?

2. Can deep contextual learning models be used to enhance the detection of paraphrase plagiarism with performances comparable or better than the current state-of-the-art?

3. Can a multilingual translation model that does not rely on the internet be built from the Word2Vec (word representation) model and applied to effectively detect cross-lingual plagiarism (CLP) with performances comparable to the state-of-the-art?

The next section outlines the structure of the methodology, which includes the methods and approaches chosen to address the above research questions.

3.2 Structure of Chapter

The methodology is structured so that simple but effective methods (proposed in this research) for detecting plagiarism using surface features were first described, followed by more advanced methods (proposed in this research for detecting more complex cases of plagiarism that may or may not have surface features) that uses features deeply embedded in text for measuring semantic similarity. The final aspect of the methodology looks at cross-lingual plagiarism (CLP) detection (where textual features are written in different languages), and describes a novel method proposed in this research to address the problem of CLPD.

The evaluation of each method, including evaluation datasets, metrics and baselines were also described as part of the methodology (and experimental design).

The rest of this chapter is organized into the following sections:

- *Section 3.3: Datasets;* this section describes the datasets used in this research to address the questions raised; the datasets include corpora for evaluating paraphrase and cross lingual plagiarism detection systems, the datasets come with ground truth which contains details of plagiarism cases for evaluation purpose.
- *Section 3.4*: *Method chosen to address research question 1*; this section describes the approach used to address research question one (1), this includes the evaluation of methods that uses surface textual features to detect plagiarism.
- *Section 3.5: Method chosen to address research question 2;* describes the approach used in this research to address research question two (2), and includes evaluating methods that incorporate state-of-the-art NLP tools that use deeply embedded features in text to measure semantic similarity into existing methods used in plagiarism detection to enhance the detection paraphrase plagiarism.
- *Section 3.6: Method chosen to address research question 3;* this section describes the approach used in this research to address research question three (3), and involves evaluating a method proposed in this research for detecting CLP that uses a multilingual plagiarism detection tool proposed in this work built using a non-contextualised word embedding model.
- *Section 3.6* summarises the entire methodology, with a list of experiments described in the subsequent sections.

3.3 Datasets

The datasets used in this research to evaluate plagiarism detection systems include corpora for evaluating paraphrase plagiarism, cross-lingual plagiarism and monolingual plagiarism (with alterations of varying degrees of intensity) detection systems.

3.3.1 Paraphrase Datasets

The datasets chosen to evaluate paraphrase plagiarism detection systems are the Crowdsourcing paraphrase corpus (Burrow et al., 2013), the P4P (paraphrase for plagiarism) corpus (Barrón-Cedeño et al., 2013) and the Microsoft research paraphrase corpus (MRPC) (Dolan and Brockett, 2005).

These datasets are described in details below.

3.3.1.1 Crowdsourcing Paraphrase Corpus

The Crowdsourcing corpus (Burrows et al., 2012) contains 7,859 pairs of passages, of which 4,067 are paraphrased and the remaining 3792 pairs are non-paraphrased. The passages were obtained from crowdsourcing, which involves enlisting paid workers with specific instructions to create paraphrase passages using one or more of the following alteration techniques; synonym replacement, word or phrase reordering, insertion/deletion, inflectional changes of texts etc. The corpus comes with groundtruth that contains details about whether a pair is paraphrased or not.

Table 3.1: Statistics on the Crowdsourcing paraphrase corpus

Classes	Number of pairs
Paraphrased pairs	4067 (51.75%)
Non-paraphrased pairs	3792 (48.25%)
Total	7,859

Statistical distributions of pairs of paraphrase and non-paraphrase texts in the Crowdsourcing paraphrase corpus.

3.3.1.2 The P4P Paraphrase Plagiarism Corpus

The P4P paraphrase plagiarism corpus was created from the Pan@Clef 2010 evaluation corpus for plagiarism detection, which was artificially generated from web data. The P4P paraphrase corpus contains 859 pairs of short texts of which 847 have been paraphrased at sentence level and annotated with paraphrase types. There are 20-paraphrase types in the corpus, of which same lexical substitution is the most prevalent. See Barrón-Cedeño et al., (2013) for more details on the P4P paraphrase corpus. For plagiarism detection at sentence level as in the case of this research, a total of 6030 pairs of sentence comparison can be carried with the corpus, of which 2481 are annotated as paraphrase, and the remaining 3549 are non-paraphrase according to the ground truth. The table below are example of pairs of texts taken from the P4P corpus that contains paraphrase fragment of text and the specific paraphrase types as annotated in the groundtruth.

Table 3.2: Examples of paraphrase instances in the P4P corpus

Paraphrase passage	Original passage	Paraphrase type
This question is linked closely to the often-debated issue of the Pointed Style's beginnings. Still, in my opinion, the use of "Gothic" might well have origins unconnected to the emergence of the pointed arch.”	This Query is, of course, intimately connected with the much-disputed question of the origin of the Pointed Style. But yet I imagine that the application of the term "Gothic" may be found to be quite distinct, in its origin, from the first rise of the Pointed Arch.”	Lexical change annotated as ‘*same polarity lexical substitution*’---- The substituted lexicons are*: [(Question, Query), (linked, connected), (often-debated, much-disputed), (beginning, origin)]*.
Since the principles regulating the constitution have already been established and talked about, next will be an examination of the specific powers it's supposed to have, as per the way the convention set it up.	Having thus laid down and discussed the principles which ought to regulate the constitution of the federal judiciary, we will proceed to test, by these principles, the particular powers of which, according to the plan of the convention, it is to be composed.	Syntactic change annotated as *‘Syn-diathesis’*—structural alterations around a verb.

This table contains examples of paraphrase plagiarism in the P4P corpus, each example is a pair of paraphrase and source text. Highlighted in yellow are parts (text fragments) of the examples that have been paraphrased (altered) and their respective sources (original).

Table 3.3: Statistics on the P4P paraphrase plagiarism corpus

	Paraphrased	Non-paraphrased	Total
Short text	847	12	859
sentences	2481	3549	6030

Table 3.4 shows statistical distributions of paraphrase and non-paraphrase texts in the P4P paraphrase plagiarism corpus.

3.3.1.3 Microsoft Research Paraphrase Corpus (MRPC) dataset

The MRPC (Dolan and Brockett, 2005) contains pairs of texts annotated (labelled) as paraphrase or not by human. The corpus contains 5801 pairs collected from newswire articles divided into training and text sets, the training subset contains 4076, of which 2753 are paraphrased, while the test subset contains 1,147 paraphrased pairs out of 1725 pairs. The Remaining pairs are non-paraphrased. The MRPC is a standard dataset commonly used as benchmark for evaluating classification models in the task of paraphrase identification.

Table 3.4: Distribution of paraphrase and non-paraphrase samples in the MRCP corpus

	Paraphrased	Non-paraphrased	Total
Training subset	2753 (67.5%)	1323 (32.5%)	4076
Test subset	1147 (66.5%)	578 (33.5%)	1725

This table shows statistical distribution of paraphrase and non-paraphrase examples in the MRPC. The corpus is clearly imbalance, with significantly more paraphrase instances than non-paraphrase on both the training and test subsets.

The first two paraphrase corpora described above were created specifically for the task of paraphrase plagiarism detection, while the MRPC corpus was created for paraphrase identification tasks (general paraphrasing). As discussed earlier in section 2.2.1.2, paraphrasing in general is similar to paraphrase plagiarism, the same alteration techniques (mechanism) are employed, and they both ensure paraphrase texts are semantically similar to their original. However unlike in corpora used for paraphrase identification task that are often a collection of pairs of paraphrase and original sentences, corpora created for paraphrase plagiarism detection often contain instances of plagiarised and original (source) text that are usually passages of text (two or more sentences) that contains fragments (i.e. sentences) that have been paraphrased. On account of the above, it is reasonable to say that in the absence of paraphrase plagiarism corpus, a corpus created for paraphrase identification could be used for training and evaluating paraphrase detection models.

As mentioned earlier, the P4P and Crowdsourcing paraphrase corpora were created for the task of paraphrase plagiarism detection. To determine how similar they are, we compared them across four dimensions, this include groundtruth, paraphrase features, alteration techniques and variation in intensity of paraphrases computed as variance of proportion of source text present in paraphrase text for all paraphrase instances in a corpus.

Table 3.5: Comparison between P4P and Crowdsourcing paraphrase corpora

	Groundtruth	Features of instances	Formation of paraphrases
P4P corpus	Contains information about whether a pair is paraphrase or not, and details of offset and length of paraphrase and source fragments.	Pairs of source and paraphrase text each comprising of two or more sentences (i.e. passage)	Contains common paraphrases found in plagiarised text such as lexical, semantic and syntactic changes
Crowdsourcing paraphrase corpus	Contains information about whether a pair is paraphrase or not	Pairs of source and paraphrased text each comprising of two or more sentences (i.e passage)	Paraphrase samples contains semantic, syntactic and lexical changes

Table 3.4 contains comparison of the P4P and Crowdsourcing paraphrase plagiarism corpora. The table shows that they both have similar groundtruth, although that of the P4P corpus contains additional details of offsets and lengths of paraphrased text and their sources which are required for post-processing. In terms of features of paraphrase instances, each instance comprises of two or more sentences for a pair of paraphrase and source text, they both have paraphrase samples formed using similar alteration techniques and similar variation in paraphrase intensity across all examples as seen in their variance, which means the distribution of surface features across instances in the corpora are similar. From the above comparison, it is clear that the P4P and Crowdsourcing paraphrase corpora have similar characteristics, and are therefore ideal for evaluating paraphrase plagiarism detection models.

3.3.2 Cross-lingual Plagiarism Datasets

The datasets used to evaluate models proposed in this research for cross-lingual plagiarism detection are the Pan2011 and 2012 evaluation corpora. The two corpora come with detailed groundtruth that includes offset and length of paraphrase and source passages.

3.3.2.1 The Pan 2011 Evaluation Corpus

This corpus contains 5142 manually and automatically generated cases of CLP distributed across 550 suspect documents of which 4709 were generated automatically using internet translation services, and 433 were generated using both automatic and manual correction processes. The automatically generated cases were created using Google translate to translate text passages from one language to another; the process typically involves removing a passage from a non-English source document, translating the passage into English and inserting it in a document written in English. The manually created cases were artificially generated and corrected by humans to appear like real plagiarism cases. The translations are from {Spanish and Danish) to English.

Table 3.6: Statistics on Pan2011 evaluation corpus (cross-lingual plagiarism class)

Nature of plagiarism cases	Number of plagiarism cases
Automatically generated	4709
Automatic + manual correction	433
Total	5142

The table contains statistical distributions of the different cross-lingual plagiarism cases in the Pan2011 corpus, and also the nature of the cases in terms of how they were created.

3.3.2.2 The Pan 2012 Evaluation Corpus

This corpus contains 2500 source documents and 500 artificially generated cases of cross-lingual plagiarism distributed across 500 suspect documents. The cross-lingual plagiarism cases were created using the multilingual europarl corpus; from a non-English source document, a text passage is removed and used to retrieve its corresponding English version from the multilingual europarl corpus. The retrieved English version is then inserted into a Gutenberg book (suspicious documents). The translated passages are from {Danish and Spanish} to English. The lengths of the plagiarised passages are between 75-150 words, and the average similarity between the plagiarised documents and their sources is 0.018 using the cosine measure and *TF* weighting.

Table 3.7: Statistics on Pan2012 corpus

Number of Source docs	No of docs with CLP cases	Number of docs without CLP	Length of plagiarised passages	Translations
2500	500 (20%)	2000 (80%)	75--150 words	De, Es→Eng

The table contains statistical distributions of cross-lingual plagiarised documents in the Pan2012 evaluation corpus. The table also contains addition details such the inter-lingual translations and the minimum and maximum length of the plagiarism cases. The dataset is clearly imbalance with far more non-CLP (80%) examples than examples with CLP cases (20%), it was created this way to mimic reality because real life plagiarism detection is done over the web where there is significantly more non-plagiarised documents than plagiarised ones. Although the difference between the plagiarism classes is not as large as what is expected in reality, the corpus is still representative to a reasonable extent.

3.3.3 Dataset(s) for Monolingual Plagiarism (with varying degrees of alterations) Detection

The dataset chosen for building and evaluating models for detecting plagiarism with varying degrees of intensity in monolingual text is the Clough and Stevenson corpus of short plagiarized answers, this corpus is described below.

3.3.3.1 Clough and Stevenson Corpus of Short Plagiarised Answers

The Clough and Stevenson (2010) corpus of plagiarised texts contains simulated cases of text reuse (plagiarism) carried out by humans (a human corpus). The corpus contains 100 text documents; of which 5 of them are questions, and the remaining 95 are responses to the questions. The corpus was created by issuing out questions to students (respondents) with instructions on how to answer the questions. The questions are original text taken from Wikipedia, and the respondents were given the following instructions; some were asked to copy and paste their answers, some were asked to paraphrase their answers lightly, while others were asked to paraphrase heavily and so on. The collective responses for all five questions resulted in four categories of plagiarism that can be demarcated by levels of intertextual similarity. The categories of plagiarism include cut and paste (verbatim copy), light paraphrased, heavy paraphrased and non-plagiarised. The corpus comes with a ground-truth file that contains the categories in which each response (suspect) document belongs according to human judgment (the creator's judgment).

Table 3.8: Statistics on Clough and Stevenson corpus

Plagiarism categories	Number documents
Cut	19
Light revision	19
Heavy revision	19
Non-plagiarism	38
Total	95

This table shows the statistical distributions of plagiarised documents with different formation plagiarism, as well as non-plagiarised documents in the Clough and Stevenson corpus.

Table 3.9: A summary of the corpora used in this thesis

Corpus	Nature of corpus	Features and suitability
Crowdsourcing paraphrase corpus	Automatically generated + correction by human	Comprises of pairs of text passages that contain paraphrased plagiarism and non-plagiarised texts. Suitable for evaluating models on paraphrased plagiarism detection tasks.
P4P paraphrase plagiarism corpus	Automatically generated + human correction	Comprises of pairs of text passages that contain paraphrased plagiarism and non-plagiarised texts. Suitable for evaluating models on paraphrased plagiarism detection tasks.
Pan@Clef 2011 evaluation corpus	Automatically generated, and automatically generated + human correction	Contains plagiarised texts with varying levels of obfuscation including translation plagiarism and non-plagiarised texts. Suitable for evaluating detection models on both mono and cross-lingual plagiarism detection tasks.
Pan@Clef 2012 evaluation corpus	Automatically generated	Contains plagiarised texts in pairs with varying levels of complexity including translation plagiarism and non-plagiarised texts. Suitable for evaluating detection model on both mono and cross-lingual plagiarism detection tasks.
Clough and Stevenson corpus of short	Simulated	Contains plagiarised texts that have been obfuscated to varying levels of complexity in categories. Suitable for

plagiarised answers.		evaluating a models ability to detect both simple and challenging cases of plagiarism.

This table outlines the corpora used in this research and the nature of plagiarism in them.

Although the paraphrase corpora chosen in this research are old, they are common and reliable, and many of them are still being used in current research to train and evaluate plagiarism detection and text classification models (Devlin et al., 2018; Sánchez-Vega et al., 2019). In addition it is worth mentioning that techniques used in paraphrasing (and plagiarising) text do not really change much over time, which is why old paraphrase dataset are still currently being used, and there are hardly any new reliable plagiarism corpora out there.

Here are the criteria used in this research for corpus selection:

- Corpus must contain pairs of plagiarised and non-plagiarised instances in sufficient amount,
- Groundtruth file must be included,
- Paraphrases must be embedded in text
- Corpus must contain information about the alteration techniques used in creating plagiarised (or paraphrased) instances which should not be different from common techniques used in paraphrasing text,
- Paraphrase instances must show clear semantic similarity with their sources, and non-paraphrase instances must show the reverse.

The next section describes the methods proposed in this research to address research question 1, which investigates the effectiveness of varying combination of surface similarity measurement tools on different formation of plagiarism.

3.4 Methods Chosen to Address Research Question One

This section describes the methods chosen to address research question one, which investigates whether specific combinations of tools used for measuring surface similarity could be determined for detecting plagiarized texts that have been obfuscated to different levels of complexity with performances comparable, and even better than standard baselines.

Hypothesis: Since plagiarized text often contain fragments of unaltered texts that could link them to their sources, it is therefore possible that *with the right combination of tools used for* *measuring surface level similarity which includes similarity measures, ngram textual features and tem weight methods, different formation of plagiarized text could easily be detected, including cases with high degree of obfuscation.*

Description of Methods

The methods chosen to address research question 3 involves evaluating the performance of various combinations of selected surface similarity measurement tools in detecting plagiarized texts that have been obfuscated to varying levels of complexity, and comparing performance against baselines and previous studies. The evaluation metrics used are common in IR for evaluating the performance of retrieval systems, they include precision, recall and F1-score. See section 2.8 for details of these evaluation metrics. For each level of obfuscation, the performance of the surface combination methods and baselines are computed and evaluated.

The surface similarity measurement tools considered in this research are common in plagiarism detection (IR and NLP) for computing similarity between texts. They include ngram document representation models, term weighting methods/schemes and similarity measures.

The method chosen to compute performance involves using each combination model to detect potential plagiarized sentences in pairs of source and suspect texts by comparing sentences in a suspect text with sentences in a source text, and retrieving sentence pairs with maximum similarity scores for every suspect sentence. The similarity scores (at sentence level) are then aggregate to passage level similarity by dividing the sum of similarity scores across all sentences by the length of the number of sentences in the suspect text (containment). A pair of source and suspect passage is detected as plagiarized if their aggregate similarity score exceeds a threshold determined from the dataset.

The process of plagiarism detection often begins with data pre-processing to remove irrelevant elements from texts that may influence the accuracy of similarity computation, and to present texts in uniform comparable format. The subsection below specify the data pre-processing techniques chosen.

Data pre-processing

The data pre-processing tools used to implement the models are common in NLP, they include case folding (normalize text to lower alphabetic case), tokenization, white space and special character removal. The pre-processing steps used when implementing character ngram models were slightly different, only case folding and tokenization were applied on text. On both character and word ngram models, texts passages were tokenized to sentences, this ensures that text comparison is done on sentence level, targeting actual plagiarized fragments.

3.4.1 Surface Similarity Measurement Tools Implemented

The surface similarity measurement tools considered in this research are common in plagiarism detection (IR and NLP) for computing similarity between texts. They include ngram document representation models, term weighting methods/schemes and surface similarity measures.

Here are the specific surface intertextual similarity measurement methods chosen:

3.4.1.1 Ngram Text Representation Models Used

Ngram models were implemented to detect sequence of plagiarized texts; ngrams maintain word order (unlike the bag of word model) and preserve the context of words which improve similarity measurement especially for verbatim and lightly altered texts. Two types of ngrams were chosen in this research, they are character and word ngrams. Word ngrams are efficient and effective for detecting verbatim and lightly altered texts, while character ngrams are slightly less efficient, but effective for capturing verbatim copy and has the advantage over word ngrams of being able to detect sub-word similarity when spelling errors or morphology changes are present.

3.4.1.2 Term Weighting Methods Used

Term weighting methods were implemented to enhance the accuracy of intertextual similarity measurement. Term weighting assigns weights to terms (i.e. words/tokens) based on their relevance in texts, this is important because certain terms carry more information about the semantics (thematic) of text than others. The specific term weighting methods chosen are term frequency (TF), term frequency inverse document frequency (TF-IDF) and binary weighting. The rationale for choosing these weighting methods is that they are common in IR for improving the relevance of information retrieval, they have different characteristics which allows for a comprehensive evaluation of a wide range of term weighting methods so as to determine the most suitable one to optimize the detection of plagiarized texts of a specific obfuscation levels. Term weighting methods are also relevant for transforming surface textual features into vectors that most similarity measures require to work with.

3.4.1.3 Similarity Measures Used

Similarity measures were implemented to estimate the degree of intertextual similarity of feature vectors derived from texts. The chosen similarity measures are Cosine similarity, Jaccard index, Dice coefficient, Bhattacharyya coefficient, Kullback-Liebler divergence and Euclidean distance. These surface similarity measures are chosen because they are common, and have been successfully used in many applications that require texts similarity measurements including plagiarism detection, text clustering, duplicate and near duplicate detection, recommendation systems etc. See section 2.5.2 in the literature for previous studies where these measures were used.

3.4.2 Algorithm Used To Combine Surface Tools for Plagiarism Detection

- START
- Pre-process text by applying case normalization, tokenization, white space removal and stemming.
- Transform pre-processed text into n-gram representation model;
- Vectorize ngram model by assigning weights to terms (n-grams) using a term weighting method (e.g. TF);
- Compare vectors for similarity using a similarity measure (i.e. cosine measure) and compare similarity score with threshold derived from corpus;
- Return text as potential plagiarized if similarity score >=threshold
- END

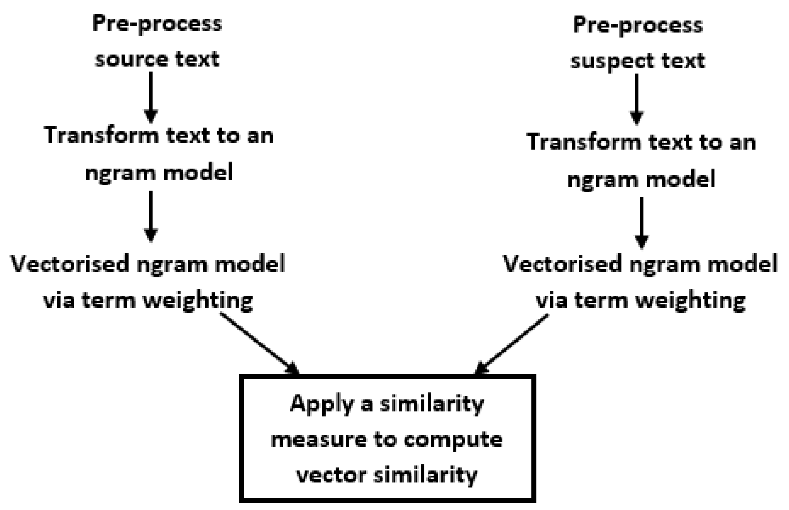

Pre-process source text
Transform text to an ngram model
Vectorised ngram model via term weighting
Pre-process suspect text
Transform text to an ngram model
Vectorised ngram model via term weighting
Apply a similarity measure to compute vector similarity

Figure 3.1: Combination of surface tools for text similarity measurement.

3.4.3 Datasets

The dataset chosen to address research question one is the Clough and Stevenson corpus of short plagiarized answers. The dataset contains plagiarized texts that have been obfuscated to varying degrees of complexity, it is therefore suitable for the evaluation required. Details of this dataset could be found in section 3.3.5.

3.4.4 Baselines

The baseline models used in this work include string matching and semantic methods, they are Ferret (Malcolm et al., 2006) a string matching method commonly used as baseline for evaluating the performance of plagiarism detection systems, and query expansion (using WordNet), which is a semantic method that have been used in the literature of semantic text similarity and plagiarism detection.

Ferret (tri-gram overlap)

The Ferret model was implemented by transforming pairs of source and suspect texts to trigrams and computing the Jaccard index of the trigrams as a measure of similarity. Data pre-processing were carried out on texts before transformation to ngrams; case normalisation and tokenization were applied.

Query expansion (via WordNet)

This baseline was used to detect plagiarized texts that have been obfuscated by replacing words with their synonyms (lexical substitution), studies show that lexical substitution is the most prevalent technique used in plagiarizing texts (Barrón-Cedeño et al., 2013). This baseline was implemented by retrieving synonyms for words in a suspect (query) text from WordNet lexical database, and comparing the suspect text (expanded with synonyms) with its corresponding source text. To resolve word ambiguity, we retrieved all unique synonyms for each word in a paraphrase text which includes different senses of the word, and matched them with source words for possible overlaps. This flexibility allows the WordNet baseline to detect possible matches of ambiguous terms.

Here are the experiments carried out to address research question one.

- Experiment to determine the best performing combination of surface similarity measurement tools for detecting plagiarised texts of different obfuscation complexity.
- Experiment to determine the best performing combination of surface similarity measurement tools for detecting plagiarised texts of a specific obfuscation level.

3.5 Methods Chosen to Address Research Question Two

This section describes the methods chosen to address research question two, which investigates whether deep contextualised learning models (DCLMs) could be used to detect plagiarised texts that have been obfuscated using different paraphrase techniques that are challenging to conventional plagiarism detectors, given their recent success in many NLP tasks.

Datasets: the dataset chosen for this task are the P4P and the Crowdsourcing paraphrase plagiarism corpora. These datasets contain cases of paraphrase plagiarism and are therefore suitable for evaluating paraphrase plagiarism detection systems.

3.5.1 Description of Methods

The approach used to address the problem of paraphrase plagiarism detection involves extending a generic plagiarism detection framework by integrating a DCLM into the framework to detect obfuscation plagiarism that have been altered using different paraphrase types (or expressions such as lexical, syntactic and semantic changes), and evaluating the models performance in the task of paraphrase plagiarism detection.

The generic plagiarism detection framework and the method used to integrate a DCLM into the framework is first described, followed by a description of the chosen DCLMs and the rationale for choosing them. Finally an evaluation framework that describes how the performance of the proposed model was computed and evaluated against standard baselines and a SOTA model using paraphrase plagiarism corpora. Both pre-trained and fine-tuned DCLMs were considered, the evaluation therefore includes fine-tuning the best performing DCLM on a dataset that contains different types of paraphrase plagiarism.

3.5.1.1 Integrating A DCLM into A Generic Plagiarism Detection Framework

This subsection describes the generic plagiarism detection framework and the method used to integrate a DCLM into the generic framework to enhance the detection of paraphrase plagiarism. The main components of the generic framework include data pre-processing, candidate selection, detailed document similarity search (pairwise comparison to detect semantically similar text fragments) and post-processing (merging and filtering). The integration of a DCLM into the framework takes place at the exhaustive document similarity search stage; a DCLM is applied at this stage to detect semantically similar text sequences (sentences) in pairs of suspect and source texts. The candidate selection step is excluded because candidate text are already provided for suspect text in the evaluation corpora. Details of the components of the generic model relevant to this work, the specific methods chosen and rationale for choosing them are described below:

- Pre-processing is usually carried out using NLP techniques to remove unwanted elements from texts that may interfere with detection processes, and to normalise texts to uniform comparable format. Tokenisation of texts to individual sentences is the only pre-processing required here. The reason for tokenising texts to sentences is to ensure that the input text is consistent with the DCLM, given that DCLMs are trained on text sequences, and a sentence is a sequence of words. In addition, sentence level similarity measurements result in more accurate estimation of semantic similarity than passage or document level, and at the same time, it targets actual paraphrase fragments without losing contexts as in word level similarity computation, which makes it even more accurate.

- Text comparison is an exhaustive similarity search process that involves comparing fragments of texts in a paraphrased and source texts for semantic similarity. Comparison is usually carried out either at word, sentence or paragraph level (Gupta, 2016) using a similarity function (i.e. cosine measure) to estimate the degree of semantic similarity between text fragments. This is where CLMs come in, given that they are trained to learn the semantics of text sequences, and generate representations that capture such semantics. CLMs are used here to generate representations for pairs of paraphrase and source sentences that could be used to compute their similarity using a similarity function. The cosine measure is the chosen similarity function because it measures content similarity between vectors in the form of angular distance, and not based on difference in vector length (Cha, 2007; Thompson et al., 2015), so it blends well with CLMs.
- The detection and retrieval method involves using threshold as cut-off to determine whether a pair of sentences should be retrieved as plagiarised or not based on their semantic similarity score obtained from the previous step (i.e. text comparison step). Pairs of sentences with similarity scores that satisfy a threshold requirement are retrieved as paraphrased or potentially plagiarised. Human intervention is usually required to confirm plagiarism (Foltýnek et al., 2020).

Here is how the model works in detecting paraphrase plagiarism:

For a given pair of suspect (potential paraphrased text) and source texts,

- Tokenize text passages to individual sentences using a sentence tokenizer (i.e. NLTK toolkits sentence tokenizer) to be consistent with the input of a DCLM (sequence of text of certain length).
- Retrieve fixed length contextualised embeddings (representations) for sentences in the pair from a DCLM.
- Compare pairs of sentence representations (from source and paraphrased texts) using the cosine measure and return a real number (similarity score in the range of 0 and 1) as a measure of semantic similarity for each pair compared.

- Retrieve all paraphrased sentences with similarity scores that satisfy a threshold (derived from the corpora) as paraphrased and potentially plagiarised.

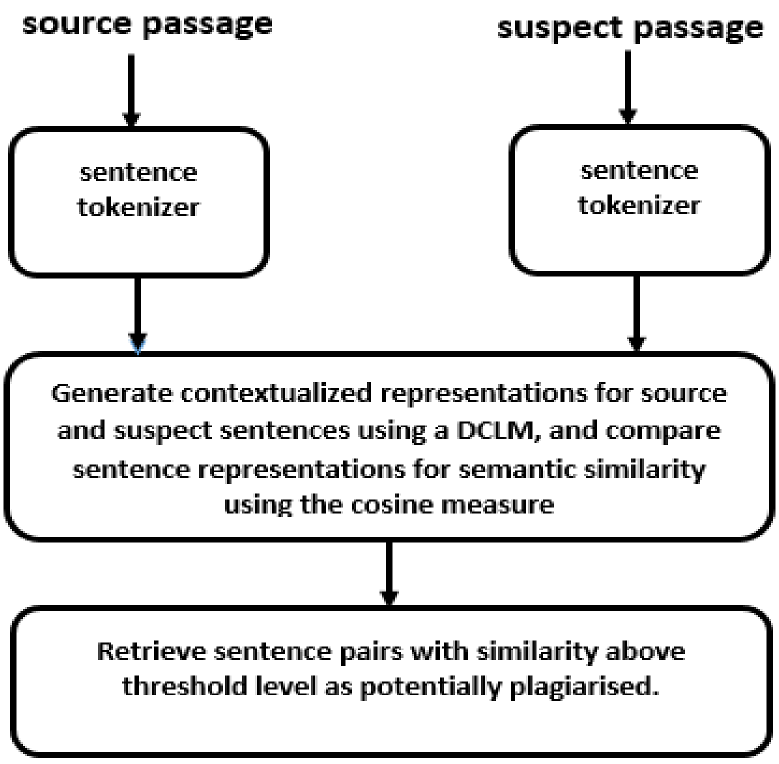

source passage
suspect passage
sentence tokenizer
sentence tokenizer
Generate contextualized representations for source and suspect sentences using a DCLM, and compare sentence representations for semantic similarity using the cosine measure
Retrieve sentence pairs with similarity above threshold level as potentially plagiarised.

Figure 3.2: A DCLM integrated into a generic plagiarism detection model

The final step is to aggregate sentence level similarity into passage level using containment measure and discard passages with similarity below threshold. To go from sentence level to passage (document) level plagiarism detection, we aggregate sentence level similarity into passage level; this is done by dividing the sum of sentence level similarity by the number of unique sentences in the paraphrase text. For example, given a pair of paraphrase and original text (T1 and T2) each comprising of two or more sentences;

T1= {$S_1, S_2..S_n$} and T2= {$S_1, S_2..,S_n$}

$$sim(T1, T2) = \frac{\sum(paraphrase\ sentences\ detected)}{len(T1)}$$

Next are the chosen DCLMs implemented in this research.

3.5.1.2 The Chosen Deep Contextualised Learning Models (DCLMs)

The CLMs chosen in this paper are SBERT (fine-tuned RoBERTa) and ELMo, they are the most common from the two architecture described in the literature. RoBERTa is the most common transformer model, and ELMo is the most common LSTM model used for contextualised learning. CoVe is another CLM implemented in this research, but as one of the baselines.

SBERT (sentence BERT)

SBERT is chosen because it is the most common BERT model for generating sentence embeddings, and does so much faster than the other models which makes it suitable for the task of plagiarism detection that requires real time results, in addition, SBERT was fine-tuned for the task of semantic text similarity, which is at the core of plagiarism detection. In comparison with two other common deep learning models for generating sentence embeddings (i.e. Universal Sentence encoder and inferSent), SBERT significantly outperformed both models (Reimers and Gurevych, 2019) in the task of semantic text similarity.

ELMo (embeddings from language model)

ELMo is a bidirectional LSTM, it is the most common LSTM used for contextualised learning, the original ELMo model was pre-trained on large web data specifically for tasks such as paraphrase identification which is similar to paraphrase plagiarism detection, in addition it can be used to generate contextualized representations for text sequences via linear combination of character embeddings and therefore captures contextualized information similar to the BERT model. Based on the above features, the ELMo model is a good choice of CLM to explore in the task of paraphrase plagiarism detection. The ELMo model used in this research is simple ELMo, a Python library that uses a pre-trained ELMo model trained on 5.5B (billion) tokens of with 1.9B are from Wikipedia and the remaining 3.6B are WMT monolingual news crawl data from 2008-2012; it is the original pre-trained ELMo.

3.5.1.3 Methods Chosen to Fine-tune DCLMs for Paraphrase Plagiarism Detection

This subsection describes the methods chosen to fine-tune the best performing DCLMs for paraphrase plagiarism detection. Fine-tuning typically involves replacing the output layer of a pre-trained DCLM with a task specific layer (Zang et al., 2020) using corpus from the task area, and can result in slight adjustment of the parameters of the pre-trained model to optimise performance in the specific task. The corpus chosen to fine-tune a DCLM for paraphrase plagiarism detection is described below.

Corpus

The dataset used to fine-tune a DCLM for paraphrase plagiarism detection are pairs of paraphrase and non-paraphrase sentences/phrases retrieved from the P4P and Microsoft paraphrase corpora (MRPC); we limit the training set to only pairs that contain at least one sentence with minimum sequence length of 10-characters, this ensures that majority of the text sequences have contexts, and single words that lack contexts were removed. We arrived at a total of 10000 pairs, of which 6000 are paraphrased texts retrieved from the P4P corpus (using annotations in ground truth), and 4000 are non-paraphrased pairs (2100 from P4P corpus and 1901 from the Microsoft paraphrase corpus), and named this mixed training set P4P-MRPC corpus.

Training (fine-tuning) and validation method

A train-test split of 70:30 is chosen to fine-tune a pre-trained DCLM for paraphrase plagiarism detection, this means 70% of the corpus will be used for training and the remaining 30% for testing. Upon completion of the training, the performance of the fine-tuned model is validated on the Crowdsourcing paraphrase corpus.

Fine-tuning BERT for Paraphrase Plagiarism Detection

The method chosen to fine-tune BERT is based on the SBERT model which is designed to generate fixed size vectors for sequence of texts from a pre-trained BERT model.

The SBERT model uses Siamese network comprising of two identical BERT models (heads) to generate embeddings for pairs of text sequences that are passed through a pooling layer that averages the embeddings of the sentences into fixed length vectors, and uses cosine similarity and MSE (mean square error) loss for optimisation.

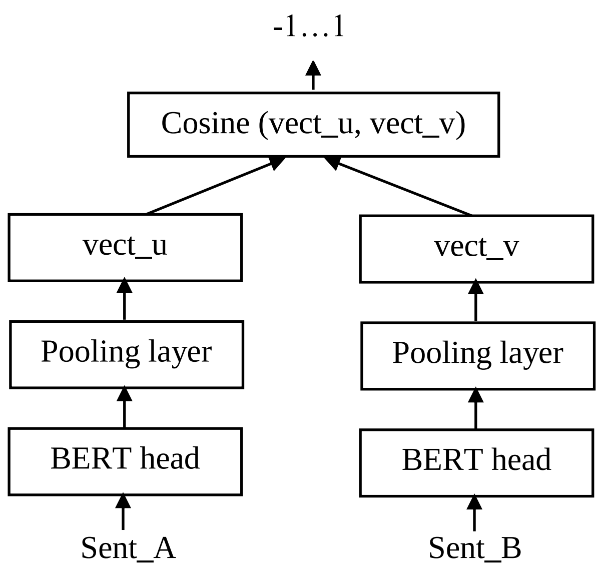


Figure 3.3: Fine-tuning using the SBERT architecture (Reimers and Gurevych, 2019)

We chose to implement this model using the sentence transformer library that contains different SBERT module, the library simplifies the implementation and only requires training data, a pre-trained BERT model and knowledge of hyper-parameters to optimise performance. The rationale behind this method is based on previous studies that revealed that reasonable performance could only be achieved on small dataset if fine-tuning is carried out using a BERT model that has already been fine-tuned on a much larger dataset (Phang et al., 2018; Zhang et al., 2020).We chose distilbert-base-nli-mean-tokens; a sentence transformer model that was fine-tuned for STS using the NLI dataset (over 100,000 training samples) and a pre-trained BERT model known as distilbert (a small but fast BERT model with significantly less parameters relative to other BERT models).

A train-test split of 70%/30% was used to train (fine-tune) and evaluate (test) the pre-trained BERT model, this involves using 70% of the dataset for training (fine-tuning) and the remaining 30% for evaluating the performance of the fined-tuned model.

In terms of parameter turning to optimise performance during training, the hyperparameters of interest are learning rate, batch size, max token length (input length) and number of epoch. The number of hidden-layer stays the same as in the pre-trained model to prevent completely altering the learned weights of the pre-trained model. The method chosen for parameter tuning involves testing different parameter values during training and using a set of parameter values that coincided with the best performance.

Fine-Tuning ELMo for Paraphrase Plagiarism Detection

Fine-tuning an ELMo model is more straight-forward than BERT, it is simply a continuous process of training a pre-trained ELMo model using task specific dataset. The checkpoint of a pre-trained ELMo model that includes learned parameters are used as starting point for pre-training on additional dataset.

ELMo models pre-trained in a number of languages can be found in a repository called AllenNLP, we chose the 5.5B ELMo model pre-trained on English corpus containing 5.5 billion tokens retrieved from both Wikipedia and web crawling sources.

3.5.2 Baseline and SOTA Evaluation Models

The baseline models used in this work is comprehensive, and includes contextualised and non-contextualise word embedding models, common semantic method used in plagiarism detection, and a string (lexical) matching model (the generic model). The SOTA model is a paraphrase plagiarism detection model based on content and stylistic character n-gram features. To ensure the implementation of the baselines are accurate, they were validated on dataset used in previous studies where they were implemented, and their performances compared with what was obtained in those studies. An accurate implementation is one with little or no difference in performance with the original model.

CoVe (contextualised vectors): the CoVe model learns contextualised vectors from a machine translation MT-LSTM trained with word vectors as input. We used a pre-trained CoVe model that uses GloVe vectors as inputs to a MT-LSTM to generate context dependent word vectors for sentences; contextualised vectors for each word in a sentence were retrieved from a CoVe model and averaged into a single representation for the sentence.

GloVe (Global Vectors): the GloVe model uses a matrix of global co-occurrence statistics to learn word vectors that are representative of the meaning of words. Similar to many other studies where GloVe was used as baseline, we represent sentences with their average GloVe vectors, which is the average embeddings of the individual words in a sentence. We used a pre-trained GloVe model downloaded from Stanford University (the 42Billion pre-traind GloVe embeddings) to retrieve embeddings for each word in a sentence, and averaged the embeddings into a single sentence vector.

Cosine similarity (with term frequency vector): Here we represent sentences as vectors of frequency counts that can be compared for similarity using the cosine measure. This baseline is similar to a string matching method commonly used in the STS competition (Agirre et al., 2016; 2017), although the STS baseline is uses binary vectors. It is also similar to the vector space model (VSM) approach proposed for plagiarism detection (Gupta et al., 2016).

Character ngrams (SOTA): this model combines content and stylistic textual features to detect paraphrased plagiarism and outperformed a number of character and string based models, and models based on semantic network (WordNet). Hence, was used as a state-of-the-art model for comparison. The model was implemented as described in Sánchez-Vega et al., (2019). We created 5 different character n-gram representations for pairs of paraphrased and source sentences, all of which were 3-characters in length, they include inner word, between words, prefixes, suffixes, and punctuation character n-grams. To detect plagiarism, we compared the 5-character ngram representations for a paraphrase sentence and those of a source sentence for similarity using the Dice coefficient measure, and obtained a vector of length five (five similarity scores) for every comparison between paraphrased and source sentences. A binary classification scheme based on Naïve Bayes was then used to classify vectors for all sentence pairs as either paraphrased on not.

3.5.3 Performance Evaluation Method

This section describes the method chosen to evaluate the performance of the proposed paraphrase detection model. The method chosen involves computing the models' performance using standard corpora that contains paraphrased plagiarism, and comparing performance against baseline models and a state-of-the-art model proposed in the literature for detecting paraphrased plagiarism. The chosen evaluation metrics used for measuring performance are precision, recall, F1-score and AUC-ROC measure, see section 2.8 in the literature for details of these evaluation metrics.

Here are the experiments required to address research question 1:

- Experiments to determine whether integrating DCLMs in paraphrase plagiarism detection could result in performances comparable to a SOTA model proposed in the literature for paraphrase plagiarism detection.
- Experiments to determine paraphrase types that could be detected with the help of DCLMs, and those that are challenging even with the help of DCLMs.

3.6 Methods Chosen to Address Research Question 3

This section describes the methods chosen in this work to address research question three, which investigates whether the exact language translations of an online machine translator could be captured and used in CLPD without relying on, or limited by internet translation services, and with performances comparable to, or even better than a standard translation plus monolingual analysis (T+MA) model, which is the most common approach used in CLPD but limited by internet translation services.

Datasets: the datasets chosen for this task are the Pan2011 and 2012 evaluation corpora on plagiarism detection. These datasets contain cross-lingual plagiarism cases, and are therefore suitable for evaluating CLPD models.

3.6.1 Description of Methods

To address the research question two, a model for detecting CLP that captures the translation precision of a common online machine translator, without relying on the internet is proposed. The CLPD model follows the standard architecture proposed in previous studies for detecting mono and cross lingual plagiarism (Potthast et al., 2011; Barrón-Cedeno et al., 2013), which include candidate selection, detailed comparison and extraction of plagiarised passages (text alignment), and post-processing. In the retrieval process, a suspect document is tokenised, keywords are extracted and expanded using a multilingual translation model, and the expanded query is used to retrieve candidate source documents from an inverted index built with a collection of source documents. Matched query words (during the candidate selection stage) and their corresponding source words are mapped to the sentences in which they appear in the suspect and candidate source documents respectively for detailed similarity analysis. Sentences with similarity scores above certain thresholds are used as plagiarised fragments to retrieve plagiarised passages. The next section describes the proposed multilingual translation model (MTM) and how it was used in this research to detect CLP.

3.6.1.1 The Proposed Multilingual Translation Model (MTM)

This subsection describes how the MTM is designed to learn the language translation of a common online machine translator and apply them offline without using the internet. The MTM uses word embeddings to capture words and their translations in different languages; it is designed to reproduce the translation of words from an online machine translator when detecting CLP without using internet translators, and to detect semantically similar words (synonyms) by leveraging the potentials of a Word2Vec model in linking similar words that occur in different contexts in an embedding space. In summary, to build the MTM, generate the translations of words in different languages using Google translate (or any other online translation tool), map the words and their translations in a common space and replicate the embeddings to optimise performance, and then train the simulated embeddings using the Mikolov et al., (2013a) Word2Vec CBOW model.

The premise of the model is that similar words occur in similar contexts, if the probability of finding a word in a context is magnified, then higher similarity should be assigned to words that share similar contexts than to non-contextual words (words not in the context). The contexts are simulated, each comprises of a pivot word in English and its equivalent translations in other languages.

Example of a context: {man (Eng), homme (French), mann (Germ), hombre (Spanish)}

The context in the example above consists of the word '*man*' in English and its corresponding translations in French, German and Spanish. Our objective is to maximise the probability of retrieving a context *c* given a word *w*.

$$p(c \mid w) = \frac{\exp(c^T w)}{\sum_{c \in V} \exp(c^T w)}$$ --Equation 3.1

v in equation 3.1 represents the vocabulary of the model.

To create contexts, we used the *top-k* most common English words (based on their frequency) from the British national corpus (BNC; Leech, 1992) and create context for each word by retrieving the translations of the word in Danish, French, German, Spanish etc. from Google translate. The *top-K* could be the *top-10,000* word*s*, but *k* has to be carefully chosen for optimal performance. For optimal performance, the best value for *k* would have to be determined experimentally. Similar to Faruqui and Dyer (2014), the top-*100* most common words are considered too common and non-discriminatory (noisy), hence excluded.

When contexts are created, the final stage is to train a feed-forward neural network, using the back propagation algorithm with stochastic gradient descent to learn the word distributions in the embeddings with the objective of maximising the conditional probability of retrieving an output word given an input context. However training the network this way would result in poor performance (inaccurate predictions) as sufficient data (statistics) is required about each context in order to increase the probability of retrieving a context when a word in the context is searched (similar to the curse of dimensionality).

To achieve higher similarity for contextual words, each context has to be replicated *n*-times. N has to be carefully chosen because each increase in the value of *n* increases the search space exponentially (exponential time complexity $O(V^n)$, v=model vocabulary size). Hence a trade-off between accuracy and computational time has to be carefully resolved. An experiment is required to determine the best value of *n* for optimal performance (Thompson, 2017).

Steps used in building the MTM:

- Retrieve the *top-n* most common English words from the BNC based on frequency
- Create contexts by retrieving the semantic equivalent of each word in other languages using Google translate, which is one of the most common tools used by plagiarist because it is freely available and easily accessible for limited use. Hence building CLPD models to reproduce similar language translations is vital for detecting CLP.
- Replicate each context *n*-times, where *n* is determined experimentally.
- Train the embeddings with the Word2Vec CBOW model; we used Gensim a Python library with the following parameter settings: window size (context) =5, negative sampling=5, minimum word count=50, attributes size =300.

Applying the MTM in CLPD

This sub-section describes how the MTM can be applied to detect CLP in a suspect document given

- a collection of source documents in different languages (require candidate selection)
- a source document in a different language (documents in pairs).

3.6.1.2 Using the MTM to Detect CLP given a Collection of Source Documents

The application of the MTM in CLPD when given a collection of source documents starts with candidate selection; the method chosen for candidate selection is similar to that of Ehsan et al., (2016) in that matching words are searched and mapped back to the sentences in which they appear in source and suspect documents using information retrieval. In this work, the task of candidate selection is performed using the MTM with query expansion and inverted indexing, while in Ehsan et al., key-words/phrases (ngrams of variable sizes between 1-3) are selected in source documents based on *tf-idf* and *tf* scores, and the key-words are translated using Google translate. Details of the method used in this work is as follows:

The MTM is applied in candidate selection through query expansion, query terms are expanded with their translations and used to retrieve potential candidate sources from an inverted index built from a collection of source documents. Query expansion using the MTM involves reformulating a query so that the translations of each query word are retrieved from the MTM. This is possible through vector comparison. The MTM takes a query word and converts it into a word vector and then compares it with word vectors in the model using cosine measure. The outputs from the vector comparison is a list of words and their similarity to the query word, where the most similar words are the translations of the query word in other languages. For example; when the query word *'friend'* is presented to the MTM, the model outputs;

```
'friend': [('ami', 0.99845964), ('freund', 0.99622697), ('amigote', 0.99596620), ('crony', 0.99569368), ('boezemvriend', 0.99561995), ('vriend', 0.99513805), ('amigo', 0.98925573), ('pal', 0.98832619)]
```

French='ami', German='freund', Spanish='amigote', 'amigo', Dutch='vriend', 'boezemvriend'

The output includes synonyms such as *'crony' and 'pal'* in English, *'amigote'* and *'boezemvriend'* in Spanish and Dutch respectively. The model is able to detect related words (synonyms) because semantically similar words do have similar translations, which means they have similar contexts and therefore similar word vectors. When a word is searched, the model outputs the context of the word and similar words in other contexts. This is a key feature of the MTM, and one of advantages it has over online machine translators.

When translations are retrieved for all query words, together they form an expanded query for searching and retrieving potential plagiarised sources from an inverted index. For each candidate document retrieved the original query words (before expansion), and their corresponding matched words in a source document are stored and used for detailed comparison (see figure 1 below) and retrieval of offsets for plagiarised passages.

Detailed Comparison and Extraction of Plagiarised Passages

Detailed comparison is carried out on a sentence level similar to (Pataki et al., 2012); this brings the search for plagiarism closer to plagiarised fragments and makes it easier to extract plagiarised passages from clusters of nearby plagiarised sentences. The method used involves mapping the matching word pairs (from the previous stage) to the sentences in which they occur in the source and suspect documents, and normalising the number of matched words by sentence length. Sentences with similarity scores less than a predefined threshold are discarded. When plagiarised sentences are detected, nearby sentences not more than certain characters apart are merged into plagiarised passages; nearby passages are merged, and passages less than certain characters in size are discarded (post-processing).

Suspect text in English: 'technology is the application of science'

```
Source text in German: 'technologie ist das anwendung von
wissenschaft'
```

```
    1) Tokenize and expand suspect sentence using the MTM
       (exclude stopwords); output: [['la technologie',
       'technologie', 'tecnologia', 'technology',…],
       ['anwendung', 'toepassing', 'app', 'aplicacion',….],
       ['elm', 'la science', 'wissenschaft', 'ciencia',
       'wetenschap', 'science'..,]]
```

2) Query inverted index with expanded query and return matching word pairs in English and German; **output**: ('technology', 'technologie'), (application, anwendung), ('science', 'wissenschaft')
3) Similarity(suspect,source)=len(match-pairs)/len(sentence)=3/3=1

Figure 3.4: A simple application of the MTM in CLPD

3.6.1.3 Using the MTM Model to Detect Cross-Plagiarism in Document Pairs

When documents/texts are provided in pairs (suspect and source), such as in the SemEval STS competition (Agirre et al., 2016; Cer et al., 2017), there will be no need for candidate selection, as suspect documents are paired up with their potential sources.

The MTM detects CLP in document pairs by simply aggregating word level similarity to sentence level, and using sentences with similarity scores above a predefined threshold to map out plagiarised passages.

To compute similarity at word level, the MTM compares pairs of words (in source and suspect sentences) by measuring their word vectors using cosine measure, and outputting a similarity score that ranges between *0 and 1*. Word pairs in a pair of sentences with similarity score above a predefined threshold are retained; the threshold used for word pair similarity is *0.97*.

$$sim(word1,word2) = \cos ine(wordvec1,wordvec2) = \frac{|wordvec1 \bullet wordvec2|}{|wordvec1| \bullet |wordvec2|} \text{ ----Equation 3.2}$$

To compute similarity at sentence level, the number of similar words in a pair of sentences is normalised by the sentence length (containment). Sentence pairs with similarity scores above a certain threshold are used to map out plagiarised passages in the source and suspect documents by retrieving the offset and length of each sentence, and merging nearby sentences not more than certain distance apart into plagiarised passages. The sentences should line up with the documents they appear in.

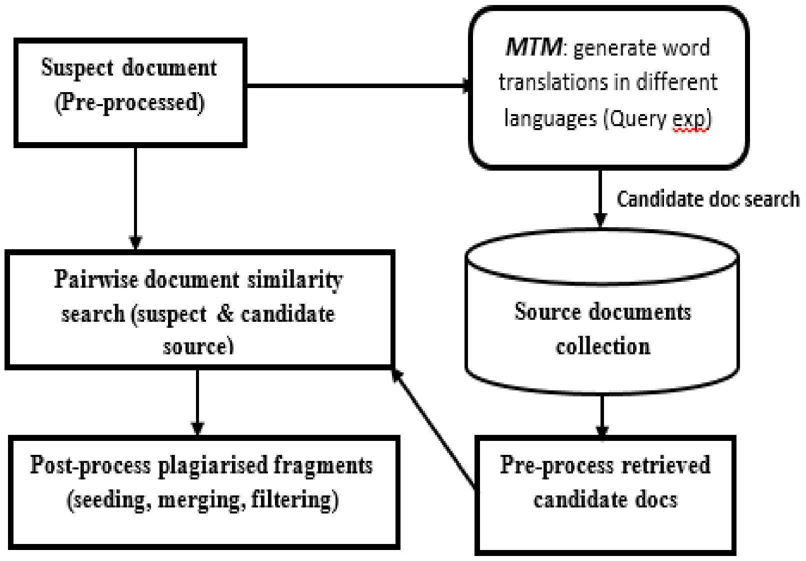

Suspect document (Pre-processed)
MTM: generate word translations in different languages (Query exp)
Candidate doc search
Pairwise document similarity search (suspect & candidate source)
Source documents collection
Post-process plagiarised fragments (seeding, merging, filtering)
Pre-process retrieved candidate docs

Figure 3.5: Proposed CLPD that uses the MTM for language translation

3.6.2 Evaluation method

This section describes the method chosen to evaluate the performance of the proposed CLPD model. The method chosen to evaluate the proposed CLPD model involves computing the performance of the model using datasets that contain CLP, and comparing the performance with a baseline model, as well as with results from previous studies (that made used of the same datasets). The performance evaluation metrics chosen are precision, recall, granularity and plagdet score. These metrics are the standard metrics used for evaluating the performance of plagiarism detection systems. The chosen baseline is the Kasprzak and Brandejs (2010) T+MA model which emerged as the best performing system in the Pan2010 competition on Plagiarism detection, and given that the baseline is T+MA model makes it even more suitable for this evaluation.

Here are the experiments required to address research question 2:

- Experiments to determine the best vocabulary size for the base embedding model to optimise the multilingual translation model.
- Experiments to determine the number of replications required to simulate the embedding space to optimise to optimise the multilingual translation model.
- Experiments to implement and evaluate the performance of the proposed CLP detector against state-of-the-art baseline models.

3.7 Summary

In summary, the methods and approaches chosen to address each of the questions raised in this research were described in this chapter.

Here are the main experiments and tasks discussed:

- Experiments to determine whether integrating DCLM into existing plagiarism detection methods could address the problem of paraphrase plagiarism detection with performances comparable to a state-of-the-art model.
    1) Experiment to implement and evaluate the performance of DCLMs and a number of baselines and a SOTA model in the task of paraphrase plagiarism detection.
    2) Experiment to determine the specific paraphrased types that could be detected when a DCLM is integrated in paraphrase plagiarism, and the paraphrase types that are challenging to detect.

- Experiments to determine whether the exact language translations of an online machine language translator could be captured and used in CLPD without relying on, or limited by internet translation services, and with performances comparable to, or even better than a standard T+MA model.
    1) Experiment to determine the best vocabulary size for the base embedding model to optimise the multilingual translation model.

2) Experiment to determine the number of replications required to simulate the embedding space to optimise to optimise the multilingual translation model.
3) Experiment to implement and evaluate the performance of the proposed CLP detector against state-of-the-art baseline models.

- Experiment to investigate whether a specific combination of surface similarity measurement tools could be determined for detecting plagiarised texts that have been obfuscated to different levels of complexity.
  1) Experiment to determine the best performing combination of surface similarity measurement tools for detecting plagiarised texts of different obfuscation complexity.
  2) Experiment to determine the best performing combination of surface similarity measurement tools for detecting plagiarised texts of a specific obfuscation level.

4 Experiments on Detecting Obfuscation Plagiarism Using Combination of Surface Tools.

The last chapter described the methodology of this research, this chapter addresses research question 1, which is as follows:

- What are the best performing combination of surface similarity measurement tools/techniques (as measured by precision, recall and F1-score) from those described in the literature for detecting similar and near similar text?

  Objective(s): to determine the best combination of surface similarity measurement tools/techniques for detecting plagiarised texts that have been obfuscated to varying degrees.

  Hypothesis: Since plagiarized text often contain fragments of unaltered texts that could link them to their sources, it is therefore possible that *with the right combination of tools used for detecting surface level similarity which include similarity measures, ngram textual features and tem weight methods, different formation of plagiarized text could easily be detected, including cases with high degree of obfuscation.*

To address the above question, several experiments were carried out to evaluate the performance of different combination of surface similarity measures and textual features in the task of detecting similar and near similar texts, and in the context of plagiarism detection. The textual features are different types of ngram document models and term weighting schemes (relevance).

The specific surface similarity measurement tools are:

Similarity measures used

Bhattacharyyan coefficient, Cosine similarity, Dice-coefficient, Euclidean distance, Jaccard-index, Kullback-Leibler divergence and Pearson correlation coefficient.

Term weighting methods and ngrams models used

- Term frequency-inverse document frequency (TF-IDF),
- Term frequency (TF) and
- Binary weighting

- Character and word n-grams

The algorithm used to implement the combination of these surface tools for plagiarism detection is described in the methodology in section 3.6.2. Throughout this chapter, the combination of these surface similarity measurement tools will be referred to as hybrid surface similarity measurement model or HSSMM.

4.1 Experiment: Determination of the Best Performing Combinations of Surface Tools for Measuring Textual Similarity.

4.1.1 Aims/Objectives of Experiment

1. To determine the best performing combinations of surface similarity measurement tools for detecting plagiarized text that have undergone varying degree of obfuscation.
2. To determine the best performing combinations of surface similarity measurement tools for detecting plagiarized texts that have been altered to specific obfuscation level (i.e. light or heavy obfuscation plagiarism).

4.1.2 Datasets

The corpus used for evaluation is the Clough and Stevenson corpus of short plagiarised answers. The corpus contains plagiarised text (or text reused cases) divided into four obfuscation levels (four complexity levels), they include cut and paste (cut and paste), light paraphrased, heavy paraphrased and non-plagiarised (highly dissimilar). See section 3.3.5 for details about the corpus.

Description of Experiments

Several combinations of the above mentioned surface similarity measurement tools were implemented and their performance evaluated in the task of plagiarism detection using the Clough and Stevenson corpus of short plagiarized answers. Each combination is a HSSMM (hybrid surface similarity measurement model) comprising of a similarity measure, a term weighting method and an n-gram document; an HSSMM combines three different dimensions of textual features.

The application of HSSMMs for document comparison was carried out as follow:

Suspect and source documents were pre-processed (case normalization, sentence tokenization, white space removal) and transformed into ngram document models (character and word ngrams) and vectorized by term weighting; binary, TF, TFIDF weighting methods. Document Comparison between suspect and source documents was carried out by vector comparison using similarity measures listed earlier in this chapter.

In determining the best performing HSSMM, we evaluated the performance of different HSSMM created by permutation. The evaluation task involves using an HSSMM model to detect plagiarism in the named corpus by comparing suspect documents with a number of potential source documents, if plagiarism is detected, a suspect document is classified into a plagiarism category depending on the degree of obfuscation (of surface textual features) in the plagiarized text, a suspect document is classified either as cut and paste, light or heavy obfuscation plagiarism category, or as non-plagiarised. An HSSMM is designed to classify plagiarized text with similar features in the same category. Performance was measured in precision, recall and F1-score by comparing the detections of an HSSMM with the groundtruth. A fivefold cross validation was used to determine a robust HSSMM that can generalize well on unknown data; this was carried out by splitting the dataset into 5-equal groups and averaging the performance of an HSSMM model across all groups. We chose fivefold because the dataset is small and to ensure each group is properly represented with sufficient data.

The baseline models used for evaluation are trigram overlap (with Jaccard similarity) and query expansion using WordNet synsets (knowledge based semantic network), the two baselines are standard methods that have been used in the literature to evaluate plagiarism detectors. See section 3.4.4 in the methodology for the implementation of these baselines.

4.1.3 Results and Discussion

This section presents and discusses the results obtained from the evaluation of different HSSMM models in the task of plagiarism detection on the Clough and Stevenson corpus. Tables 4.1-4.4 are the results obtained from the task carried out to determine the best performing HSSMM model across all the obfuscation levels. Tables 4.5-4.6 are the results obtained from the task carried out to determine the best performing HSSMM on specific obfuscation levels. The tables contain performances in precision, recall and F1-score, the tables also contain details of the specific similarity measure, term weighting methods and n-gram document models for each HSSMM model.

Table 4.1: Evaluation Results for the HSSMMs and baselines on the Cut and Paste

Similarity	**Precision**	**Recall%**	**F1-score%**	**N-gram type**	**Ngram size**	**Term weighting**
Bhattacharyya	84.21	84.21	84.21	Word	5	Binary, TF, TFIDF
Cosine	84.21	84.21	84.21	Word	5	Binary, TF, TFIDF
Dice	85	89.474	87.18	Word	5	Binary, TF, TFIDF
Euclidean	70.833	89.474	79.07	char	2	TF
PCC(R)	77.273	89.474	82.927	char	3	TFIDF
Jaccard	85	89.474	87.18	Word	5	Binary, TF, TFIDF
Kullback-Leibler	89.474	89.474	**89.474**	Word	5	TF
Baselines						
Trigram overlap	60	78.947	68.182			
Query expansion (WordNet)	77.778	73.684	75.676			

The results in this table are the highest performance obtained for the HSSMMs and baselines on the cut and paste plagiarism category measured in precision, recall and F1-score.

The highest performance on the cut and paste category is 89.474% (F1-score) as seen in table 4.1, this performance was obtained when Kullback-Leibler divergence (KLD) measure was combined with 5-word n-gram document model and TF weighting. The performance of the HSSMM models on the cut and paste category was quite high relative to the other categories; the second best performance is 87.18%, while the average performance and variance are 84.893% and 11.6921% respectively. The lowest performance on this category is 79.07%, and was observed when Euclidean distance was combined with character n-gram document model of size 2.

Table 4.2: Results of the HSSMM models on the Light Obfuscation Plagiarism

Similarity measures	**Precision**	**Recall%**	**F1-score%**	**Ngram type**	**Ngram size**	**Term weighting**
Bhattacharyya	62.5	78.947	**69.767**	Word	5	TFIDF
Cosine	62.5	78.947	**69.767**	Word	5	TFIDF
Dice	68.421	68.421	68.421	Word	5	Binary
Euclidean	58.824	52.632	55.556	char	3	TF
PCC(R)	57.692	78.947	66.667	char	3	TFIDF
Jaccard	68.421	68.421	68.421	Word	5	Binary
Kullback-Leibler	65	68.421	66.667	Word	5	TF
Baselines						
Trigram overlap	50	52.632	51.282			
Query expansion (WordNet)	60	63.158	61.538			

The results in this table are the highest performance obtained for the HSSMMs and baselines on the light obfuscation plagiarism category measured in precision, recall and F1-score.

The best performance on the light obfuscation plagiarism category is 69.767% (F1-score), this performance score was observed when Cosine measure and Bhattachrryan coefficient were combined with word ngram of size 5 and with TFIDF weighting. Similar to the results obtained on cut and paste category, the performance of the HSSMM models on the light obfuscation category was quite close (with a variance of 24.758% and an average of 66.467%), most of the models have an F1-score that falls within the range of 66.667-68%, although the performance in this category was generally lower than that of the cut and paste category. The lowest performance score in this category is 55.556%, and was observed when Euclidean distance was combined with character n-gram document model of size 3.

Table 4.3: Results of the HSSMM models on the Heavy Obfuscation plagiarism Category

Similarity measures	**Precision %**	**Recall %**	**F1-score %**	**N-gram type**	**N-grams size**	**Term**
Bhattacharyya	64.286	47.368	**54.545**	Word	5	TFIDF
Cosine	64.286	47.368	**54.545**	Word	5	TFIDF
Dice	66.667	42.105	51.613	Word	5	Binary
Euclidean	44.444	42.105	43.243	char	3	TF
PCC(R)	85.714	31.579	46.154	char	3	TFIDF
Jaccard	66.667	42.105	51.613	Word	5	Binary
Kullback-Leibler	50	31.579	38.71	Word	5	TF
Baselines						
Trigram overlap	66.667	31.579	42.857			
Query expansion (WordNet)	64.706	57.895	61.111			

The results in this table are the highest performance obtained for the HSSMM and baseline models on the heavy obfuscation plagiarism category in the Clough and Stevenson corpus, the performance are measurements in precision, recall and F1-score.

The best performance on the heavy obfuscation category is 54.545%, this performance score was observed when Cosine measure and Bhattachrryan coefficient were combined with 5-word ngram document model, and with TF weighting scheme. The performance of the HSSMMs on the high obfuscation category is the lowest relative to the other categories, and more varied; the average performance is 48.6321%, and the variance is 36.89%. There was a general drop in recall on the high obfuscation category, the recall was as low as 38.71% for the least performed model, which is surprisingly not a model formed with Euclidean distance.

Table 4.4: Best performing HSSMMs on Cut and Paste, Heavy and Light Obfuscation Levels

HSSMM	**Precision %**	**Recall %**	**F1-score %**
KLD + 5word ngram + TF	89.474	89.474	**89.474**
Cosine/Bhat + 5word ngram + TFIDF	62.5	78.947	**69.767**
Cosine/Bhat + 5word ngram + TFIDF	64.286	47.368	**54.545**

This table contains results for the best performing HSSMM model on the three obfuscation level in the Clough and Stevenson plagiarized corpus.

Table 4.5: Results from Experiments to Determine the Best Performing HSSMM for Detecting Light Obfuscation Plagiarism

HSSMM model	**Precision**	**Recall**	**F1-score**
Bhat + 5char ngram + TFIDF	65.2174	78.9474	71.4286
Cosine + 5char ngram + TFIDF	65.2174	78.9474	71.4286
Dice + 5char ngram + binary	62.5	78.9474	69.7675
Euclid + 3-char-ngram + TFIDF	58.824	52.632	55.556
PCC(R) +4-word ngram + TFIDF	54.8387	89.4737	68.0
Jaccard + 5char ngram + binary	62.5	78.9474	69.7675
KLD + 5-char ngram + TFIDF	62.5	78.9474	69.7675

The results in table 4.5 are the performance for a number of HSSMM models in the task to determine the best performing HSSMM on light obfuscation plagiarism. The best performance of majority of the models was observed when implemented with 5-character ngrams and TFIDF weighting.

Table 4.6: Results from Experiments to Determine the Best Performing HSSMM for Detecting Heavy Obfuscation Plagiarism.

HSSMM model	**Precision**	**Recall**	**F1-score**
Bhat + 4-char-ngram + TFIDF	57.1429	84.2105	68.0851
Cosine + 3-char-ngram + TFIDF	61.905	68.421	65.0
Dice + 4-char-ngram + binary	56.6667	89.4737	69.3878
Euclid + 3-char-ngram + TFIDF	41.6667	52.6316	46.5117
PCC(R) + 3-char-ngram + TFIDF	43.2432	84.2105	57.1428
Jaccard + 4-char-ngram + binary	57.1429	84.2105	68.0851
KLD + 2-word-ngram + TFIDF	50.0	68.4211	57.7778

The results in table 4.6 are the best performing HSSMM models on the heavy obfuscation plagiarism category. The majority of the models performed their best when implemented with ngrams of sizes in the range of 3 and 4 characters, however the best performance was obtained when Dice coefficient was implemented with binary weighting methods.

Table 4.7: Best Performing HSSMMs for Specific Obfuscation Levels.

HSSMM	Precision %	Recall %	F1-score %	Obfuscation level
KLD + 5-word ngram + TF	89.474	89.474	**89.474**	Cut and paste
Cosine/Bhat + 5char ngram + TFIDF	65.2174	78.9474	**71.4286**	Light
Dice + 4-char-ngram + binary	56.6667	89.4737	**69.3878**	Heavy

The results in this table are for HSSMM models that performed best from evaluation carried out to determine the best performing HSSMM for specific obfuscation levels.

4.1.4 Analysis of the Results

This section analysis the results obtained from the evaluation of the HSSMM models with respect to individual components of the HSSMM models, which include similarity measures, ngrams and term weighting methods.

4.1.4.1 Analysis of Performance Based on Obfuscation Levels

The results in table 4.1-4.4 show that the highest performance was obtained on the cut and paste plagiarism category, this is likely due to absence of obfuscation in the plagiarism cases resulting in large chunks of overlapping fragments. The high performance may also be due to the application of higher order ngram models (long) that can easily capture and discriminate texts with high overlaps. For instance, the chances of having an overlap of up-to five sequence of words (5-gram) in a pair of text is very slim, and when such overlaps occur, they suggest plagiarism. Hence discriminating text that contain cut and paste plagiarism was relatively easy for majority of the models. A closer look at the results revealed that the performance of the HSSMM models decreased with increase in degree of obfuscation (alteration) (see fig 4.1 below)

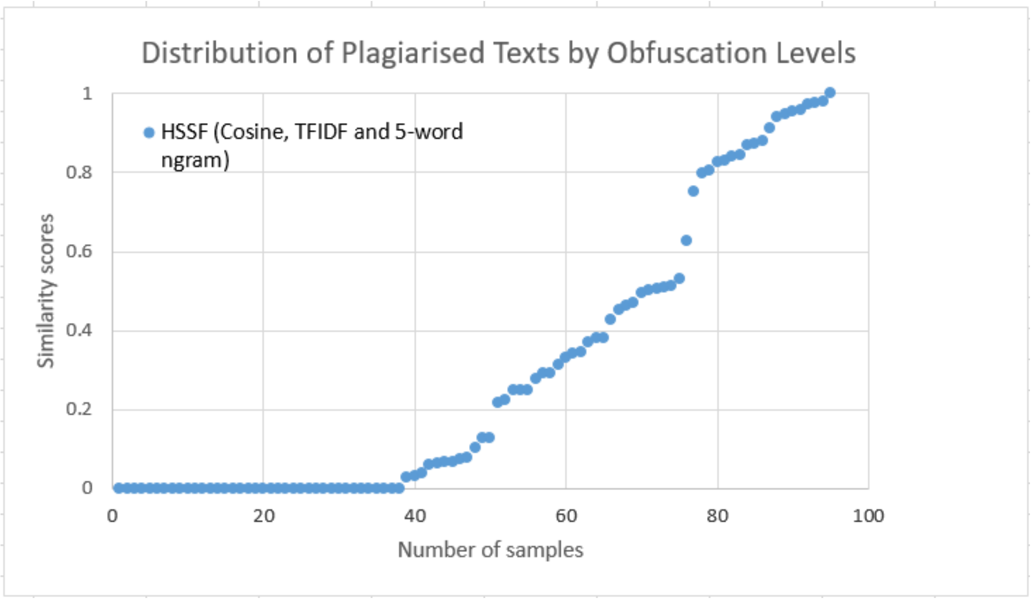


Figure 4.1: Visualization of obfuscation levels in Clough and Stevenson's corpus.

The scatter plot in fig 4.1 shows the distribution of document clusters that represent the four obfuscation levels in the Clough and Stevenson's corpus as classified by an HSSMM (composed of cosine, TFIDF and 5-word ngram).

This trend is consistent with Clough and Stevenson's (2011) findings, and suggests that the more texts are rewritt[3]en the higher the uncertainty in the accuracy of intertextual similarity estimation. In more analytical terms, the progressive decrease in performance can be attributed to increase in the degree of textual alteration (paraphrase); increasing the degree of alteration results in decrease in overlapping textual features available for the HSSMM to work with, this brings about decease in true positive rate, recall and performance. This effect is clearly seen in the performance of the models on the heavy obfuscation category (in table 4.3) where recall for majority of the HSSMM was below 50%, it can also be visualize in fig 4.1, the smallest cluster just above zero represents the heavy obfuscation category, the size of the cluster is indicative of the recall, and in this case, fewer documents that actually belong to that category of plagiarism were detected, relative to the other categories.

The data points in fig 4.1 along the horizontal axis at zero are non-plagiarised documents, as similarity increases above zero are clusters of plagiarized documents in the heavy and light obfuscation categories respectively, and the cluster at the very top is documents in the cut-and paste plagiarism category.

4.1.4.2 Analysis of Performance Based on n-grams and Term Weighting

With respect to n-grams and term weighting methods, the results show that the HSSMM models performed their best on the cut and paste plagiarism category when implemented with higher order n-grams (long) than with lower order ones (see tables 4.1, 4.5, 4.6) and vice versa, this trend is consistent with literature (Sanchez et al., 2019). This means that as the degree of intertextual similarity increases, the size of n-gram required for optimal detection also increases. The results can therefore be used to characterize plagiarism by n-gram sizes; for detecting cut and paste plagiarism, 5-word ngram (or 9-12 characters) is ideal, for light and heavy paraphrased plagiarism, ngrams of 5-character and 3 to 4-character ngrams are ideal respectively. This characterization of plagiarism by n-gram sizes could be used as benchmark (or baseline) for classifying plagiarism cases by intensity or degree of obfuscation.

Regarding term weighing methods, the results revealed that majority of the HSSMM models performed best on the cut and paste category when implemented with binary, TF and TFIDF weighting methods. This implies that when text overlap significantly, such as in cut and paste plagiarism, term relevance becomes insignificant when discriminating such text, a simple overlap technique is enough to capture areas of interest that suggest plagiarism. On the light obfuscation plagiarism category, the results show that the best performance was observed when majority of the models were implemented with TFIDF weighting method as seen in table 6.5. This means that the combination of both local and global term weighting is important for discriminating light and heavy obfuscated plagiarized text, and that certain terms have more discriminating power than others and should be assigned higher weights.

4.1.4.3 Analysis of Performance Based on Similarity Measures

The results in tables 4.1-4.3 revealed that the HSSMM that was implemented with Kullback Leibler divergence (KLD) outperformed the other HSSMM models on the no-obfuscation (cut and paste) plagiarism category. The outstanding performance of KLD in this study is likely due to the fact that KLD measures similarity between pairs of text by comparing probability of distributions of textual contents, this takes into account content similarity, and ensures that all terms in a pair of texts under comparison are considered when measuring similarity (divergence or dissimilarity in this case). In addition, using probability distributions minimise errors when samples are adequately represented such as in cut and paste plagiarism where there is significant overlaps between suspect and source texts. This outstanding performance of KLD is consistent with Huang (2008) and Deng et al., (2019) studies on identifying the best similarity measures to be used with a clustering algorithm, and for Collaborative filtering respectively.

The performance of the HSSMMs implemented with Cosine and Bhattacharrayan coefficient were the best on the light and heavy obfuscation plagiarism category. This is likely due to the fact that they both measure content similarity by applying inner product, although Bhattacharrayan converts text vectors to probability distribution before applying inner product, cosine on the other hand apply inner product directly to vectors and normalize by norm product (vector length). Inner product ensures that similarity is based on angular distance between vectors (topic/content similarity), and not on distance between vector lengths which is easily influenced by outliers. The results imply that inner product which magnifies content similarity is a major determining factor in detecting obfuscation plagiarism as seen in the relatively higher performance of HSSMMs implemented with Cosine similarity and Bhattacharyya coefficient on both the light and heavy obfuscation category. Cosine similarity has always been outstanding in previous studies (Magara et al., 2018; Amer and Abdalla, 2020), but Bhattacharyya coefficient is not well known in the literature of text similarity analysis.

The performance of HSSMM implemented with Dice coefficient and Jaccard index emerged second best even though they both account for content similarity in the form of set intersection. Dice and Jaccard index however do not take term relevance into account as they apply equal weights to all terms (binary), which may have been the likely reason for the small difference in their performance relative to the best performing HSSMMs. In addition, the performance of Dice and Jaccard were similar across all obfuscation levels, this was however expected given how similar the two set theoretical measures are. In terms of HSSMMs implemented with Pearson correlation coefficient (PCC), the performance of PCC was decent across the obfuscation categories, but was however lower than cosine similarity even though PCC is a centered cosine, and as mentioned earlier, a centered cosine is the cosine of a pair of vectors with zero mean, which is equivalent to their PCC. The lower performance of PCC relative to the cosine measure indicates that the mean of vectors formed from the obfuscated texts is often not zero, otherwise the performance of PCC would have equalled that of the cosine measure.

The performance of HSSMMs implemented with Euclidean distance was the lowest in almost of the evaluation tasks carried out, which is consistent with the literature (Strehl et al., 2000; Huang, 2008). One possible reason for the poor performance is that Euclidean distance computes similarity in terms of distances (or differences) between vector lengths, and not based on angular distance. The downside of using distances between vectors as a measure of similarity is that documents that share some terms in common but at significant distance apart, are assigned lower similarity scores than documents that share relatively fewer words but at relatively less distance apart. This implies that Euclidean distance does not compute similarity on the basis of textual content, and therefore prone to error. One other possible reason for the poor performance is that Euclidean distance is sensitive to outliers (extremely high or low vector components). A term with extremely high weight in a document has strong influence on similarity, and can easily result in false positive. Jones and Furnas (1987) described this effect as the single component influencing monotonicity. It is worth pointing out that normalised Euclidean distance (using z-score) was implemented in this research to tackle outliers, and preliminary experiments revealed better performance in comparison to traditional Euclidean distance. However, it is obvious from the results that normalising Euclidean distance does not completely eliminate the effect of outliers in vector comparison.

4.1.4.4 Comparison with Baselines.

In comparison with baselines, the results show that the best performing HSSMM (cut: 84.2%, light: 69.767%, heavy: 54.545%) clearly outperformed the tri-gram overlap baseline model (68.182%, 51.282%, 42.857%) on the three levels of obfuscation as seen in tables 4.1-4.3. With respect to the WordNet/query expansion baseline (75.676%, 61.538%, 61.111%), the HSSMM performed better on two out of the three obfuscation categories (cut and paste and light obfuscation plagiarism category). However, on the heavy obfuscation category, the performance of the WordNet model was higher. Further analysis on the heavy obfuscation category using AU-ROC revealed little difference in performance between the HSSMM (AUC: 0.704) and the WordNet/query_exp (AUC: 0.719). The AUC results of the models is consistent with the F1-score, imply that the query_exp model is slightly better at discriminating heavy obfuscation plagiarism than the HSSMM.

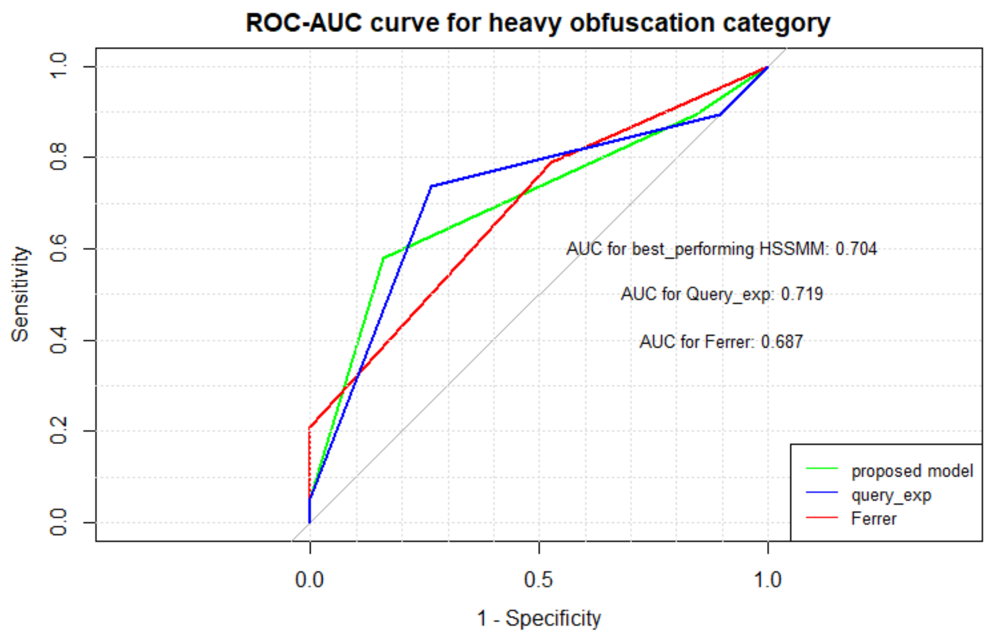


Figure 4.2: AU-ROC curves for best performing HSSMM and baselines on the heavy obfuscation category.

The likely reason for the superior performance of the WordNet model over the HSSMM on the heavy obfuscation category is that, the textual alterations carried out on the heavy obfuscation category involved replacing words with their synonyms, which is exactly what the WordNet baseline model is designed to detect using query expansion. It is worth noting that semantic networks such as WordNets are used primarily for semantic similarity measurement tasks, such as the task of detecting heavy obfuscation plagiarism. This therefore reinforces the need for the inclusion of semantic methods into conventional plagiarism detection, which is one of the questions addressed in this research.

4.1.4.5 Comparison with Results from Previous Studies

In comparison with results from previous studies, the proposed HSSMM outperformed Bär et al., (2012) and Clough and Stevenson (re-implementation) models as seen in figure 4.10. To the best of my knowledge, the results from Bär et al., represent the state-of-the-art on this corpus.

Table 4.4.8: Macro average for the best performing HSSMM when light and heavy obfuscated categories were merged.

Class Metrics	Precision	Recall	F1-score
Cut and paste	0.85	0.895	0.872
Light + heavy	0.968	0.79	0.87
Non-paraphrased	0.881	0.861	0.871
Macro-average	0.9	0.849	0.871

[4]Merging was carried out to ensure consistency with results from previous studies.

Table 4.4.9: Confusion matrix for the best performing HSSMM when light and heavy obfuscated categories were merged.

	Cut	Light + heavy	Non-plagiarised
Cut	17	5	0
Light + heavy	0	30	1
No-plagiarised	2	3	37

Table 4.4.10: Results from best performing HSSMM and previous studies

	Accuracy	F1-score (macro-average)
HSSMM (our model)	0.884	**0.871**
Bär et al., (2012)	0.884	0.859
Clough and Stevenson (2012)	0.821	0.788

The HSSMM performed comparably to the Bär et al., (2012) with similar accuracy but slightly better in terms of macro-average F1-score as seen in table 4.1. Both models clearly outperformed the Clough and Stevenson method.

The Clough and Stevenson method combined features from ngrams of various sizes with the longest common subsequence (LCS) using a machine learning classifier (Naïve Bayes classifier), while Bär et al., model combines features across a number of dimensions including stylistic, content and semantic dimensions. In spite of how complex Bär et al., model is, the best performing HSSMM performed comparably and even better as seen in their accuracy and macro-average respectively (macro-aver: HSSMM—0.871, Bär et al.,---0.859, Clough and Stevenson—0.788). While the HSSMM and models from the previous studies all apply some form of string matching using ngrams and similarity measures, the strong performance of the

The performance of the HSSMM on the obfuscation (light and heavy) class significantly increased as a result of merging the two classes. This revealed that majority of the misclassification take place between the heavy and light obfuscation categories, which is a reflection of the dataset, and implies that the boundary (separation) between the light and heavy obfuscation categories in the Clough and Stevenson corpus is very thin.

HSSMM is likely due to the application of the most suitable term weighting method in combination with the best of the other surface tools.

4.2 Discussion

In relation to the research objectives, the results revealed that no single combination of surface similarity measures and textual features performed best on all the obfuscation levels present in the corpus as seen in table 4.4; HSSMM based on KLD + 5-word ngram + TF emerged best for detecting cut and paste plagiarism, but fell short on the other obfuscation categories, while HSSMMs based on Cosine/Bhat + 5-word ngram + TFIDF emerged best on the light and heavy obfuscation plagiarism category, but fell short on the cut and paste category. However, given that it is more challenging to detect light and heavy obfuscation plagiarism, it is only rationale to recommend Cosine/Bhat + 5-word ngram + TFIDF for detecting obfuscation plagiarism of all levels. In terms of the hypothesis, the results show that the best performing HSSMM performed comparably and in many cases better than established methods. This means that with the right combination of surface similarity measures and textual features, better performance could be achieved in the task of detecting plagiarized texts with different degrees of obfuscation, the hypothesis is therefore accepted.

Regarding the objective to determine the best performing HSSMMs for specific obfuscation level, the results show that KLD + 5-word ngram + TF is best for cut and paste, Cosine/Bhat + 5-word ngram + TFIDF is most suitable for detecting light obfuscation plagiarism and Dice + 4-char-ngram + binary is most suitable for light obfuscation plagiarism as seen in table 4.7. A common trend seen in the performance of the HSSMMs is a decrease in performance with increase in obfuscation levels from cut and paste down to the heavy obfuscation category as seen in tables 4.1 to 4.3. Heavy obfuscation plagiarized text were the most challenging to detect as they contain relatively few overlaps for surface similarity measurement tools to work with, and majority of the HSSMM performed their best in this category when implemented with character n-grams and TFIDF. The performance of the HSSMMs were for the most part better than the baselines and results from previous studies, however the WordNet baseline model which is designed for semantic similarity outperformed the best performing HSSMM model on the heavy obfuscation category as seen in table 4.3. This means integrating an appropriate semantic similarity method could enhance the performance of the best performing HSSMMs in detecting heavy obfuscation plagiarism.

4.3 Conclusion

Experiments were carried out to determine the best performing combination of similarity measures and textual features in the task of detecting plagiarized texts that have been obfuscated at different levels (degrees). The objectives of the experiments were to determine whether a specific combination of surface similarity measurement tools/techniques could be determined for detecting obfuscation plagiarism of varying complexity. Experimental results revealed that no single combination performed best on all the levels of obfuscation plagiarism experimented with. Some combinations performed well on some obfuscation type, but not so well on others, while some combinations performed averagely across all categories. A single combination was recommended that performed best on majority of the obfuscation levels, especially on the challenging ones. Recommendations were also made concerning the best combination to use for specific obfuscation levels. In comparison with the baselines, many of the combinations clearly outperformed the baseline models, however one of the baselines which is designed for semantic similarity measurement outperformed the best combination model on the most challenging obfuscation level, and reinforces the need to enhance plagiarism detection using semantic similarity measurement tools.

The results obtained from the experiments clearly support the hypothesis which states that; with the right combination of tools used in estimating surface level intertextual similarity, both simple (cut and paste) and to a large extent difficult cases of plagiarism (altered cases) could be detected. This finding is particularly useful considering the fact that the tools used for surface level similarity measurement (such as similarity measures) are relatively easy to implement and compute similarity much faster than methods that rely on external resources (semantic methods), which include knowledge or corpus based methods (WordNet and word embeddings i.e. Word2Vec respectively). Unlike semantic methods, surface similarity methods do not require lookups on external resources that slows down detection processes, and plagiarism detection especially in academic environment require systems than are fast enough to keep up with large number of students and scalable enough to deal with huge assignment materials such as projects and dissertation. These requirements are easily satisfied with surface similarity methods, but surface similarity measures alone cannot deal with the increasing complexity in obfuscation techniques device by plagiarist, especially challenge in detecting heavily paraphrased plagiarism. Although the dataset used in the experiments is limited in size, there were no better datasets out there to use, the dataset has also been commonly used in many studies on plagiarism detection and text reuse. Due to the size limitation of this dataset, the conclusion drawn here is subject to further evaluation and analyses using much larger dataset as they become available, or a research to create much larger dataset of the same structure may be undertaken in the future. Future work will also be focused on combining the best performing combination models into a hybrid that works best across all the obfuscation levels, and also a hybrid that integrates a little semantics to enhance the performance (effectiveness) of surface similarity methods.

5 Experiments on Paraphrase Plagiarism Detection

5.1 Introduction

The last chapter described experiments carried out to detect different formation of plagiarism using combination of surface detection tools. This chapter addresses research question two which is stated as follows:

- Can deep contextual learning models be used to enhance the detection of paraphrase plagiarism with performances comparable or better than a state-of-the-art (SOTA) model?

Aim(s)/Objective(s)

- To determine whether the application of a deep contextualised learning (DCLM) in paraphrase plagiarism detection could result in performances comparable to, or even better than the current state-of-the-art.

To address the above research question, this chapter describes experiments carried out to evaluate the performance of a proposed paraphrase detection model described in the methodology in section 3.5.1.1. The proposed model is an extension of a generic plagiarism detection framework by integrating DCLM into the framework. We evaluated two implementations of the proposed model using two common DCLMs described in the methodology in subsection 3.5.1.2, using corpora that contain paraphrase plagiarism, and with standard evaluation metrics used in information retrieval (IR) and NLP for measuring the performance of plagiarism detection systems. We re-implemented the best performing model with a DCLM fine-tuned on a dataset that contains paraphrase plagiarism and evaluated its performance. We compared our model with baseline systems, a state-of-the-art model and with the generic plagiarism detection framework upon which our model was built. We also carried out additional experiments to determine the types of paraphrases plagiarism that could be detected with our model and those that are challenging or difficult to detect even with the help of a DCLM.

5.2 Experiment 1: Performance Evaluation of Proposed Paraphrase Plagiarism Detection Model

This experiment was carried out to implement and evaluate the performance of the proposed paraphrase plagiarism detection (PD) model when implemented with the chosen DCLMs on datasets that contain paraphrase plagiarism, and to make comparison against baselines and a SOTA model.

5.2.1 DCLMs Implemented With the Proposed Model

- SBERT (Sentence BERT, a fine tuned RoBERTa model with a bidirectional transformer architecture).
- ELMo (Embeddings from language model, a bidirectional LSTM architecture)

5.2.2 Aims/Objectives of Experiment

- To determine how well the proposed paraphrase plagiarism detection model would perform relative to a SOTA paraphrase plagiarism detector, and standard baselines.
- To determine the best performing DCLM from the chosen models that could be used in the proposed paraphrase plagiarism detection model.

5.2.3 Datasets

The datasets used in the experiments are the P4P and the Crowdsourcing paraphrase corpora, they both contain cases of paraphrase plagiarism and are therefore suitable for this particular evaluation, see section 3.3 in the methodology for details about these datasets.

Description of Experiments

The proposed plagiarism detection model (PPDM) was implemented using the two DCLMs mentioned above (in subsection 5.2.1) and performance was evaluated in the task of paraphrase plagiarism detection (PPD) using corpora that contain paraphrase plagiarism (the P4P and Crowdsourcing paraphrase corpora). The evaluation metrics used are precision, recall, F1- and AUC-ROC, see section 2.8 in the literature for details of these evaluation metrics.

The evaluation was carried out using a 10-fold cross validation scheme; the evaluation corpus was split into 10-equal groups, and performance was computed on each group and averaged across all the groups. This was done to avoid overfitting so that a realistic (and robust) model that generalises well on unknown dataset is achieved.

We carried out further evaluation to determine whether a DCLM that is fine-tuned on a paraphrase plagiarism dataset could outperform the other models. We fine-tuned the best performing DCLM using dataset that contains paraphrase and non-paraphrase sentence pairs (the P4P-MRPC). The model was fine-tuned as described in section 3.5.1.3; 70% (7000 examples) of the dataset was used for training and the remaining 30% (3000 examples) for validation (to avoid overfitting). Training was carried out by running the model on the training set while evaluating its performance by testing different hyper-parameter settings; search space was defined for each hyper-parameter, at every 500-training cycle the model was evaluated on the validation set, and the model with the best performance (using accuracy and loss metrics) was saved and integrated into the generic PD model, and evaluated on the Crowdsourcing paraphrase corpus.

Hyper-parameters for the best performing model are; Adam optimiser with a learning rate=0.001, epoch=1, batch size= 16 and max token length=512, and as stated earlier, the number of hidden layers remained the same to avoid completely altering the weight of the pre-trained DCLM.

Implementation and validation of baselines and SOTA

The baselines and SOTA model were also implemented and their performances evaluated. The implementation of each baseline was validated on the corpus used in the original study they were proposed, the validation was carefully done to ensure that their performance (baselines) was consistent with that of their original.

CoVe baseline: we used the original pre-trained CoVe model (McCann et. al. 2017), which is an implementation MT-LSTM in Pytorch to generate contextualized sentence vectors that can be compared for semantic similarity using the cosine measure. Upon validation, we obtained the same results as the original implementation on text entailment using the SNLI (Stanford natural language inference) corpus (Bowman et al., 2015).

Average GloVe baseline: this baseline was implemented using 42Billion pre-trained GloVe embeddings downloaded from Stanford university website, the pre-trained model was used to generate word vectors that are averaged across all words in a sentence into a single vector representation for the sentence that can be compared with other sentence representation for semantic similarity. The transformation of sentences to average GloVe embeddings was carried out as follows; given a sentence from a passage, we first tokenised the sentence into words and transformed the words into GloVe vectors (using a pre-trained GloVe), we then concatenated (dimension wise) and averaged the word vectors along each dimension into a single sentence embedding (vector). With regards to validation, we obtained the same performance with the implementation of Reimers and Gurevych (2019) on the STS benchmark dataset (Cer et al., 2017).

Cosine (with frequency vector)/generic model: The generic model was obtained by transforming sentences into frequency vectors that can be compared for semantic similarity using the cosine measure. Short sentences with less than 3-words are joined to proceeding sentences as in (best model). We validated this model by implementing it with both binary and frequency vectors and obtained similar results to Agirre et al., (2016) on the STS monolingual plagiarism evaluation subset.

State-of-the-art model (SOTA): the implementation of the SOTA model was carried out using 5-character ngram features to capture content and stylistic textual features in sentences that could be compared with other sentences for semantic similarity. See section 3.5.2 for details of this implementation. Our implementation was validated on a subset of the P4P corpus used in (Sanchez et al., 2019) and we obtained the same result.

5.3 Experiment 2: Determination of the Performance of DCLMs on Different Paraphrase Types.

This experiment was carried out to determine the relative performance of DCLMs with respect to paraphrase types in plagiarised texts.

5.3.1 Aim(s)/Objective(s)

- To determine the paraphrased types that could be detected with the DCLMs and those that are challenging or difficult to detect even with the help of DCLMs.

5.3.2 Corpus

The P4P corpus was used in this experiment, it contains sentences that are annotated with paraphrased types, and hence it is suitable for this evaluation.

Evaluation of performance with respect to paraphrase types

To evaluate the performance of the DCLMs with respect to paraphrase types, the proposed paraphrased plagiarism detection model was implemented in the same way as in experiment 1 (in section 5.2), and thei'r performances were measured at the point of intersection of precision and recall (the equilibrium point where precision equals recall). The threshold at the intersection was used as cut-off to retrieve all detected paraphrased text, and performance was measured in recall with respect to specific paraphrase types (i.e. proportion of each paraphrase type retrieved). Recall was chosen because the objective is to determine the proportion of each paraphrase type retrieved at the optimal performance point (intersection of pr/rc) of each model, which is the same as recall.

Results from the Experiments on the P4P Corpus

This section presents the results obtained from the experiments carried out to evaluate the performance obtained for the proposed model and baselines on the P4P- paraphrase corpus.

Table 5.1: Results on Proposed Model and Baselines on the P4P Corpus.

Model	Precision	Recall	F1-score	AUC
Proposed model with SBERT	0.807	0.795	0.801	0.890
Proposed model with ELMo	0.796	0.755	0.775	0.875
	Baselines			
CoVe *(*McCann et al., 2017*)*	0.750	0.778	0.764	0.860
Average GloVe	0.751	0.772	0.762	0.845
Cosine/frequency vector (generic framework)	0.568	0.577	0.572	0.685
	SOTA			
Character features (Sánchez-Vega et al., 2019)	0.879	0.601	0.714	0.835

The results in this table are the performance of the proposed model, baselines and a SOTA model in precision (pr), recall (rc), f1-score (f1) and AUC based on 10-fold cross validation.

The results in table 5.1 show that for the context learning models (SBERT, ELMo and CoVe), SBERT (F1-score: 0.801) outperformed ELMo (F1-score: 0.775) and CoVe (F1-score: 0.764) as seen in their F1-scores, while ELMo outperformed CoVe. Overall, SBERT performed best, and ELMo comes second best; the two contextualised learning models under investigation (SBERT and ELMo) outperformed the baselines. Statistical test for significance using paired t-test revealed significant difference in performance between the SBERT model and the baselines (P<0.05); this implies that in 95% or more of the time the SBERT model will outperform the baselines. This is supported by the AUC which show higher values for the DCLMs, particularly for the SBERT model, which means the SBERT model is able to separate the paraphrased plagiarism examples from the non-paraphrased ones much better than the baselines, and has a higher probability in classifying a random example into the right class. The results imply that SBERT is able to detect the type of obfuscation associated with paraphrase plagiarism much better than the other models, which means that SBERT is more effective in learning semantic, syntactic, lexical and miscellaneous relationships embedded in paraphrased texts than the other models. The results also imply that the transformer architecture (of BERT) is better in contextualised learning than the LSTM (in ELMo and CoVe).

A number of factors may have contributed to the difference in performance between SBERT and ELMo, two of the most likely reasons are the difference in training objectives of the models, and in the way input sequences are processed and used during training for contextualised learning. While both models use bidirectional learning and attention mechanism, they are however trained using different learning objective methods, SBERT (fine-tuned RoBERTa) is trained with MLM objective, while ELMo is trained with traditional language modelling objective (next word prediction) as in LSTMs. As stated earlier, in MLM a model learns to predict a number of randomly selected tokens of an input sequence (about 15%), this exposes the model to a wide range of relationship between input tokens and results in deeper representation learning, as opposed to next word prediction where a model learns to predict only the next word given the previous words in a sequence of tokens, and therefore not able to learn the relationship between different combination of input tokens. This relationship could be lexical, semantic, syntactic or miscellaneous, which are core characteristics of paraphrased texts. Another possible reason is the differences in which input sequences are processed during training. Transformers are designed to process an entire sequence of text during training, and therefore avoids information loss, while LSTMs use gates to selectively retrieve a limited amount of previous information for next word prediction learning, giving room to information loss from long dependencies (Vaswani et al., 2017).

The contextualised and non-contextualised learning models (SBERT, ELMo, CoVe and GloVe) which are designed to detect semantic similarity at word and sequence (i.e. sentence) levels clearly outperformed the string/lexical matching methods (Cosine and SOTA) that rely mainly on surface content features. This findings is consistent with the literature that suggests that detecting paraphrased or high obfuscated plagiarism require models that are designed for semantic text similarity (Gupta, 2016; Foltýnek et al., 2019).

It is worth noting that the recall of the generic model (cosine with frequency vector) is obviously lower than the other models which is likely due to absence of matching strings (overlaps), it may also be due to the diverse types of paraphrases in the corpus, because even when significant overlaps (matching strings) are present, they may not necessarily mean that pairs of texts are semantically similar, for example, the sentences below have significant matching terms but are not semantically similar;

a) The quick brown fox jumps over the fence.
b) The slow brown fox did not jump over the fence.

The paraphrases in the P4P dataset may also have significantly reduced the ability of string matching models to detect semantic similarity, this is also seen in the performance of the SOTA model, although to a lesser extent.

Table 5.2: Performance of the Models Different Paraphrase Types

Paraphrase types	**SBERT**	**ELMo**	**CoVe**	**Aver_Glove**	**Cosine/freq. vect**	**Num. of samples**
dis_direct_indirect	0.583	0.694	0.556	0.278	0.361	36
dis_punct_format	0.833	0.849	0.842	0.755	0.697	538
dis_sent_modality	0.857	0.914	0.829	0.714	0.571	35
Insert_delete	0.9	0.89	0.879	0.727	0.618	1576
lex_converse	0.939	0.879	0.909	0.667	0.606	33
lex_opposite_polarity	0.877	0.846	0.815	0.662	0.662	65
lex_same_polarity	0.920	0.926	0.884	0.772	0.617	5071
lex_spelling_and_format	0.892	0.867	0.909	0.746	0.735	437
lex_synt_ana	0.945	0.928	0.919	0.828	0.680	669
mor_derivational	0.958	0.935	0.935	0.759	0.582	261
mor_inflectional	0.913	0.929	0.925	0.78	0.63	254
mor_modal_verb	0.853	0.888	0.862	0.647	0.612	116
order	0.915	0.905	0.894	0.835	0.767	576
semantic	0.862	0.829	0.761	0.538	0.453	340
syn_coordination	0.824	0.857	0.833	0.671	0.595	210
syn_diathesis	0.969	0.915	0.869	0.769	0.662	130
syn_dis_structure	0.955	0.946	0.901	0.738	0.658	313
syn_ellipsis	0.99	1.0	0.954	0.92	0.782	87
syn_negation	1.0	0.939	0.849	0.788	0.576	33
syn_subord_nesting	0.93	0.908	0.91	0.784	0.663	597
Variance	**0.00795**	**0.0039**	**0.0074**	**0.01729**	**0.00939**	
Average	**0.896**	**0.892**	**0.862**	**0.72**	**0.626**	

The results in this table are performance in *recall* for paraphrase types in the P4P corpus taking at the threshold point where pr/rc intersect as seen in fig 5.1 and 5.3.

Although there were some difference in performance between SBERT and ELMo as seen in table 5.2 where SBERT outperformed ELMo on certain paraphrase type, and the same goes for ELMo on other paraphrase types; statistical analysis to measure dispersion (variability) in performance across paraphrase types revealed that among the contextualized learning models SBERT has the highest variance, while ELMo has the least variance (ELMo**:** 0.0039, CoVe, 0.0074, BERT: 0.008 (0.00795)**).** This means the performance of the SBERT model vary across paraphrase types more than the other contextualized learning models, and implies that the SBERT model is more sensitive to changes in paraphrase types, while the ELMo model is least sensitive. In general, the variability in performance of the contextualized learning models was less than the baselines (non-contextualised learning models); lower variability in performance could be interpreted as better stability and consistency in performance irrespective of paraphrase types, although more study may be required to establish this fact. With regards to specific paraphrase types, the analysis of the performance of the models is largely focused on DCLMs because of their superior performance overall as seen in Fig 5.1 below. The results in table 5.2 revealed that representations from DCLMs could be used to enhance the detection of the following paraphrase types with relatively higher probability; *mor_derivational, syn_diathesis, syn_dis_structure, syn_ellipsis and syn_negation*, all of which have recall above 0.949 as seen across the models. While *syn-coordination, dis-direct-indirect and dis-punct-format*, all of which have recall lower that 0.85 appear to be challenging to the DCLMs, with dis-direct-indirect being the most challenging, even to the other models evaluated. Paraphrase types with recall that lie between the above thresholds appear to neither difficult nor easy to detect. These threshold values are hypothetical, but points to the relative strength of DCLMs with respect to paraphrase detection, and where improvement could be made so as to build DCLMs that can generate good quality representations for better inferences on a wide range of NLP tasks, including paraphrase plagiarism detection.

Table 5.3: Paraphrase types in the P4P Corpus Divided into Six Groups

Paraphrase class	Generic features	Paraphrase types
Morphological changes	All changes that affect the form in which a lexical unit appear in text.	Inflectional, derivational and modal verb changes
Lexical changes:	All changes that involves substituting one lexical unit with another, or altering the structure of a unit.	Spelling and format changes, converse, opposite-polarity, synthetic/analytic and same-polarity substitutions.
Semantic changes	All changes that involves rewriting portion of text without changing the meaning of the text.	Semantic alteration
Syntax based changes	All changes that affect the structure of texts such as moving lexical units around, they include coordination changes, subordination and nesting changes, ellipsis, negation switching and diathesis.	Coordination changes, subordination and nesting changes, ellipsis, negation switching and diathesis.
Discourse changes	All changes that affect the style, format, or mode in which text is presented,	punctuation and format changes, direct/indirect
Miscellaneous changes	All changes to text that involves reordering, insertion or deletion of one or more lexical units.	Insertion/deletion, reordering

This table contains the paraphrase types in the P4P corpus, their groupings and respective generic features.

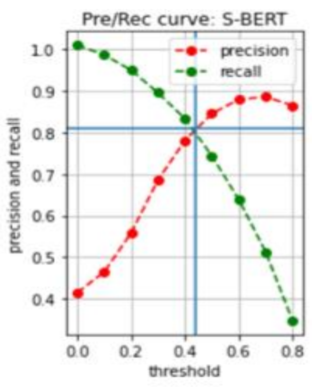


Fig 5.1a: precision/recall curve for SBERT

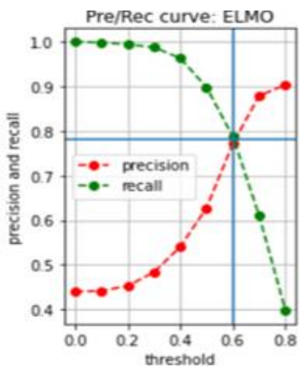


Fig 5.1b: precision/recall curve for ELMo

Figure 5.5.1: Precision/Recall curves for SBERT and ELMo across all possible thresholds. Each curve shows the point where precision and recall intersect, and the threshold at that point.

The pr/rc curves in fig 4.1 show higher performance for SBERT (F1: slightly above 0.8) than ELMo (F1: slightly below 0.8) at the intersection of precision and recall. The curves also revealed that ELMo uses higher detection thresholds than SBERT, which implies that the ELMo model is able to detect higher semantic similarity in paraphrased texts than the SBERT model.

Although the pr/rc curve used in this thesis is not the traditional one, the method was however used in Chong and Specia (2011) for plagiarism detection, and it does produces similar results to the traditional pr/rc method, but with clearer visualisation of the optimal performance point and its corresponding threshold at the intersection of pr/rc. For example, fig 5.2 is a traditional pr/rc curve for the proposed, the optimal point is somewhere just above 0.8 for both precision and recall, which is the same as what is observed using standard method as in fig 5.1a.

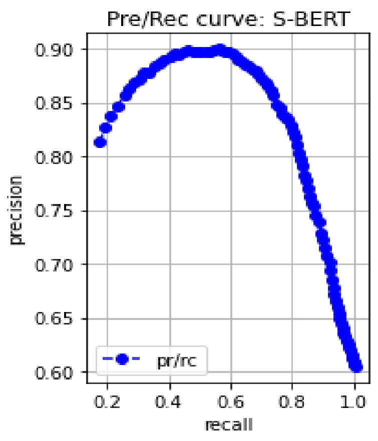


Figure 5.2: PC/RC curve for SBERT using traditional method

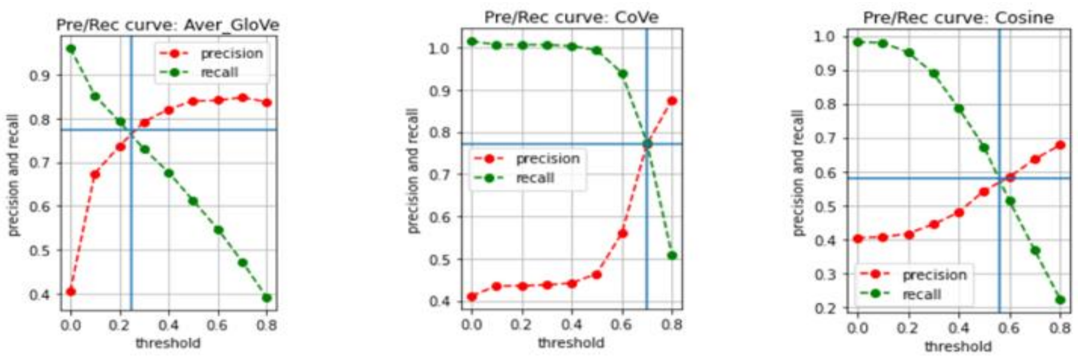


Figure 5.3: Precision/Recall curves plotted over every possible thresholds for the baseline models. Each curve shows the point where precision and recall intersect, and the threshold at the intersection point.

The pr/rc curves in fig 5.3 revealed similar performance for GloVe and CoVe (but lower than those of SBERT and ELMo). However, significant difference does exist between the intersection thresholds of GloVe and CoVe. The similarity in performance between CoVe and GloVe (and difference in thresholds) is likely due to the fact that CoVe utilises GloVe word embeddings to generate contextualised word representations, and that the contextualised word representations were not contextualised enough to bring about difference in their performances, but results in difference classification thresholds. The performance for the generic model (cosine with frequency vector) was once again the least as seen in the pr/rc curve in fig 5.3 (far right), and once again prove that the detection of paraphrased plagiarism require semantic models, and that string matching models are not effective enough.

When compared with previous results on three common paraphrase types, namely semantic, lexica and syntactic changes; on average our proposed model performed better in detecting syntactic (0.945) and lexical (0.915) changes than semantic (0.862), a similar trend was observed in Sanchez et al., (2019), although with a slightly different performance measure. This similarity in trend revealed the challenge in detecting plagiarised texts that have undergone semantic changes.

5.4 Results from the Experiments on the Crowdsourcing Corpus

This section presents and discusses results obtained from the evaluation of the models on the Crowdsourcing paraphrase plagiarism corpus.

Table 5.4: Results from Evaluation on the Crowdsourcing paraphrase Corpus

	Precision	*Recall*	*F1-score*	*AUC-ROC*
Proposed model (with SBERT)	**0.828**	0.951	**0.885**	**0.895**
Proposed model (with ELMo)	0.776	**0.960**	0.858	0.875
CoVe (McCann et al., 2017)	0.762	0.948	0.845	0.883
Average GloVe (global vectors)	0.764	0.92	0.835	0.875
Cosine with frequency vector (VSM)	0.743	0.957	0.836	0.849
N-gram feature (SOTA, Sánchez-Vega et al., 2019)	0.759	0.854	0.804	0.860

This table contains performance measurement in precision, recall, F1-score, and AUC-ROC for the proposed model (implemented with SBERT and ELMo) and baselines (including the SOTA model).

The results in table 5.3 show that the proposed paraphrase detector outperformed the other models as seen in their F1-scores and AUC-ROC values (SBERT: 0.885 and 0.895, ELMo:

0.858 and 0.875). This implies that on average the proposed model (when implemented with SBERT and ELMo) has better sensitivity and positive predictive value than the other models, in other words, the proposed model is more sensitive to paraphrased text, and has a higher probability of detecting paraphrased text in a corpus containing paraphrased and non-paraphrased text. In terms of evaluation using the AUROC measure, the results revealed that SBERT has a higher AUROC than the other models, this implies that SBERT is able to discriminate between the two classes (paraphrased and none-paraphrased) much better than the other models, and that at every possible threshold, the probability of positive (accurate) prediction is relatively higher for SBERT comparatively. This can be seen in the AUC-ROC curves in the next page in fig 5.3, the closer to the vertical a curve is, and the higher and parallel to the horizontal its surface is, the more area under the ROC curve is covered, and the higher the true positive rate. Fig 5.3 shows AUC-ROC curve for the top three best performing models on the Crowdsourcing paraphrase corpus, it is clear the ROC curves that the proposed model (implemented with SBERT) occupy a larger area under the curve than the other models, and there is very little difference in performance between the other two models (CoVe and ELMo).

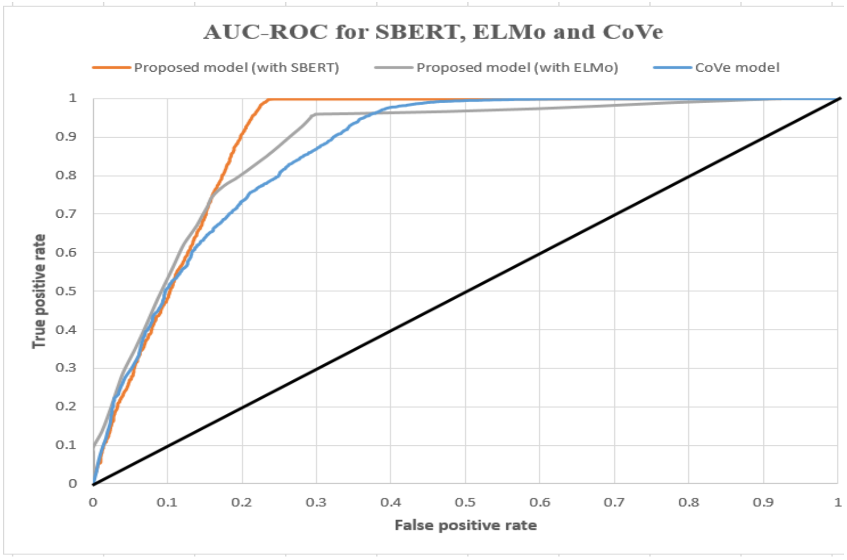


Figure 5.4: ROC curves for the top three best performing models showing area under the curve (AUC) for the proposed model (imple[5]mented with SBERT and ELMo) and the best performing baseline (CoVe).

The AUC values for the models align with their confusion matrix in fig 5.4-5.6 which clearly show lower misclassification rate for the SBERT model relative to the other models, the proportion of paraphrase plagiarism detected as non-paraphrase plagiarism and non-paraphrase

The further from the diagonal an ROC curve is, the higher the true positive rate. The diagonal line represents random guess.

plagiarism detected as paraphrase plagiarism (false positive rate) was much less for SBERT than the other models (SBERT: 0.128, ELMo: 0.164 and CoVe: 0.18).

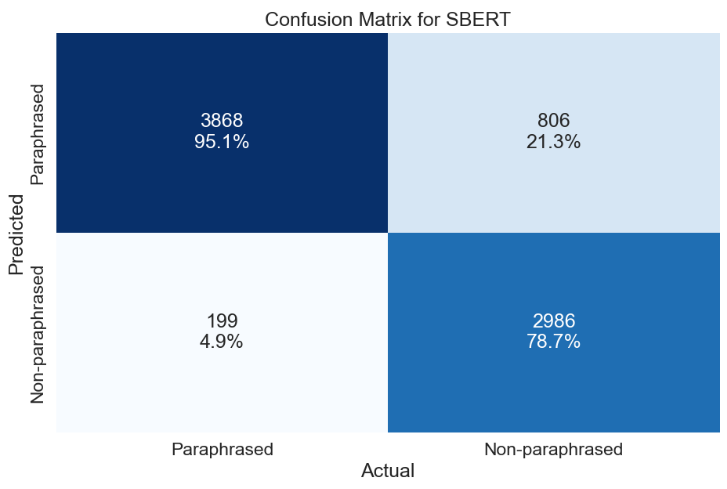


Figure 5.5: Confusion matrix for the proposed model (implemented with SBERT).

Error analysis of the SBERT model as seen in fig 5.4 revealed a relatively higher misclassification on the non-paraphrased class than on the paraphrased class.

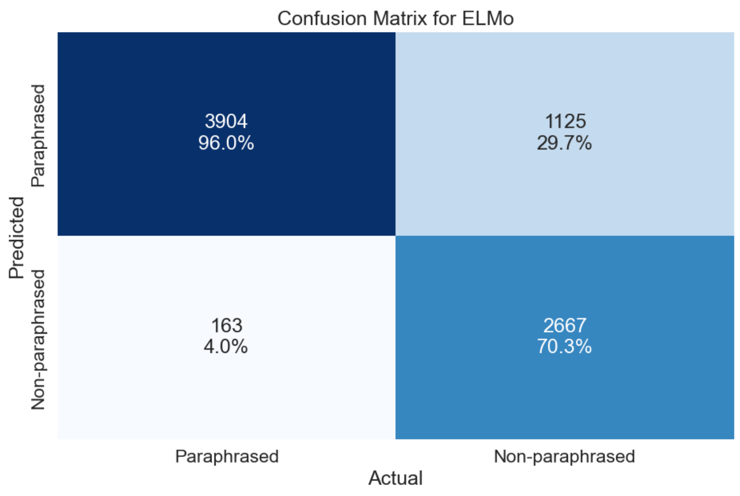


Figure 5.6: Confusion matrix for proposed model (implemented with ELMo)

The plot in fig 5.5 shows higher misclassification on the non-paraphrased class for the ELMo model relative to SBERT, but still better than the baselines, see appendix for the confusion matrices of the baselines.

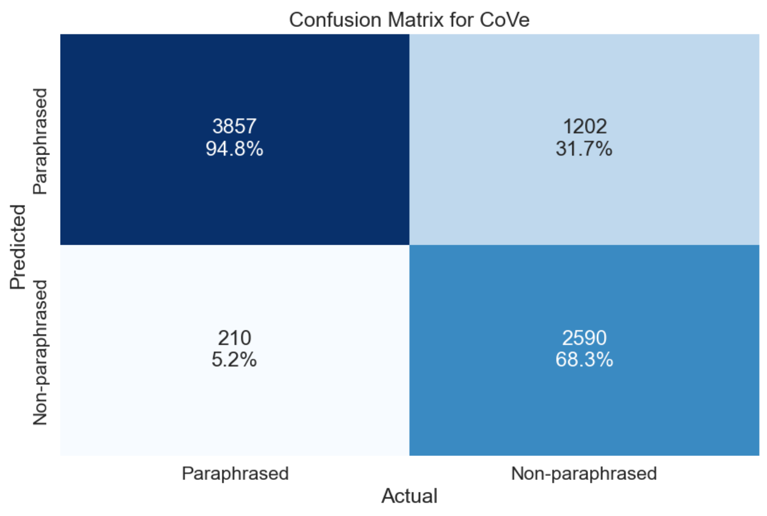


Figure 5.7: Confusion matrix for the CoVe model

Error analysis of the CoVe model as seen in fig 5.6 shows that the degree of misclassification on the non-paraphrased class is close to six times that of the paraphrased class.

The confusion matrix plots in fig 5.4-5.6 display different levels (degrees) of misclassification, however one thing they share in common is that majority of the misclassification originate from the non-paraphrased class, which is more or less a reflection of the nature of the corpus as this is observed across the models. A closer look at the confusion matrix plots revealed that the proposed model performed better on the paraphrased class when implemented with ELMo than when it was implemented with SBERT, the ELMo model detected more paraphrase examples (ELMo:3904/96%---SBERT:3868/95.1%) and less false negatives in comparison to the other models (ELMo:163/4%---SBERT:199/4.9%). However, SBERT clearly outperformed the other models on the non-paraphrased class with far fewer false positives (BERT:806/21.3%---ELMo:1125/29.4%). The results therefore suggests that the ELMo model is more sensitive to paraphrased texts due to its high true positives, while the SBERT model seems more precise because of its relatively low false positives. This finding is consistent with the results in table 5.3 which shows relatively higher recall for ELMo (sensitivity) and higher precision for SBERT (specificity).

Further analysis of the results in table 5.3 revealed that CoVe and GloVe, which are designed for semantic text similarity tasks fell short in performance in comparison to the proposed model. This is likely due to the fact that GloVe is a context independent model that does not capture the real meaning of words in contexts, a requirement that is essential for detecting semantic text similarity and paraphrase plagiarism ultimately. CoVe which is a contextualised learning model does not learn deep relationships between contextualised words as compared to the DCLM used in the proposed model; CoVe does not combine feature representations from the internal layer of the training model, but uses only representations from the top layer of an LSTM, and therefore does not take advantage of the deep relationships between contextualised words at the hidden units (Peters' et al., 2018), in addition, the DCLMs used in this work learns bidirectional contexts which is deeper relative to a unidirectional learning model such as CoVe that is unidirectional (Peters' et al., 2018; Ethayarajh et al., 2019). Although lower than the contextualised learning models, the performance of the SOTA and Cosine baseline were surprisingly high on the Crowdsourcing paraphrase corpus (relative to P4P corpus) given their simplicity and non-reliance on complex external resource. However the importance of precision which is a reflection of accuracy cannot be overemphasized given the sensitivity of plagiarism and the importance of reducing false detection. Hence every little improvement in precision, such as what was observed in the relatively superior performance of the DCLMs is worth it.

5.4.1 Results from BERT Fined-Tuned on Paraphrase Plagiarism Dataset

This subsection presents results obtained when the proposed model was implemented with a RoBERTa model fine-tuned on a corpus that contains paraphrase plagiarism and evaluated on the Crowdsourcing paraphrase corpus.

Table 5.5: Performance of Fine-Tuned BERT

	Precision	Recall	F1-score
Paraphrased	0.831	0.962	**0.891**
Non-paraphrased	0.951	0.79	0.863
Macro-average	0.891	0.876	0.877

The results in table 5.4 revealed that when the proposed model was implemented with a BERT model fine-tuned on paraphrase plagiarism corpus and evaluated on the Crowdsourcing paraphrase corpus, the performance obtained was slightly higher (F1-score=0.891) than when implemented with a BERT model fine-tune on generic corpora (SBERT F1-score=0.885). The performance of the model fine-tuned on paraphrase corpus is more balanced; margin between precision and recall is relatively smaller for the fine-tuned model. The superior performance is likely due to the fact that fine-tuning a pre-trained BERT adjusts the parameters (weights) of the model to be more sensitive to the types of paraphrases embedded in plagiarised texts. Much better performance could have been obtained if the dataset used for fine-tuning was much larger. See error analysis of the fine-tuned model in the confusion matrix below.

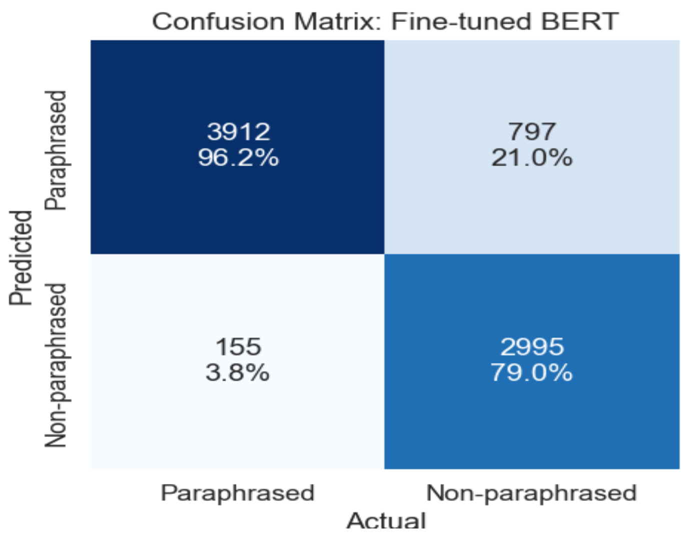


Figure 5.8: Confusion matrix for BERT fine-tuned on paraphrase plagiarism dataset

Error analysis of the confusion matrix in fig 5.7 revealed lower false positives and negatives for the fine-tuned BERT model relative to the other models, the proportion of the non-paraphrase plagiarism detected is much less than the paraphrase class, but still better than the other models.

5.4.2 Comparison with Results from a Previous Study

In comparison with results from a previous study where performance was computed as macro F1-score (taking into account performance for non-paraphrased class), Table 5.9 revealed that the proposed model outperformed Bars' et al., (2012) model as seen from their macro-average F1-scores (SBERT--0.871, previous study--0.85) respectively. The likely reason for the superior performance of the proposed model is that the Bars' et al., model is a combination of content, stylistic and knowledge based similarity measurement methods, content based methods measures similarity based on string matching, stylistic method measures similarity based on similar writing style, and knowledge based methods rely on semantic networks e.g. WordNet. None of these methods is able to decipher the real meaning of words in context, or learn deep semantic, syntactic and miscellaneous features associated with paraphrased text. These features are what DCLMs are trained to learn from text, and the likely reason for the superior performance of the proposed model over the other models, including the baselines and SOTA.

Table 5.6: Results for proposed model and a previous study on the Crowdsourcing corpus

		Precision	Recall	F1-score
Proposed model (with SBERT)	Paraphrased	0.828	0.951	0.885
	Non-paraphrased	0.938	0.787	0.856
	Macro-average	0.883	0.869	**0.871**
Proposed model (with fine-tuned BERT)	Macro-average	0.891	0.876	**0.877**
Previous study (Bars' et al., 2012)	Paraphrased			
	Non-paraphrased			
	Macro-Average			0.85

This table contains the macro-average F1-scores for our proposed model and results from a previous study where Crowdsourcing paraphrase corpus was used for evaluation.

Similar to the results obtained on the P4P corpus, the proposed model outperformed the baselines and SOTA model, and results from previous evaluation on the same dataset. The likely reason for the significant difference in performance is similar to those given for the P4P corpus, which is the ability of the proposed model to detect semantically related text sequences using a DCLM, which is an essential requirement for detecting paraphrased plagiarism.

Implications of Results With Respect to the Objectives of the Experiment

In relation to the research objectives, the proposed model outperformed the baselines and a SOTA paraphrase detection model on two corpora the contain paraphrase plagiarism, which suggest that the application of DCLM in plagiarism detection enhances the detection of paraphrase plagiarism, and addresses the objective that investigates whether the use of DCLMs in plagiarism detection would result in performance comparable to what could be obtained by a the state-of-the-art model. With respect to determining the most suitable DCLM for paraphrase plagiarism detection, the results revealed that SBERT (a transformer model) outperformed ELMo (an LSTM model) for the most part on both datasets, and therefore suggest that a DCLM with a transformer architecture such as SBERT is better suited for the task of paraphrase plagiarism detection. With respect to the objective on the types of paraphrases that could be detected with the help of CLMs, and those that are challenging, the results show that the proposed model could detect most of the paraphrase types, but a few paraphrase types were found challenging to detect; a trend seen in the performance of the other models, and suggest that representations from DCLMs are yet to capture features of certain paraphrase types in texts, and that more emphasis should be placed on the difficult to detect paraphrased types when building DCLMs. The cosine (with frequency vector) baseline which is the generic framework into which DCLMs were integrated performed relatively low in all the experiments, this imply that the proposed model, which is an extension of the generic model via integration with DCLMs brought about significant increase in the detection of paraphrase plagiarism given its superior performance over the generic and SOTA models. Although misclassification did occur across the models, which for the most part are likely due to the variety and complexity of the paraphrase types in the corpora, especially the P4P corpus. In depth analysis of the individual paraphrase types in table 5.2 revealed that at optimal performance, the DCLM models were able to detect certain paraphrase types with almost 100% recall (at reasonable precision), while the recall on other paraphrase types were as low as 60% or less. Further in depth analysis of the results revealed that the models performed well on paraphrase plagiarism with structural alterations (syntax) and not so well on discourse and semantic based changes.

Additional experiments to evaluate the performance of proposed model using a DCLM fine-tuned on a paraphrase plagiarism dataset revealed performances that were better than DCLMs fine-tuned on a generic dataset. Although the performance difference was small, it would have been much higher if the model had been fine-tuned with much larger dataset. The results is quite promising and leaves room for further experiments to evaluate the performance of DCLMs fine-tuned on large plagiarism corpora.

In terms of how the proposed method will be used in an end to end plagiarism detection system that comprises of candidate document retrieval, pairwise document comparison (intensive similarity search) and detection of potential plagiarised text fragments, and post-processing to identify plagiarised passages and filter off false passages, the proposed method aligns with the middle stage, which involves comparison and detection of plagiarised text fragments. The middle stage is where evidence of plagiarism is detected; it is about the most important stage.

5.5 Summary

Several experiments were carried out to evaluate the performance of DCLM in the task of paraphrase plagiarism detection. A proposed model described in the methodology that utilizes DCLMs for paraphrase plagiarism detection was implemented with DCLMs of two different architectures, and its performance evaluated. Experimental results revealed that the proposed model clearly outperformed a state-of-the-art model, and a number of established baselines. The findings suggests that DCLMs could be used to enhance the detection of paraphrased plagiarism. Further experiments to determine the performance of a DCLM fine-tuned on a paraphrase plagiarism corpus revealed performances that were better than those obtained from a DCLM fine-tuned on a generic dataset (SBERT). The performance difference could have been much better if the model had been fine-tuned on a larger plagiarism dataset, which is indeed promising and leaves room for future in that respect. Additional experiments to determine the performance of DCLMs with respect to different paraphrase types revealed that DCLMs could be used to detect most paraphrase types embedded in plagiarised texts, although a few were challenging or difficult to detect, even to the other models evaluated. The results suggest that DCLMs can be effectively used to detect paraphrase plagiarism, and are likely to be more effective when fine-tuned on paraphrase plagiarism corpora.

6 Experiments on Cross-lingual Plagiarism Detection

6.1 Introduction

The previous chapter described experiments carried out to address the challenge of paraphrase plagiarism detection. This chapter addresses research question three (3), which is stated as follows:

- Can a multilingual translation model that is independent of internet translation services be built using a Word2Vec (word embedding) model and applied to effectively detect cross-lingual plagiarism (CLP) with performances comparable to a state-of-the-art CLPD model?

Objective(s)

- To determine whether a multilingual translation model can be built by leveraging the predictive power of word embedding (particularly the Word2Vec model) and applied in CLPD to achieve performances comparable to a state-of-the-art CLPD model (based on the T +MA model).
- To determine whether a Word2Vec model could be trained to reproduce the output of an online machine translator and used in CLPD to produce similar performance to a commonly used CLPD model (T+ MA) which is dependent (and limited) by its reliance on internet translation services.

To address the above research question, this chapter describes experiments carried out to evaluate the performance of a CLPD model proposed in the methodology in section 3.6.1.2 for that does not rely on internet translation tools. The proposed CLPD model is an integration of relevant tools used in detecting plagiarism with a multilingual translation model (MTM) proposed in the methodology in section 3.6.1.1 for translating texts across multiple languages. The MTM depends on predictions from a Word2Vec model trained with simulated embeddings that comprise of semantically related words in different languages as context. The evaluation includes comparison against established methods, which includes state-of-the-art (SOTA) methods, baselines and results from previous studies. This chapter also includes experiments to test the effectiveness of the proposed CLP detector on low resource languages.

6.2 Experiments

The following sections of this chapter describe experiments carried out to train a Word2Vec model for multilingual translation, and to evaluate the performance of our proposed multilingual translation model (MTM) and a CLPD model that uses the MTM for language translation. The experiments are:

- Experiments to determine whether a Word2Vec model could be trained with multilingual embeddings to predict semantically similar words in different languages and used as a multilingual translation model (MTM).
- Experiment to evaluate the performance of a proposed CLP detector that uses the MTM for translating text across languages when detecting CLP.
- Experiment to determine the performance of the proposed CLP detector on low resource languages, and its effective as a language independent tool.

These experiments are described in the subsequent sections.

6.2.1 Experiments to determine whether a Word2Vec model could be trained and used for multilingual translation (MTM).

This section describes experiments carried out to train and optimize multilingual word embeddings using the Word2Vec model built with simulated embedding space.

Aims/Objective(s): To determine whether a Word2Vec model could be trained with a multilingual embedding space to predict semantically similar words in different languages, and be used as an MTM.

Description of experiments

The MTM uses multilingual word embeddings from a Word2Vec model trained with simulated embeddings space. We used a Word2Vec model from the Gensim NLP library and created a multilingual embeddings space that maps semantically similar words in different languages (e.g. English, Spanish, German and French) into the same context/space. For example, a context for an English word ‘car’ is as follows:

{‘car in English’, ‘car in French’, ‘car in German’, ‘car in Spanish’}

We created multilingual contexts for the *top-100,000* thousand most common English words with the help of Google translate (see section 3.5.1.1 in the methodology for more details) and trained the Word2Vec model with an objective of assigning similar vector representations to words that appear in similar context so that words that appear in similar contexts (semantically similar words in different languages) are predicted with relatively high probability.

To achieve the training objective, we experimented with two optimization parameters namely minimum context frequency and appropriate vocabulary size of the embedding space, the experiments to determine these parameters are:

a) Experiment to determine the frequency of co-occurrences of semantically related words in different languages to use as context for training the Word2Vec model. The contexts need to be replicated to increase the probability of predicting the semantic equivalent of a word in other languages.
b) Experiment to determine an appropriate vocabulary size of the embedding space of the Word2Vec model. The vocabulary should include all common English words and beyond. However, since the efficiency of a Word2Vec model is related to the size of the embedding space, rarely used words should be minimised in the vocabulary, this is determined experimentally.

The above experiments are described in the subsequent sections as experiments 1A and 1B.

6.2.1.1 Experiment 1A: Determination of the Frequency of Context Words in the Embedding Space of the Word2Vec Model for Optimal Language Translation.

Aims/Objectives: the aim of this experiment is to determine the number of times to replicate the embedding space of the Word2Vec model in order to optimise the prediction of target words, given a source word in a different language.

Description of the Experiment

The task here is similar to candidate retrieval as it involves retrieving relevant information for query terms; the query is a word in a language, and the expected information is the semantic equivalent of the query word in a different language. The embedding model is replicated *n*-number of times and trained so that each context appears n-times (where *n* is an integer; a multiplier), replication increases the probability of retrieving the context of a word, which are the translation of the word in other languages. Preliminary experiments revealed that the translations of a word in other languages is usually among the top-10 words returned by the model after replication; this threshold is set to improve the precision of the model.

The Word2Vec model was trained to perform the above task with different values of *n*, and its performance in terms of predicting target words was measured each time in recall, *n* is the number of times the embedding space is replicated. The values of *n* tested in the experiments ranges from 10 to 100, at an interval of 10; where *n=number of replications*= {10, 20, 30, 40, 50, 60, 70,…100}

The *top-1000* most frequently used English words (excluding the first 100) were used as source words, and their translations in Spanish were the target words; for each English word presented to the model, the models' output (top-10 words) is expected to contain the translation of the word in Spanish, and recall increases when the translation of the word is present, but decreases when not present.

Table 6.1: Results from the experiments to determine the best replication parameter

No. of reps (n)	10	20	30	40	50	60	70	80
Performance (recall)	0.265	0.383	0.534	0.728	0.925	0.927	0.927	0.927

Changes in recall as number of replications of the embedding space (contexts) of the Word2Vec model increases when optimising the MTM.

The results in table 6.1 show that optimal performance was achieved when the embedding space was replicated 50-times, and further increase did not result in any significant improvement in performance. The result also revealed a linear relationship between increase in the models performance with increase in the value of *n* (number of replications), and flattens afterwards as the value of *n* approaches 60 replications; which means as the value of n-increases, the performance of the model in terms of returning the exact translation of an English word in Spanish also increases, but as *n* approaches 60, the model becomes noticeably less linear, and the performance flattens out. See fig 6.1 for visualisation of the pattern in the results. The maximum performance was observed when the replications were between 50 and 60, and additional replications could not bring about significant increase in performance, the likely reason for this is that at 50 replications about 93% of the target words were already retrieved, and the amount of target words left was not large enough to bring about any significant increase in performance beyond 60 replications.

In terms of choosing the ideal number of replications to optimise the models' performance, the ratio of the difference in performance (recall) between 40 and 50, and between 50 and 60 is about 3:1 which makes 50 an ideal cut off (n-value) to use because the last significant change in performance was observed when the number of replications increased from 40 to 50. This value may vary depending on the vocabulary size of the embedding model; here the embedding model comprises of the top-25000 English words and their translations in German, French, Spanish, Dutch and Chinese.

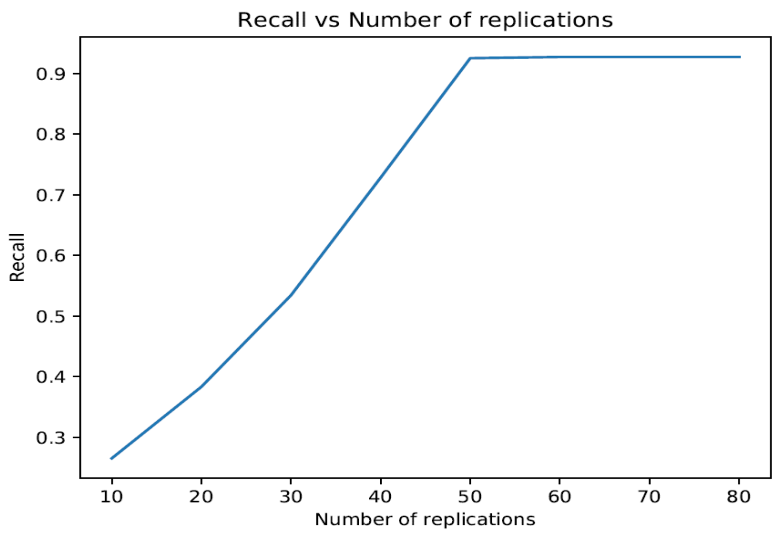


Figure 6.1: Changes in recall as number of replications increases

The curve in figure 6.1 shows changes in recall as the number of replications (*n*) of the embedding space increases, and flattens out afterwards as *n* approaches 50 replications (see table 6.1 for details).

6.2.1.2 Experiment 1B: Determination of an Appropriate Vocabulary Size for the Embedding Model for Optimal Language Translation.

Aims/objectives: to determine an appropriate vocabulary size for optimising the performance of the Language Translation Model.

Dataset: the dataset used in this experiment is the Pan2012 evaluation corpus on plagiarism detection. The corpus contains CLP cases, and therefore suitable for this experiment.

Description of Experiment

This experiment is carried out to determine the appropriate words and vocabulary size to simulate an embedding space for training a Word2Vec skip-gram model to optimise the prediction of translated words. The vocabulary comprises of English words as pivots and their translations in order languages as contexts. The vocabulary sizes experimented with ranges from the top-5000 to the top-45000 most common English words, at an interval of 10,000 words. Contexts were created for each word by retrieving the translations of the word (in Danish, French, German, Spanish and Chinese) using Google translate, and each context was replicated. Translation models of different vocabulary sizes were tested and their performance measured in terms of the effectiveness and efficiency (runtime) of a model to be used for real time language translation in CLPD.

Vocab sizes= {5000, 15000, 25000…,.., 45000)

6.3 Results and analysis

The results obtained from the experiments on the determination of an appropriate vocabulary size for optimising the performance of the translation model are presented in table 5.2.

Table 6.2: Results from the experiments to determine appropriate vocabulary size.

Vocabulary size (words)	Precision	Recall	Granularity	Plagdet score	Time (sec)
5000	0.875	0.802	1.020	0.819	1934.41
15000	0.932	0.835	1.003	0.878	2844.79
25000	0.930	0.846	1.003	0.884	3068.94
35000	0.926	0.850	1.003	0.884	3363.19

The results in this table are the performance of the MTM in precision, recall, granularity, plagdet score and runtime when implemented with different vocabulary sizes.

The main aspect of the results in table 6.2 is the performance difference observed as the vocabulary size increases, which points to the effect increasing vocabulary size has on performance, and reveals an appropriate vocabulary size to optimise the performance of the MTM. The results show that the highest increase in performance was observed when the vocabulary size increases from 5000 to 15000 (0.819→0.884) and flattens out afterwards. The results suggest that most of the translated CLP cases in the corpus could be detected with an MTM built with a vocabulary size of the top-25000 most common English words; about 418 out of the 500 cases in the Pan 2012 evaluation corpus.

Moving upwards to a vocabulary size of the top-25,000 most common English words, the models' performance increased, but slightly, and the run time was still reasonable enough for plagiarism detection tasks; about 6.2 seconds per suspect document. In real world situation that require much more document comparison to be made (i.e. in monolingual plagiarism detection), the detection time reported (6.2 seconds) can be considered average. However, typical CLPD require the use of external resources which tend to slow down the process, unlike in monolingual plagiarism detection. There are a number of ways to speed up the process and reduce detection time, one way is to use parallel processing that allows different stages of plagiarism detection to be carried out simultaneously. Another way is to use cache memory to store pre-processed source text to avoid going through pre-processing phase during detection time. One other approach is to limit searched query terms to only key words with high TFIDF scores (most relevant terms) thereby reducing the amount of translations to be carried out, this method could however result in the omission of potential suspect documents. As stated earlier, the reported detection time is considered reasonable, hence the top-25000 most common English words and their translations in other languages were found to be appropriate for optimising an MTM for CLPD.

In terms of pattern, the results in table 6.2 show that as the vocabulary size increases, the effectiveness (recall) of the model increases, and the run time increases as well. The recall increases because the models' ability to detect more translated words increases, while the increase in run time is likely due to increase in search time. The increase in vocabulary size was also met with a corresponding decrease in precision which could be attributed to coincidental matches that may have occurred in the corpus, and may also be due to inaccurate translations of some words. The decrease in precision was however much smaller in comparison to the increase in recall.

6.3.1 Experiment 2: Evaluating the Performance of the Proposed CLPD Model

This experiment is carried out to evaluate the proposed CLPD model described in section 3.6.1. The proposed language translation model (MTM) is combined with relevant plagiarism detection tools, and used to detect CLP in the Pan2011 and 2012 evaluation corpora.

6.3.1.1 Evaluation on the Pan 2011 corpus

The proposed CLPD model was used to detect CLP in the Pan2011 evaluation corpus, and its performance was measured. Below are the details of the implementation and evaluation.

Candidate documents were first retrieved from an inverted index build with the source document collection using PyLucene search engine library. This process was carried out by querying the inverted index table with key-words extracted from a suspect document and retrieving source documents that contain the key-words as candidates. Key-words were extracted from query documents as follows; a query (suspect) document is pre-processed by case folding (reduce text to lower alphabetic case) and tokenisation to single words, term frequency weighting was then used to retrieve all words that occur not more than 3-times as key-words; this number was determine empirically via heuristics by trying different values while computing performance, and the value that corresponded to the best performance was chosen. To retrieve cross lingual candidates, query words (from a suspect document) were expanded using the MTM to identify semantically similar words in other languages which are used to retrieve candidate documents from the inverted index (source collection), this was done by querying the inverted index table with the expanded key-words and retrieving the top-10 source documents that contain significant amount of the query words as candidates. When source documents have been retrieved for a query (suspect) document, detailed search for plagiarism 112 was then carried out between the suspect document and its candidate documents, and plagiarised passages (in the suspect document) and their respective sources (in candidate documents) were mapped out as described in the methodology in section 3.5.1.2.

The above process was used to detect CLP in suspect documents, and the performance of the model was measured in precision, recall, granularity and plagdet score. The evaluation also includes performance comparison with established methods; baseline models and results from previous studies.

6.3.1.2 Evaluation on the Pan2012 corpus

The Pan@clef 2012 corpus contains documents in pairs; each suspect document is assigned to a source document, comparison is therefore expected to be between pairs of source and suspect documents. Taking this into account, the evaluation on the Pan 2012 corpus was carried out twice as follows;

- The first evaluation follows the normal plagiarism detection framework that begins with searching for plagiarised sources in a large collection of source documents; we followed the steps used in the previous evaluation on the Pan2011 corpus.
- The second evaluation involves pairwise document comparison; we used the proposed method for detecting plagiarism in pairs of documents as described above between a suspect and a candidate documents.

6.3.2 Implementing the Baselines

The baselines used for evaluation are the Kasprzak and Brandejs (2010) T+MA model, and the Francisco et al., (2013) model. These models were implemented as follows:

The Kasprzak and Brandejs (2010) T+MA Model

The kasprzak and Brandejs model follows the normal implantation of a T+MA model, language identification and translation were carried out on source documents using a Python library known as Landetect (Danilak, 2016) and Google translate respectively, source documents were translated to the language of the suspect documents ready for monolingual plagiarism detection.

This model was implemented as follows; Langdetect (Danilak, 2016), a Python programming language identifier and Google translate were used to identify languages and translate source documents into the language of suspect documents before applying monolingual plagiarism analysis. Candidate selection was carried out by transforming the source documents in the collection into 5-gram fingerprinting models (using an MD5 hash function) and indexed, candidate documents for a suspect document was retrieved by transforming the suspect document into a 5-gram fingerprinting model in the same way as the source documents, and then used as query to retrieve all indexed source documents that contain up to 20-matches; gap between any two consecutive chunks was set to a maximum length of 50 characters. Matched fragments between a query document and a candidate document was used to retrieve plagiarised passages by retrieving every line of text (offsets) that contains a fragment, and merging neighbouring lines into a passage. See Kasprzak and Brandejs (2010) for the merging parameters used.

The Francisco Et Al., (2013) Model

The Francisco et al., (2013) baseline model is similar to the one proposed in this work in that it is multilingual, and also language independent, but differs in that it uses BabelNet; a multilingual semantic network to disambiguate texts written in different languages, while the approach proposed in this work uses the MTM for language disambiguation.

The Francisco et al., (2013) model was implemented as follows; for a pair of source and suspicious documents under investigation for plagiarism, synonym sets were generated for each word, followed by transformation to knowledge graphs that relate words to similar concepts in the documents using BabelNet. Each word is assigned a weight that is a reflection of the strength of relationship between the word and a concept. Detailed comparison was carried out by comparing the knowledge graphs of the pair using the similarity function described in Francisco et al., (2013). Pairs of documents with similarity scores above a threshold value were retrieved as plagiarised and co-occurring words that relate to similar concepts in their knowledge graphs were merged as seeds to form plagiarised passages. The threshold was determine on the fly as performance was computed, different values were tested from a search space of 0.1-1.0 (because the maximum similarity score is 1), and the value that corresponded with the best performance (F1-score) was used as threshold.

The implementation of these baselines were validated by making comparison with results in the original papers; the baseline implementations were evaluated on the corpora used in the original paper and the performances obtained were compared with performances in the original papers.

6.3.3 Results and Analysis

In this section, the results obtained from the evaluation of the proposed CLPD system are presented and analysed. Tables 6.3 and 6.4 contain the results obtained for the proposed model, baselines and previous studies on the Pan2011 and 2012 evaluation corpora.

Table 6.3: Results from the evaluation on Pan 2011 Corpus

CPLD methods	Precision	Recall	Granularity	Plagdet score
		Manual		
Proposed detector	0.767	0.594	1.0065	**0.667**
Baseline (Francisco et al., (2013)	0.75	0.575	1.002	0.65
Previous study	0.750	0.460	1.0000	0.57
Baseline (T+MA)	0.727	0.445	1.0002	0.552
		Automatic		
Proposed detector	0.954	0.943	1.0000	**0.945**
Previous study	0.960	0.920	1.0000	0.94
Baseline (Francisco et al., (2013)	0.937	0.91	1.0004	0.923
Baseline (T+MA)	0.945	0.878	1.0000	0.91

The results in this table are the performance obtained from the evaluation of the proposed CLPD model, the baseline and from a previous study (best performing system in Pan 2011 competition) on the Pan 2011 corpus.

The results in table 6.3 show that the proposed CLPD model outperformed (0.667, 0.945) the T+MA baseline (0.552, 0.91) on the manually and automatically generated plagiarism cases as seen in their plagdet scores. The performance was much higher on the automatic CLP cases (0.945) than on the manually created ones (0.667) because the manually created cases were simulated to appear like real world cases of CLP. The results therefore point to the difficulty involved in detecting real CLP cases as opposed to machine generated ones, which is consistent with previous studies and the baseline.

In comparison with results obtained from previous studies, the results show that the proposed CLPD model outperformed the best performed system in the Pan 2011 competition which is quite difficult to beat, this is a reflection of the effectiveness of the proposed model in terms of being able to detect translated texts that have undergone further alterations. In terms of precision, both systems were more or less even.

Table 6.4: Results of the proposed CLPD model, baselines and a previous study on Pan 2012 corpus**.**

CLPD methods	Precision	Recall	Granularity	Plagdet score
Plagiarism Detection In Document Pairs				
Proposed CLPD model	0.93	**0.846**	1.003	**0.884**
Baseline (Francisco et al., 2013)	0.87	0.85	1.003	0.862
Baseline (T+MA)	0.93	0.76	1.00	0.84
Previous study	0.82	0 .727	1.00	0.771
Detection Of Plagiarism Given A Source Collection				
Proposed CLPD model	0.91	**0.78**	1.00	**0.84**
Baseline (Francisco et al., 2013)	0.87	0.75	1.00	0.81
Baseline (T+MA)	0.91	0.73	1.00	0.81

The performance of the models in table 6.4 are for the pairwise CLPD task and the standard plagiarism detection task that begins with candidate selection. The previous study in the table is the best performing system in the Pan2012 competition.

The results in table 6.4 show that the proposed CLPD model outperformed (0.884) the baselines (0.863, 0.84) on the two evaluation tasks undertaken with the Pan 2012. The difference in performance against the T+MA baseline (our benchmark) is clearly seen in the recall (proposed detector: 0.846, 0.78; baseline: 0.76, 0.73). The results also show that the proposed model outperformed the second baseline as can be seen in their relative plagdet scores (proposed detector: 0.884, 0.84; baseline2: 0.863, 0.81). The performance difference was even higher in comparison with the results from the previous study (the best performed system in the Pan 2012 competition on plagiarism detection); which emerged as the least performing. With respect to the two detection tasks, the performance of the model was much better when used to detect plagiarism in pairs of documents, than when applied in normal plagiarism detection that begins with candidate selection; this pattern is also reflected in the results of the baselines and the previous study. The likely reason for the difference in performance between the two detection tasks is that in candidate selection, there is always a chance that some plagiarised sources may not be selected, unlike in the pairwise task that plagiarised sources are given in advance; without going through the probabilistic candidate selection process. While the results from the pairwise comparison seem impressive, plagiarism detection in large collection documents is usually preceded with candidate selection.

Analysis of Performance across Corpora

In terms of analysing the performance of the models across both corpora with respect to their functionality, the T+MA uses overlapping 5-gram sequences to detect plagiarised texts, while this could result in high precision, heavily altered plagiarism cases are unlikely to have significant amount of matching 5-word sequences, but when found, the likelihood of copy is almost certain (see Thompson et al., 2015 for detailed analysis of how n-gram sizes affect the performance of plagiarism detectors). In comparison, the proposed CLPD model detects plagiarism using sentence level similarity analysis, unlike the T+MA baseline, the sentence level similarity analysis allows the proposed model to detect both lightly and heavily altered plagiarised texts. In addition, the proposed CLPD model is able to detect semantically related words such as synonyms (translated texts that have been paraphrased), and not just words that have been translated directly into their exact meanings in other languages using automatic processes. This is reflected in the significant difference in recall, but not in the precision between both systems. The difference in precision between the proposed CLP detector and the baseline was not significant, which proves that the proposed detection model actually captures the translation precision of the internet translation tool that the T+MA model uses for cross language translation.

On the other hand, the Francisco et al., (2013) method is based on concepts matching, and many non-plagiarised documents in the corpora share similar concepts, in such situation, concepts based matching methods would most likely detect similar but non-plagiarised concepts as plagiarised; which would only increase false positives, and decrease precision. In contrast, the method proposed in this research learns word translations from Google translate which are more exact and precise, and this explains why much of the difference between the proposed model and the Francesco et al., baseline model lies in their precision. In addition, when weighting words based on their relationships to a concept in BabelNet, there are cases where words that are not very related to a particular concept are assigned higher weights than more related words which would most likely result in false similarity measurement and plagiarism detection.

6.3.4 Experiment 3: Determination of Performance on Low Resource Languages.

Aim/objective: to investigate whether the proposed CLPD model could be effectively used to detect CLP in low-resource languages, and whether it can be used for CLPD independently of language.

Dataset: the dataset used is the Pan 2012 evaluation corpus, and the translation plagiarism category is the particular category of interest, it comprises of 500 source and 500 suspect documents.

For the corpus to be suitable for the evaluation task, the translation obfuscation category was changed to a low resource language (Belarusian), and CLPD systems were then used to detect plagiarism across the languages. The intuition is that the proposed CLPD system is language independent, and changing the language of the source or suspect document would have little or no effect on the performance of the system. Regardless of language change, details of the plagiarised passages (the offsets and lengths) should remain the same, hence the ground truth included in the Pan2012 should be valid (and accurate) for measuring performance.

Details of the experiment

The experiment to detect plagiarism in low resource languages (Hungarian and Belarusian) was carried out in two stages; in the first stage a mul[6]ti-lingual translation model (MTM) was built to include Hungarian and Belarusian in the embedding space. In the second stage, the Pan 2012 corpus was altered by replacing the translation category (a high resource language) with the low resource languages, this was done using Google translate to change the source languages (in Pan2012 translation category) to Hungarian for all the 500 source documents (in

Performance on the low resource language should be similar to that of the high resource language (in the Pan2012 corpus) as the evaluations were carried out on semantically similar texts expressed in different languages the translation category), the same was done for the Belarusian language, and a third, which is a mixture of Belarusian and Hungarian in equal number (250 each). At the end, three versions of the Pan 2012 corpus were generated, which differ in terms of the language of the source documents in each version.

When the corpora have been prepared by the inclusion of low resource languages, they were then used to evaluate the proposed CLPD model in the task of cross-lingual plagiarism detection, and the performance of the model was measured in precision, recall granularity and pladget score.

6.3.5 Results and Analysis

The results obtained from the experiments on CLPD on the low resource languages are presented and discussed below:

Table 6.5: Results From the Evaluation on Low Resource Languages

	Precision	**Recall**	**Granularity**	**Plagdet score**
Performance of proposed CLPD model on modified versions of Pan2012 (low resource)				
Hungarian	0.921	0.832	1.0020	0.874
Belarusian	0.915	0.819	1.0010	0.864
Hungarian + Belarusian	0.911	0.809	1.0003	0.857
Performance of proposed CLPD model and Baseline on Pan 2012 corpus (original version: high resource)				
Proposed detector	**0.93**	0.846	1.003	**0.884**
Baseline (Francisco et al., 2013)	0.9	0.85	1.003	0.874

This table contains results obtained from the experiments to investigate whether the proposed CLPD model is as effective on low resource languages as it is on high resource ones. The table shows the relative performance of the proposed model on low and high resource languages.

The results in table 6.5 show that the performance obtained from the evaluation on the low resource languages (Hungarian and the Belarusian) for the proposed model were close (0.874, 0.864) to what was obtained on the high resource language in the original version of the Pan 2012 corpus. The drop in performance can be attributed to slight variation in translations; when translating from the original sources to Hungarian and Belarusian. This is likely the case because certain words have more than one meaning across languages, and machine translators are by no means perfect. In addition, detailed examination of the translated texts revealed that the machine translator was unable to translate certain words to Hungarian and Belarusian. Performance comparison across the two low resource languages shows that the performance was better when the source was Belarusian, than when Hungarian was used as source. The difference in performance was quite small, and can be attributed to either limitation in the vocabulary of the machine translator, or Belarusian may just be richer in vocabulary than Hungarian. Detail analysis revealed that not all words in the high resource language were translated to the low resource languages, this is likely due to the absence of such words in the low resource language, or limitation in the vocabulary size of the machine translator.

In comparison with the Francisco et al., (2013) baseline, the results show that the performance obtained on the low resource language for the proposed CLPD model were almost even with the baseline method, even when the baseline method was used to detect plagiarism on the low resource languages. The results therefore prove that the CLPD model proposed in this research is not only effective on high resource languages, it is also effective on low resource languages.

6.4 Implications of Results

The results imply that when provided with an appropriate embedding space, which include words and their translations in other languages as contexts, a word embedding model such as the Word2Vec model could be trained to predict semantically related words in different languages, and the pre-trained model could be used for multilingual translation. With respect to cross-lingual plagiarism detection, an MTM based the trained Word2Vec model could be applied in CLPD to translate texts across languages with performances comparable to a state-of-the-art (SOTA) T + MA model, but without being limited by the internet as a traditional T+MA model. The results also imply that, when texts are translated automatically and then paraphrased, they become more difficult to detect than automatically created CLP cases, which is consistent with previous studies. However, CLPD systems that are designed to detect semantically similar words (such as the synonyms of translated words) such as the one proposed in this research should be able to detect most difficult cases of CLP. Although the performance of the proposed model was only marginally better than the baselines as seen in tables 5.4 and 5.5, it is worth pointing out that these baselines are the very best and difficult to beat.

As with most systems, the effectiveness of our model is limited, it is worth recollecting that we built our system on predominantly European languages that have similar syntax, it was not evaluated on non-European languages such as Arabic which has completely different syntax. The proposed model is also most likely to fail when used for CLPD (or language translations) on languages that do not form part of the vocabulary of the underlying word embeddings.

6.5 Summary

We have conducted the following experiments to evaluate the performance of a multilingual translation model that utilises the Word2Vec model for language translation, and a CLPD model that utilises the MTM.

- Experiments to evaluate the performance of a multilingual translation model that incorporates a Word2Vec model trained with simulated embedding space, and this breaks down into two sub experiments.
    1. Experiment to determine an appropriate embedding space to optimise the performance of a Word2Vec model for multilingual translation.
    2. Experiment to determine an appropriate vocabulary size for the embedding space of the Word2Vec model for multilingual translation.
- Experiment to evaluate the performance of a proposed CLP detector that uses the MTM for translating text across languages when detecting CLP.
- Experiment to determine the performance of the proposed CLP detector on low resource languages, and how effective it is as a language independent tool.

The main objective of our experiments is to determine whether a Word2Vec model could be trained to predict semantically similar texts in different languages, and used in CLPD for translating texts across languages so as to eliminate the need to connect to internet translation tools that limits common CLPD methods (i.e. the T+MA model). Experimental results revealed that the proposed model performed comparably to a SOTA T + MA model, and even better in some cases, this is because the proposed model is able to predict a range of semantically similar words that could be used to paraphrase translated texts, and not limited to predicting the exact translation of words. The results prove that it is possible to train a Word2Vec model to learn to predict semantically similar words in different languages, and applied in CLPD with performances comparable to established CLPD models, and hence satisfy the objective of the research question. Additional task carried out to test the effectiveness of the proposed CLPD model on low resource languages revealed performances that clearly indicates that the model can be effectively used to detect CLP on both low and high resource languages.

7 Discussion and Conclusion

7.1 Thesis summary

The previous chapters described experiments carried out in this research to address some of the challenges facing plagiarism detection, particularly in the areas of paraphrase and cross-lingual plagiarism. This chapter discusses the findings from the experiments carried out in relation to the research questions and aims/objectives, implications, and the literature, this chapter also covers the limitations of this research and potential areas of future work.

Here are the questions addressed in this research;

- What are the best performing combination of surface similarity measures and textual features (as measured by precision, recall and F1-score) from those described in the literature for detecting similar and near similar texts?
- Can deep contextual learning models be used to enhance the detection of paraphrase plagiarism with performances comparable to, or even better than a state-of-the-art (SOTA) model?
- Can a multilingual translation model that is independent of internet translation services be built using a Word2Vec (word embedding) model and applied to effectively detect cross- lingual plagiarism (CLP) with performances comparable to a state-of-the-art CLPD model?

The above research questions resulted in a number of novel contributions which are described below.

Regarding the problem of paraphrase plagiarism detection, which is about determining whether the application of DCLMs in paraphrase plagiarism detection could result in performances comparable to a state-of-the-the-art model, a novel paraphrase plagiarism detector that extends a generic plagiarism detection framework with an inbuilt deep contextualised learning model (DCLM) was proposed. The novel deep paraphrase plagiarism detector (DPPD) uses a DCLM for detecting semantically related text sequences that may have been altered lexically, syntactically, semantically or by other linguistic techniques. The proposed DPPD is capable of detecting different expressions (or types) of paraphrases in plagiarised text, and outperformed a number of standard baselines and a state-of-the-art paraphrase plagiarism detector. The proposed DPPD model performed even better when implemented with a DCLM fine-tuned on paraphrase plagiarism corpus. The findings of this research prove that DCLMs can effectively be used to detect paraphrase plagiarism with performances comparable to a state-of-the-art model.

Regarding the problem on cross-lingual plagiarism detection (CLPD), which investigates whether a Word2Vec model could be trained to capture and reproduce the translation outputs of an online machine translator, and applied in CLPD with performance comparable to a standard baseline that uses online machine translator (a T + MA model), a novel CLPD model and a multilingual translation model (MTM) were proposed. The proposed CLP detector uses an inbuilt language translation model to detect plagiarism across multiple languages without connecting to internet translation tools. The inbuilt language translator is a novel multilingual translation model (MTM) built by pre-training a Word2Vec model with simulated embeddings to predict the semantic equivalent of a word in other languages (language translation). The proposed MTM is designed to be extended (scaled) to multiple languages, and to tackle the problem of CLPD in low resource languages. The proposed CLP detector performs comparably and in many cases better than a standard T+MA baseline and a SOTA CLPD model. The findings therefore prove that the capabilities of a Word2Vec model could be applied in CLPD to reduce the overly dependent on internet translation services of common CLPD systems.

Regarding the question that investigates whether specific combination of surface tools/techniques used for measuring intertextual similarity could be determined for detecting plagiarised texts that have been obfuscated to different levels of complexity, a novel hybrid surface plagiarism detector (HSPD) was proposed, the HSPD is capable of detecting plagiarised texts of varying degrees of obfuscation ranging from obfuscated texts that contain large fragments of verbatim (cut and paste) text reuse to heavily obfuscated text with scanty overlaps. The novel HSPD is a combination of the most suitable surface similarity measures and textual features which includes suitable term weighting method, ngram document model and surface similarity measure. On majority of the complexity levels, the HSPD performed better than standard baselines and results from previous studies, which proves that suitable surface similarity measurement tools and techniques could be combined to detect plagiarised texts with varying degrees of obfuscation, and confirmed the hypothesis that plagiarised text often contain unaltered remains of their sources which are surface features that could be detected (and used in plagiarism detection) by the right combination surface detection tools/techniques, even when such features are in trace amount.

7.2 Discussion of Findings In Relation To Existing Research

This section describes how the findings of this work relate to the existing body of knowledge (literature) in plagiarism detection.

In terms of the problem of paraphrase plagiarism detection, our novel model (DPPD) significantly outperformed a SOTA paraphrase plagiarism detection model proposed by Sánchez-Vega et al., (2019) on two standard paraphrase plagiarism datasets (see tables 4.1 and 4.3) . The superior performance over the SOTA model is as a result of the integration of DCLM into an existing plagiarism detection framework to detect semantically related texts that have undergone obfuscation using different paraphrase techniques. DCLMs are trained to learn deep semantic, syntactic and miscellaneous relationships between texts, this enhances the DPPDs' ability to detect pairs of texts that are semantically related but lexically different (paraphrased texts). The SOTA model on the other hand uses character trigrams to capture content and stylistic similarity in paraphrased texts, lexical matching methods such as character trigrams do not have the ability to detect semantic relationships between texts that are lexically different, and this explains the superiority of the proposed DPPD over the SOTA model. This finding is consistent with the literature that suggests that detecting paraphrase or high obfuscated plagiarism require models that are designed to detect semantic text similarity (Gupta, 2016; Foltýnek et al., 2019), or the integration of such models with existing lexical matching models. In addition, the proposed DPPD also outperformed a number of baseline models which include models designed to detect semantic similarity (including GloVe and CoVe), the average GloVe baseline is a context independent model, which means it is unable to decipher the real meaning of words in context, the model is therefore prone to errors when dealing with texts that contain lexically similar words but are semantically different, such as polysemy. The CoVe baseline is a shallow context learning model used as baseline in many studies, it uses only the outermost layer of an LSTM to represent texts, and does not take advantage of the rich contextualised features in the hidden layers. DCLMs are not limited by the short comings of these semantic models, which explains the superior performance of the proposed DPPD.

One of the most common (and effective) method used in the literature to detect CLP is the T + MA model (Barrón-Cedeño et al., 2013; Tian et al., 2017; Brychcín, 2020), which is heavily reliant on internet translation services (CLPD models that uses online machine translators to normalised text to a common language before applying a monolingual plagiarism detection analysis fall under the T + MA method). The proposed cross-lingual plagiarism detector (CPD) addressed this reliance on the internet by using a built in multilingual translation model (MTM) proposed in this research for language translation instead of internet translation tools. Evaluation was carried out to determine how well the CPD performs in comparison to a traditional T + MA and a model based on concept net (Francisco et al., 2013). Evaluation results revealed that the CPD outperformed a T + MA model proposed Kasprzak and Brandejs (2010) that emerged as the best performing cross-lingual plagiarism detection model in the Pan2011 competition. The CPD performed comparably to, and in many cases better than the T + MA model, this is likely due to the ability of the CPD not only to translate texts to their semantic equivalent in other languages, but also to detect translated texts that have been paraphrased, which is one of the most challenging aspect of translation obfuscation (Sánchez-Vega et al., 2016; Foltynek et al., 2020; Gupta and Kaur, 2021). The CPD also outperformed the Francisco et al., model which is based on concept mapping. The proposed CPD learns to predict verified language translations from Google neural machine translation (NMT) service which is often used by plagiarist as it is freely available for short text language translation. Concept mapping on the other hand is prone to errors; documents may have similar concepts without being plagiarised. In addition, architectural difference such as making comparison at sentence level which targets actual plagiarised fragments (at local level) may have contributed to the superior performance of the CPD over the other models. The proposed CPD is therefore comparable to state-of-the-art CLPD models in performance; it is indeed a T + MA model that is not limited by the internet as a traditional T + MA model. In addition, the proposed CPD is designed to be used for CLPD on low resource languages, which is currently a challenge given the lack of sufficient training data for such languages (Brychcín, 2020). Recent studies shows that methods based on neural machine translation (NMT) have been used for cross-lingual semantic text similarity, but are not well developed to keep up with the speed required for plagiarism detection, however they can be used for short text CLPD, but studies shows that they are prone to errors (Cer et al., 2017).

Research limitations

One limitation of this work, as well as many other researches on plagiarism detection is the lack of real world plagiarism datasets. The unavailability of datasets that contain real world plagiarism cases for evaluation purposes makes it difficult to know how well a proposed plagiarism detection system would perform in a production environment, such as in an academic institution, and therefore limits the options to fine-tune a system for general purpose use.

Another limitation of this research is the inability to validate some of the models proposed in this thesis against a standard and commonly used commercial plagiarism detection system (such as Turnitin) due to patent protection, and there is also a lack of unified (standard) evaluation framework which includes standard datasets for training and testing, search engines, evaluation metrics etc. Efforts would however be made in the future to carry out such validation under a legal binding agreement.

7.3 Future Work

In continuation of the progress made in this research, here are potential areas of future work:

Future work would include fine-tuning a state-of-the-art pre-trained DCLM to detect virtually all types of plagiarism, and not just limited to paraphrase plagiarism. This will include acquiring sufficient amount of datasets (corpora) that contain real cases of plagiarism with different features common to plagiarised texts.

Future work would be carried out in the application of DCLMs in other areas of academic misconduct closely related to plagiarism detection; such as in the detection of falsifications, fabrications and ghost writing. This would include training DCLMs to detect these misconducts in academic research, and would involve training models to learn contextualised features associated with these academic misconducts. A major concern however would be acquiring sufficient training data in different academic domains to train DCLMs.

With respect CLPD, future work will be focused on extending the multilingual translation model proposed in this research to include more languages (including those with low resources) so as to increase the coverage and capability of the proposed CLP detector to other languages. The MTM was pre-trained with four European languages, hence the MTM in its current form can only translate texts between the four languages, which limits CLPD to only those languages. However, the MTM is designed to be extended to include more languages. Effort will also be made in future to build an efficient multilingual representation model for transfer learning, such as multilingual BERT that can effectively translate texts across languages with similar structure, but no so well across languages of different word order (Pires et al., 2019).

Future work will also include combining surface similarity measurement tools with semantic similarity measures (context based) to enhance the detection of plagiarism across all obfuscation levels. Surface similarity measurement tools are relatively more efficient and can keep up with high demand in plagiarism detection than semantic similarity methods however they are not effective in detecting high obfuscation plagiarised text. Semantic similarity measures in their current form are not well developed to meet the demand of real time plagiarism detection but are relatively more effective on high obfuscation plagiarism, hence a stepwise approach that involves carrying out initial detection with surface similarity measurement tools to reduce the work load of plagiarism detection, before applying semantic similarity detection would be a good place to start.

7.4 Summary

This section summarises this thesis in terms of the problems raised, the experiments carried out to address the research problems, the findings from the experiments and the extent to which the objectives set out were achieved.

The first research question is about determining whether DCLMs could be used to enhance the detection of paraphrase plagiarism with performances comparable to a SOTA model proposed for paraphrase plagiarism detection. This question was addressed by integrating a DCLM into the generic plagiarism detection model to enhance the detection of plagiarised fragments (sentences) that may have been obfuscated using varying paraphrase expressions. A pair of common DCLM (SBERT and ELMo) taking from the two most dominant architectures (Transformer and LSTM) were integrated into the generic model and evaluated in the task of paraphrase plagiarism detection so as to determine the most suitable model for detecting paraphrase plagiarism, and to ensure that the evaluation is not limited to a particular architecture. The evaluation also include determination of specific paraphrase types that DCLMs are able to enhance their detection, and those that are challenging to detect even with the help of a DCLM, so as identify the limitations of DCLMs in the context of paraphrase plagiarism detection, and related textual similarity measurement tasks. The findings from the experiments revealed that the extension of an existing (generic) plagiarism detection framework via integration of a DCLM resulted in performance comparable and in many cases better than a number of standard baseline models, which includes the generic model itself, and a SOTA paraphrase plagiarism detector, and of the two DCLMs evaluated, SBERT (transformers) outperformed ELMo (biLSTM). Further evaluation revealed that the performance could be better when the proposed model is implemented with a DCLM fine-tuned on datasets that contain paraphrase plagiarism. In terms of the evaluation on different paraphrase types, the findings revealed that DCLMs are able to enhance the detection majority of the paraphrase types (or expressions) associated with plagiarised text, however certain paraphrase types were found challenging.

The second problem addressed in this research is about determining whether a Word2Vec embedding model could be pre-trained for multilingual translation, and applied in CLPD with performances comparable to a state-of-the-art T+MA model, but without being limited by internet translation services as a T + MA. As stated earlier, T + MA is one of the most common (and successful) approach used for CLPD, but strongly relies on internet translation services which comes with a number of limitations stated in section 2.6.5. We addressed this problem by proposing a multilingual translation model (MTM) that uses pre-trained Word2Vec model (Mikolov et al., 2013) to predict the semantic equivalent of words in other languages. The MTM was built by pre-training a Word2Vec model with simulated embeddings which comprises of contexts of semantically related words in different languages grouped in the same space. The Word2Vec model was essentially trained to reproduce the output of a common online machine translator. The MTM can be extended to multiple languages, and does not rely on parallel corpora, it is therefore designed to address the problem of detecting CLP on low resource languages that do not have sufficient amount of parallel corpora for building language translation models. The MTM was integrated into a CLPD framework (similar to a T +MA model) for translating text across languages, and evaluated in the task of CLPD. The findings from the evaluation revealed that the MTM was able to translate text with good accuracy, and that a CLP detector that uses the MTM for language translation performed comparably, and better in some cases than a standard T + MA baseline, but without the limitations that come from relying on internet translation tools as does a standard T + MA model. Results obtained from evaluation on low resource languages were good and encouraging. An unexpected outcome is the ability of the model to detect translated texts that have been altered by replacing words with semantically similar words, which is the likely reason for the superior performance over the T+MA baseline. The findings from the experiments are encouraging and clearly satisfy the research objectives.

The third research problem is about determining the best performing combination of surface similarity measurement tools for detecting plagiarised texts that have been obfuscated to varying degrees of complexity, the performance of several combinations of surface similarity measurement tools/techniques (which includes similarity measures, term weighing methods and ngram document models) were evaluated using plagiarised texts that have undergone varying degrees of obfuscation. The findings from the experiments revealed that no specific combination performed best on all obfuscation levels; combinations that performed best on some obfuscation levels, fell short on others. However a specific combination that performed best on majority of the obfuscation levels was determined, it even outperformed standard baselines (including string matching and semantic measurement methods) in many cases. In addition, the best performing combinations for detecting plagiarised texts of specific obfuscation level were also determined. The findings also confirms the hypothesis that plagiarised text often contain unaltered remains of their sources, which are surface features that could link plagiarised texts to their sources during detection. Several combinations have been recommended for detecting plagiarised texts of different obfuscation levels, and the need to integrate efficient semantic methods into the recommended combinations to enhance the detection of obfuscation plagiarism (without significant compromise to the efficiency of surface matching tools) represent the extent to which the objectives were achieved.

7.5 Conclusion

This thesis addressed some of the challenges currently facing plagiarism detection, which includes paraphrase and cross-lingual plagiarism. New and existing state-of-the-art tools and methods used in NLP and IR were combined to address these challenges.

A number of novel contributions emerged as a consequence of the experiments carried out in this research to address the above research problems, notable ones include a novel deep paraphrase plagiarism detector (DPPD) that integrates deep contextualised learning (DCL) into a generic plagiarism detection framework to enhance the detection of paraphrase plagiarism. A novel multilingual translation model (MTM) and a cross lingual plagiarism (CLP) detector that utilised the MTM in CLP detection were proposed. The MTM is a pre-trained word embedding model (i.e. Word2Vec) designed to translate text across multiple languages, including low resource languages where parallel corpora are not available in sufficient amount for training translation models. Currently one of the most efficient (but not so effective) method for detecting plagiarism involves using surface similarity measurement tools/techniques (i.e. ngrams models, similarity measures, term weighting methods), in order to improve their effectiveness novel combinations of surface similarity measurement tools/techniques were proposed for detecting plagiarised texts that have been obfuscated to different levels of complexity.

In relation to the literature, some of the findings obtained in this research correlate closely with results from similar studies in the literature, for the most part, the results obtained in this research were comparably, and in many cases better than those from standard baselines, state-of-the-art models and previous studies. The superior performance of some of the models proposed in this research is largely due to the integration of state-of-the-art NLP tools into existing plagiarism detection frameworks. Details of the limitations of this research, which seem common to most research on plagiarism detection, including the lack of real world evaluation corpus and framework can be found in section 7.2. Details of the contributions to knowledge that emerged from this research can be found in the introductory chapter in section 1.4. While the objectives of this research were achieved to a reasonable extent, the thesis leaves room for future research work which are highlighted in section 7.3.

Relevant Publications From this thesis

Thompson, V. U., Panchev, C., and Oakes, M. (2015, November). Performance evaluation of similarity measures on similar and dissimilar text retrieval. In *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)* (Vol. 1, pp. 577-584). IEEE.

Thompson, V. (2017). Detecting cross-lingual plagiarism using simulated word embeddings. *arXiv preprint arXiv:1712.10190*.

Publication relevant to research area, but not included in thesis

Thompson, V. (2017). Methods for Detecting Paraphrase Plagiarism. *arXiv preprint arXiv:1712.10309*.

References

Aamir, M., and Zaidi, S. M. A. (2021). Clustering based semi-supervised machine learning for DDoS attack classification. *Journal of King Saud University-Computer and Information Sciences, 33*(4), 436-446.

Achananuparp, P., Hu, X., & Shen, X. (2008, September). The evaluation of sentence similarity measures. In *International Conference on data warehousing and knowledge discovery* (pp. 305-316). Springer, Berlin, Heidelberg.

Agirre, E., Banea, C., Cer, D., Diab, M., Gonzalez Agirre, A., Mihalcea, R., and Wiebe, J. (2016). Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval-2016. 10th International Workshop on Semantic Evaluation; 2016 Jun 16-17; San Diego, CA. Stroudsburg (PA): ACL; 2016. p. 497-511.* ACL (Association for Computational Linguistics).

Agrawal, M., and Sharma, D. K. (2016, October). A state of art on source code plagiarism detection. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)* (pp. 236-241). IEEE.

Ahuja, Lovepreet, Vishal Gupta, and Rohit Kumar. "A new hybrid technique for detection of plagiarism from text documents." *Arabian Journal for Science and Engineering* 45, no. 12 (2020): 9939-9952

Álvarez-Carmona, M. A., Franco-Salvador, M., Villatoro-Tello, E., Montes-y-Gómez, M., Rosso, P., and Villaseñor-Pineda, L. (2018). Semantically-informed distance and similarity measures for paraphrase plagiarism identification. *Journal of Intelligent and Fuzzy Systems, 34*(5), 2983-2990.

Alvi, F. (2020). *Monolingual Plagiarism Detection and Paraphrase Type Identification* (Doctoral dissertation, University of Sheffield).

Alvi, F., Stevenson, M., and Clough, P. (2017, April). Plagiarism detection in texts obfuscated with homoglyphs. In *European Conference on Information Retrieval* (pp. 669-675). Springer, Cham.

Alvi, F., Stevenson, M., and Clough, P. (2021). Paraphrase type identification for plagiarism detection using contexts and word embeddings. *International Journal of Educational Technology in Higher Education*, *18*(1), 1-25.

Amer, A. A., Abdalla, H. I., and Nguyen, L. (2021). Enhancing recommendation systems performance using highly-effective similarity measures. *Knowledge-Based Systems*, *217*, 106842.

Amer, A. A., and Abdalla, H. I. (2020). A set theory based similarity measure for text clustering and classification. *Journal of Big Data*, *7*(1), 1-43.

Amigó, E., Giner, F., Gonzalo, J., and Verdejo, F. (2020). On the foundations of similarity in information access. *Information Retrieval Journal*, *23*(3), 216-254.

Amzuloiu, C., Mihăescu, M. C., and Rebedea, T. (2021). Combining Encoplot and NLP Based Deep Learning for Plagiarism Detection. In *Intelligent Data Engineering and Automated Learning–IDEAL 2021: 22nd International Conference, IDEAL 2021, Manchester, UK, November 25–27, 2021, Proceedings 22* (pp. 97-106). Springer International Publishing.

Arora, S., Liang, Y., and Ma, T. (2016). A simple but tough-to-beat baseline for sentence embeddings.

Ayub, M., Ghazanfar, M. A., Khan, T., and Saleem, A. (2020). An effective model for Jaccard coefficient to increase the performance of collaborative filtering. *Arabian Journal for Science and Engineering*, *45*(12), 9997-10017.

Baak, M., Koopman, R., Snoek, H., and Klous, S. (2020). A new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics. *Computational Statistics and Data Analysis*, *152*, 107043.

Baba, K., Nakatoh, T., and Minami, T. (2017). Plagiarism detection using document similarity based on distributed representation. *Procedia computer science*, *111*, 382-387.

Bär, D., Zesch, T., and Gurevych, I. (2012, December). Text reuse detection using a composition of text similarity measures. In *Proceedings of COLING 2012* (pp. 167-184).

Barbouch, M., Verberne, S., and Verhoef, T. (2021). WN-BERT: Integrating WordNet and BERT for Lexical Semantics in Natural Language Understanding. *Computational Linguistics in the Netherlands Journal*, *11*, 105-124.

Barrón-Cedeño, A., & Rosso, P. (2009, April). On automatic plagiarism detection based on n-grams comparison. In *European conference on information retrieval* (pp. 696-700). Springer, Berlin, Heidelberg.

Barrón-Cedeño, A., Gupta, P., and Rosso, P. (2013). Methods for cross-language plagiarism detection. *Knowledge-Based Systems*, *50*, 211-217.

Barrón-Cedeno, A., Rosso, P., and Benedí, J. M. (2009, March). Reducing the plagiarism detection search space on the basis of the kullback-leibler distance. In *International conference on intelligent text processing and computational linguistics* (pp. 523-534). Springer, Berlin, Heidelberg.

Barrón-Cedeño, A., Vila, M., Martí, M. A., and Rosso, P. (2013). Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. *Computational Linguistics*, *39*(4), 917-947.

Bensalem, I., Rosso, P., and Chikhi, S. (2019). On the use of character n-grams as the only intrinsic evidence of plagiarism. *Language Resources and Evaluation*, *53*(3), 363-396.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Brychcín, T. (2020). Linear transformations for cross-lingual semantic textual similarity. *Knowledge-Based Systems*, *187*, 104819.

Bülow, W., and Helgesson, G. (2019). Criminalization of scientific misconduct. *Medicine, Health Care and Philosophy*, *22*(2), 245-252.

Burrows, S., Potthast, M., and Stein, B. (2013). Paraphrase acquisition via Crowdsourcingand machine learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *4*(3), 1-21.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., and Kurzweil, R. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Ceska, Z. (2008, August). Plagiarism detection based on singular value decomposition. In *International Conference on Natural Language Processing* (pp. 108-119). Springer, Berlin, Heidelberg.

Ceska, Z., and Fox, C. (2011). The influence of text pre-processing on plagiarism detection. Association for Computational Linguistics.

Chandrasekaran, D., and Mago, V. (2021). Evolution of semantic similarity—a survey. *ACM Computing Surveys (CSUR)*, *54*(2), 1-37.

Chen, C. Y., Yeh, J. Y., and Ke, H. R. (2010). Plagiarism detection using ROUGE and WordNet. *arXiv preprint arXiv:1003.4065*.

Chong, M., and Specia, L. (2011, September). Lexical generalisation for word-level matching in plagiarism detection. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011* (pp. 704-709).

Clough, P., and Stevenson, M. (2011). Developing a corpus of plagiarised short answers. *Language resources and evaluation*, *45*(1), 5-24.

Clough, P., Willett, P., and Lim, J. (2015, September). Unfair means: use cases beyond plagiarism. In *International Conference of the Cross-Language Evaluation Forum for European Languages* (pp. 229-234). Springer, Cham.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Dal-Ré, R., Bouter, L. M., Cuijpers, P., Gluud, C., and Holm, S. (2020). Should research misconduct be criminalized?. *Research Ethics*, 174701611989840.

Davis, J., and Goadrich, M. (2006, June). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning* (pp. 233-240).

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *JAsIs*, *41*(6), 391-407.

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

Devore-McDonald, B., and Berger, E. D. (2020). Mossad: Defeating software plagiarism detection. *Proceedings of the ACM on Programming Languages*, *4*(OOPSLA), 1-28.

Diallo, B., Hu, J., Li, T., Khan, G. A., and Hussein, A. S. (2022). Multi-view document clustering based on geometrical similarity measurement. *International Journal of Machine Learning and Cybernetics*, *13*(3), 663-675.

Dogan, T., and Uysal, A. K. (2020). A novel term weighting scheme for text classification: TF-MONO. *Journal of Informetrics, 14*(4), 101076.

Domeniconi, G., Moro, G., Pasolini, R., and Sartori, C. (2015, July). A Study on Term Weighting for Text Categorization: A Novel Supervised Variant of tf. idf. In *DATA* (pp. 26-37).

Dougherty, M. V. (2020). Translation Plagiarism. In *Disguised Academic Plagiarism* (pp. 13-36). Springer, Cham.

Draper, M., Lancaster, T., Dann, S., Crockett, R., and Glendinning, I. (2021). Essay mills and other contract cheating services: to buy or not to buy and the consequences of students changing their minds. *International Journal for Educational Integrity*, *17*(1), 1-13.

Eaton, S. E. (2021). *Plagiarism in higher education: Tackling tough topics in academic integrity*. ABC-CLIO.

Ehsan, N., and Shakery, A. (2016). A Pairwise Document Analysis Approach for Monolingual Plagiarism Detection. In *FIRE (Working Notes)* (pp. 145-148).

Ehsan, N., Tompa, F. W., and Shakery, A. (2016, September). Using a dictionary and n-gram alignment to improve fine-grained cross-language plagiarism detection. In *Proceedings of the 2016 ACM Symposium on Document Engineering* (pp. 59-68).

Eisa, T. A. E., Salim, N., and Alzahrani, S. (2015). Existing plagiarism detection techniques: A systematic mapping of the scholarly literature. *Online Information Review*.

Ekbal, A., Saha, S., and Choudhary, G. (2012, December). Plagiarism detection in text using vector space model. In *2012 12th international conference on hybrid intelligent systems (HIS)* (pp. 366-371). IEEE.

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, *19*(1), 1-16.

Espana-Bonet, C., and Barrón-Cedeno, A. (2017, August). Lump at SemEval-2017 task 1: towards an interlingua semantic similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 144-149).

Ethayarajh, K. (2019). How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. *arXiv preprint arXiv:1909.00512*.

Farouk, M. (2019). Measuring sentences similarity: a survey. *arXiv preprint arXiv:1910.03940*.

Faruqui, M., and Dyer, C. (2014, April). Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 462-471).

Ferrero, J., Agnes, F., Besacier, L., and Schwab, D. (2017a). Using word embedding for cross-language plagiarism detection. *arXiv preprint arXiv:1702.03082*.

Ferrero, J., Agnes, F., Besacier, L., and Schwab, D. (2017c). CompiLIG at SemEval-2017 Task 1: Cross-language plagiarism detection methods for semantic textual similarity. *arXiv preprint arXiv:1704.01346*.

Ferrero, J., Besacier, L., Schwab, D., and Agnes, F. (2017b). Deep investigation of cross-language plagiarism detection methods. *arXiv preprint arXiv:1705.08828*.

Foltýnek, T., Dlabolová, D., Anohina-Naumeca, A., Razı, S., Kravjar, J., Kamzola, L., and Weber-Wulff, D. (2020). Testing of support tools for plagiarism detection. *International Journal of Educational Technology in Higher Education*, *17*(1), 1-31.

Foltýnek, T., Meuschke, N., and Gipp, B. (2019). Academic plagiarism detection: a systematic literature review. *ACM Computing Surveys (CSUR)*, *52*(6), 1-42

Foltýnek, T., Ruas, T., Scharpf, P., Meuschke, N., Schubotz, M., Grosky, W., and Gipp, B. (2020, March). Detecting machine-obfuscated plagiarism. In *Sustainable Digital Communities: 15th International Conference, iConference 2020, Boras, Sweden, March 23–26, 2020, Proceedings* (pp. 816-827). Cham: Springer International Publishing.

Foltýnek, T., Všianský, R., Meuschke, N., Dlabolová, D., and Gipp, B. (2020, August). Cross-language source code plagiarism detection using explicit semantic analysis and scored greedy string tilling. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020* (pp. 523-524).

Franco-Salvador, M., Gupta, P., and Rosso, P. (2013, February). Knowledge graphs as context models: Improving the detection of cross-language plagiarism with paraphrasing. In *PROMISE Winter School* (pp. 227-236). Springer, Berlin, Heidelberg.

Franco-Salvador, M., Gupta, P., and Rosso, P. (2013, March). Cross-language plagiarism detection using a multilingual semantic network. In *European Conference on Information Retrieval* (pp. 710-713). Springer, Berlin, Heidelberg.

Franco-Salvador, M., Gupta, P., Rosso, P., and Banchs, R. E. (2016). Cross-language plagiarism detection over continuous-space-and knowledge graph-based representations of language. *Knowledge-based systems*, *111*, 87-99.

Gabrilovich, E., and Markovitch, S. (2007, January). Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJcAI* (Vol. 7, pp. 1606-1611).

Gali, N., Mariescu-Istodor, R., Hostettler, D., and Fränti, P. (2019). Framework for syntactic string similarity measures. *Expert Systems with Applications*, *129*, 169-185.

Ghosh, J., and Strehl, A. (2006). Similarity-based text clustering: A comparative study. In *Grouping Multidimensional Data* (pp. 73-97). Springer, Berlin, Heidelberg.

Gienapp, L., Kircheis, W., Sievers, B., Stein, B., & Potthast, M. (2023). A large dataset of scientific text reuse in Open-Access publications. *Scientific Data*, *10*(1), 58.

Gomaa, W. H., and Fahmy, A. A. (2013). A survey of text similarity approaches.*International Journal of Computer Applications*, *68*(13), 13-18.

Grozea, C., and Popescu, M. (2012, September). Encoplot-Tuned for High Recall (also Proposing a New Plagiarism Detection Score). In *CLEF (Online Working Notes/Labs/Workshop)*.

Gupta, D. (2016). Study on Extrinsic Text Plagiarism Detection Techniques and Tools. *Journal of Engineering Science and Technology Review*, *9*(5).

Gupta, D., Vani, K., and Leema, L. M. (2016). Plagiarism detection in text documents using sentence bounded stop word n-grams. *Journal of Engineering Science and Technology*, *11*(10), 1403-1420.

HaCohen-Kerner, Y., and Tayeb, A. (2017). Rapid detection of similar peer-reviewed scientific papers via constant number of randomized fingerprints. *Information Processing and Management*, *53*(1), 70-86.

Hagen, M., Potthast, M., and Stein, B. (2015). Source Retrieval for Plagiarism Detection from Large Web Corpora: Recent Approaches. *CLEF (Working Notes)*.

Halak, B., and El-Hajjar, M. (2019). Design and evaluation of plagiarism prevention and detection techniques in engineering education. *Higher Education Pedagogies*, *4*(1), 197-208.

Heo, J., Won, J., Lee, Y., Bharuka, S., Jang, J., Ham, T. J., and Lee, J. W. (2020, March). IIU: Specialized architecture for inverted index search. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (pp. 1233-1245).

Hourrane, O., and Benlahmar, E. H. (2017, March). Survey of plagiarism detection approaches and big data techniques related to plagiarism candidate retrieval. In *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications* (pp. 1-6).

Jayapal, A. (2012) Similarity Overlap Metric and Greedy String Tiling at PAN 2012: Plagiarism Detection.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kaur, M., Gupta, V., and Kaur, R. (2021). Review of Recent Plagiarism Detection Techniques and Their Performance Comparison. In *Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications* (pp. 157-170). Springer, Singapore.

Koehn, P., & Knowles, R. (2017). Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Kong, L. L., Han, Z. Y., Qi, H. L., and Yang, M. Y. (2019). Source retrieval model focused on aggregation for plagiarism detection. *Information Sciences*, *503*, 336-350.

Krokoscz, M. (2021). Plagiarism in articles published in journals indexed in the Scientific Periodicals Electronic Library (SPELL): a comparative analysis between 2013 and 2018. *International Journal for Educational Integrity*, *17*(1), 1-22.

Krokoscz, M. (2021). Plagiarism in articles published in journals indexed in the Scientific Periodicals Electronic Library (SPELL): a comparative analysis between 2013 and 2018. *International Journal for Educational Integrity*, *17*(1), 1-22.

Kumar, A., and Garg, G. (2019). Empirical study of shallow and deep learning models for sarcasm detection using context in benchmark datasets. *Journal of Ambient Intelligence and Humanized Computing*, 1-16.

Lancaster, T. (2020). Academic discipline integration by contract cheating services and essay Mills. *Journal of Academic Ethics*, *18*(2), 115-127.

Lane, P. C., Lyon, C., and Malcolm, J. A. (2006). Demonstration of the Ferret plagiarism detector. In *Proceedings of the 2nd international plagiarism conference*.

Lau, J. H., and Baldwin, T. (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

Lauriola, I., Lavelli, A., and Aiolli, F. (2022). An introduction to deep learning in natural language processing: Models, techniques, and tools. *Neurocomputing*, *470*, 443-456.

Le, Q., and Mikolov, T. (2014, June). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188-1196). PMLR.

Leech, G. (1992). 100 million words of English: the British National Corpus (BNC). *Language research*, *28*(1), 1-13.

Levada, A. L., and Haddad, M. F. (2021, November). A Kullback-Leibler Divergence-Based Locally Linear Embedding Method: A Novel Parametric Approach for Cluster Analysis. In *Brazilian Conference on Intelligent Systems* (pp. 406-420). Springer, Cham.

Liao, M., Shi, B., Bai, X., Wang, X., and Liu, W. (2017, February). Textboxes: A fast text detector with a single deep neural network. In *Thirty-first AAAI conference on artificial intelligence*.

Lyon, C., Barrett, R., and Malcolm, J. (2004). A theoretical basis to the automated detection of copying between texts, and its practical implementation in the Ferret plagiarism and collusion detector. *Plagiarism: Prevention, Practice and Policies*.

Magara, M. B., Ojo, S. O., and Zuva, T. (2018, March). A comparative analysis of text similarity measures and algorithms in research paper recommender systems. In *2018 conference on information communications technology and society (ICTAS)* (pp. 1-5). IEEE.

Marcos-Pablos, S., and García-Peñalvo, F. J. (2020). Information retrieval methodology for aiding scientific database search. *Soft Computing*, *24*(8), 5551-5560.

Matusov, E. (2019, August). The challenges of using neural machine translation for literature. In *Proceedings of the qualities of literary machine translation* (pp. 10-19).

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: Contextualized word vectors. *Advances in neural information processing systems*, *30*.

McNamee, P., and Mayfield, J. (2004). Character n-gram tokenization for European language text retrieval. *Information retrieval*, *7*(1), 73-97.

Meuschke, N., Siebeck, N., Schubotz, M., and Gipp, B. (2017, December). Analyzing semantic concept patterns to detect academic plagiarism. In *Proceedings of the 6th international workshop on mining scientific publications* (pp. 46-53).

Meuschke, N., Stange, V., Schubotz, M., and Gipp, B. (2018, June). HyPlag: a hybrid approach to academic plagiarism detection. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1321-1324).

Michael P. Oakes, (2014). "*Literary Detective Work on the Computer*", Amsterdam: John Benjamins.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems, 26*.

Miller, D. (2019). Leveraging BERT for extractive text summarization on lectures. *arXiv preprint arXiv:1906.04165*.

Mohamed, M., and Oussalah, M. (2020). A hybrid approach for paraphrase identification based on knowledge-enriched semantic heuristics. *Language Resources and Evaluation*, *54*(2), 457-485.

Moradi, M., Dorffner, G., and Samwald, M. (2020). Deep contextualized embeddings for quantifying the informative content in biomedical text summarization. *Computer methods and programs in biomedicine*, *184*, 105117.

Mosco, M. (2021). Plagiarism and Copyright: An Analysis of Technical Communication Textbooks. *Technical Communication*, *68*(2), 87-102.

Mozgovoy, M., Kakkonen, T., & Cosma, G. (2010). Automatic student plagiarism detection: future perspectives. *Journal of Educational Computing Research*, *43*(4), 511-531.

Nawab, R. M. A., Stevenson, M., and Clough, P. (2012, April). Retrieving candidate plagiarised documents using query expansion. In *European Conference on Information Retrieval* (pp. 207-218). Springer, Berlin, Heidelberg.

Osman, A. H., Salim, N., Binwahlan, M. S., Alteeb, R., and Abuobieda, A. (2012). An improved plagiarism detection scheme based on semantic role labeling. *Applied Soft Computing*, *12*(5), 1493-1502.

Pagliardini, M., Gupta, P., and Jaggi, M. (2017). Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

Patra, B. K., Launonen, R., Ollikainen, V., and Nandi, S. (2015). A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data. *Knowledge-Based Systems*, *82*, 163-177.

Paul, M., and Jamal, S. (2015). An improved SRL based plagiarism detection technique using sentence ranking. *Procedia Computer Science*, *46*, 223-230.

Pennington, J., Socher, R., and Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

Perkins, M., Gezgin, U. B., and Roe, J. (2020). Reducing plagiarism through academic misconduct education. *International Journal for Educational Integrity*, *16*(1), 1-15.

Pertile, S. D. L., Moreira, V. P., and Rosso, P. (2016). Comparing and combining content-and citation-based approaches for plagiarism detection. *Journal of the Association for Information Science and Technology*, *67*(10), 2511-2526.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Phang, J., Févry, T., and Bowman, S. R. (2018). Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Pires, T., Schlinger, E., and Garrette, D. (2019). How multilingual is multilingual BERT?. *arXiv preprint arXiv:1906.01502*.

Potthast, M., Hagen, M., Gollub, T., Tippmann, M., Kiesel, J., Rosso, P and Stein, B. (2013). Overview of the 5th international competition on plagiarism detection. In *CLEF Conference on Multilingual and Multimodal Information Access Evaluation* (pp. 301-331). CELCT.

Potthast, M., Rosso, P., Stamatatos, E., and Stein, B. (2019, April). A decade of shared tasks in digital text forensics at PAN. In *European Conference on Information Retrieval* (pp. 291-300). Springer, Cham.

Qaiser, S., and Ali, R. (2018). Text mining: use of TF-IDF to examine the relevance of words to documents. *International Journal of Computer Applications*, *181*(1), 25-29.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

Ratna, A. A. P., Purnamasari, P. D., Adhi, B. A., Ekadiyanto, F. A., Salman, M., Mardiyah, M., and Winata, D. J. (2017). Cross-language plagiarism detection system using latent semantic analysis and learning vector quantization. *Algorithms*, *10*(2), 69.

Ravi, R., and Gupta, D. (2015). D. Efficient Paragraph based Chunking and Download Filtering for Plagiarism Source Retrieval—Notebook for PAN at CLEF 2015. In: [2.

Reimers, N., and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. *arXiv preprint arXiv:1908.10084*.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation*.

Saiyed, S., and Sajja, P. (2022). Empirical Analysis of Static and Dynamic Stopword Generation Approaches. In *ICT Systems and Sustainability* (pp. 149-156). Springer, Singapore.

Salloum, S. A., Khan, R., and Shaalan, K. (2020, April). A survey of semantic analysis approaches. In *The International Conference on Artificial Intelligence and Computer Vision* (pp. 61-70). Springer, Cham.

Sanchez-Perez, M. A., Sidorov, G., and Gelbukh, A. F. (2014, September). A Winning Approach to Text Alignment for Text Reuse Detection at PAN 2014. In *CLEF (Working Notes)* (pp. 1004-1011).

Sánchez-Vega, F., Villatoro-Tello, E., Montes-y-Gómez, M., Rosso, P., Stamatatos, E., and Villasenor-Pineda, L. (2019). Paraphrase plagiarism identification with character-level features. *Pattern Analysis and Applications*, *22*(2), 669-681.

Singh, P. K., Sinha, M., Das, S., and Choudhury, P. (2020). Enhancing recommendation accuracy of item-based collaborative filtering using Bhattacharyya coefficient and most similar item. *Applied Intelligence*, *50*(12), 4708-4731

Sohangir, S., and Wang, D. (2017). Improved sqrt-cosine similarity measurement. *Journal of Big Data*, *4*(1), 1-13.

Stamatatos, E. (2011). Plagiarism detection using stopword n-grams. *Journal of the American Society for Information Science and Technology*, *62*(12), 2512-2527.

Sueno, H. T., Gerardo, B. D., and Medina, R. P. (2020). Multi-class document classification using support vector machine (SVM) based on improved Naïve bayes vectorization technique. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(3).

Sun, C., Qiu, X., Xu, Y., and Huang, X. (2019, October). How to fine-tune BERT for text classification?. In *China National Conference on Chinese Computational Linguistics* (pp. 194-206). Springer, Cham.

Sun, Y. C., and Yang, F. Y. (2015). Uncovering published authors' text-borrowing practices: Paraphrasing strategies, sources, and self-plagiarism. *Journal of English for Academic Purposes, 20,* 224-236.

Szymura, A. (2022). Risk Assessment of Polish Joint Stock Companies: Prediction of Penalties or Compensation Payments. *Risks*, *10*(5), 102.

Thompson, V. U., Panchev, C., and Oakes, M. (2015, November). Performance evaluation of similarity measures on similar and dissimilar text retrieval. In *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)* (Vol. 1, pp. 577-584). IEEE.

Thompson, V. (2017). Detecting cross-lingual plagiarism using simulated word embeddings. *arXiv preprint arXiv:1712.10190*.

Tian, J., Zhou, Z., Lan, M., and Wu, Y. (2017, August). Ecnu at semeval-2017 task 1: Leverage kernel-based traditional nlp features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)* (pp. 191-197).

Van Molle, P., Verbelen, T., Vankeirsbilck, B., De Vylder, J., Diricx, B., Kimpe, T., and Dhoedt, B. (2021). Leveraging the Bhattacharyya coefficient for uncertainty quantification in deep neural networks. *Neural Computing and Applications, 33*(16), 10259-10275.

Vani, K., and Gupta, D. (2015, August). Investigating the impact of combined similarity metrics and POS tagging in extrinsic text plagiarism detection system. In *2015 international conference on advances in computing, communications and informatics (ICACCI)* (pp. 1578-1584). IEEE.

Vani, K., and Gupta, D. (2017). Detection of idea plagiarism using syntax–semantic concept extractions with genetic algorithm. *Expert Systems with Applications*, *73*, 11-26.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vaux, D. L. (2016). Scientific misconduct: falsification, fabrication, and misappropriation of credit. *Handbook of Academic Integrity. Singapore: Springer*, 895-911.

Velásquez, J. D., Covacevich, Y., Molina, F., Marrese-Taylor, E., Rodríguez, C., and Bravo-Marquez, F. (2016). DOCODE 3.0 (DOcument COpy DEtector): A system for plagiarism detection by applying an information fusion process from multiple documental data sources. *Information Fusion*, *27,* 64-75

Verma, V., and Aggarwal, R. K. (2020). A comparative analysis of similarity measures akin to the Jaccard index in collaborative recommendations: empirical and theoretical perspective. *Social Network Analysis and Mining*, *10*(1), 1-16.

Wahle, J. P., Ruas, T., Foltýnek, T., Meuschke, N., and Gipp, B. (2022, February). Identifying machine-paraphrased plagiarism. In *International Conference on Information* (pp. 393-413). Springer, Cham.

Wan, Y., Yang, B., Wong, D. F., Chao, L. S., Yao, L., Zhang, H., and Chen, B. (2022). Challenges of neural machine translation for short texts. *Computational Linguistics*, *48*(2), 321-342.

Wu, S., Zhang, D., Yang, N., Li, M., and Zhou, M. (2017, July). Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 698-707).

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zechner, M., Muhr, M., Kern, R., and Granitzer, M. (2009). External and intrinsic plagiarism detection using vector space models. Proc. Of SEPLN, Spain, 47–55.

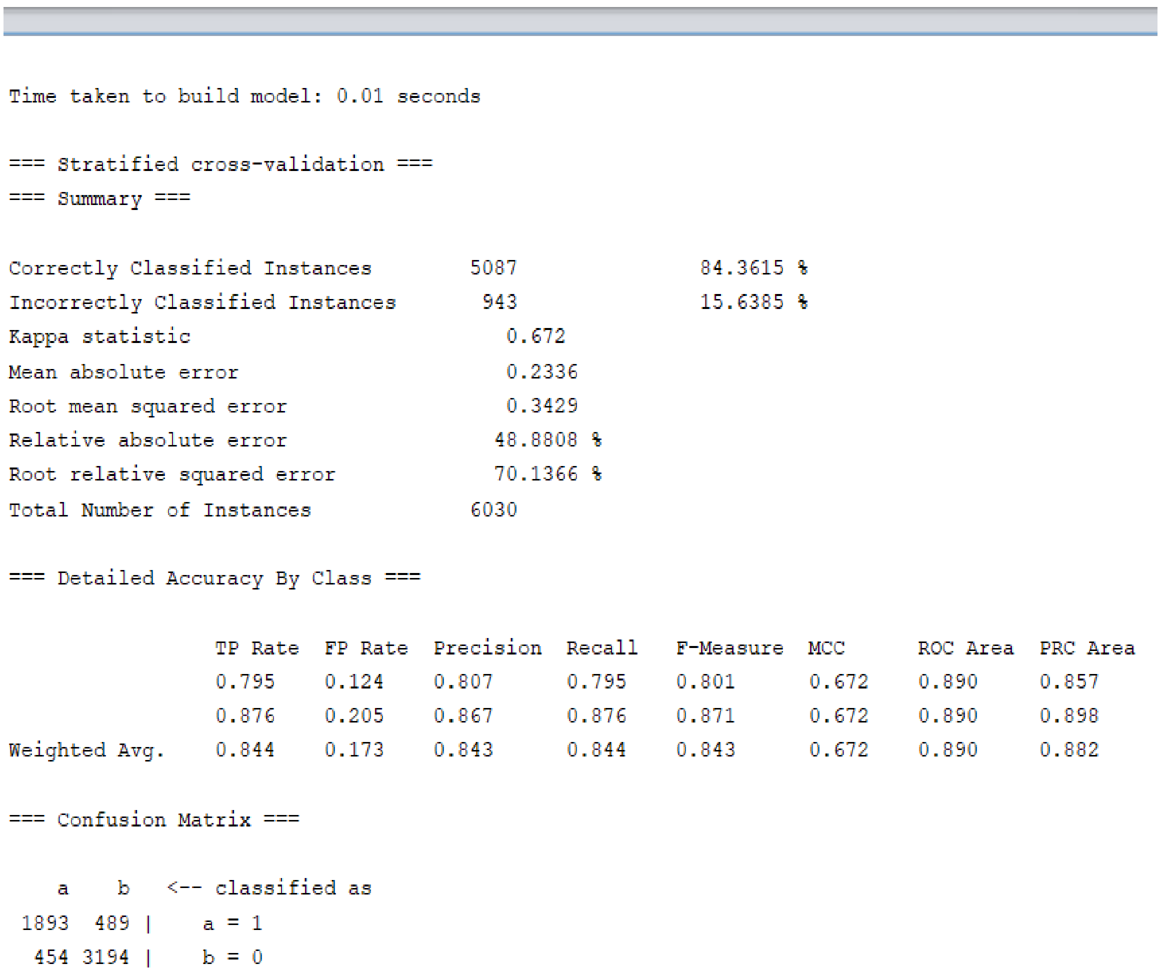
Zimba, O., and Gasparyan, A. (2021). Plagiarism detection and prevention: a primer for researchers. *Reumatologia/Rheumatology*, *59*(3), 132-137.

APPENDICES

Appendix A: Results Tables and Graphs

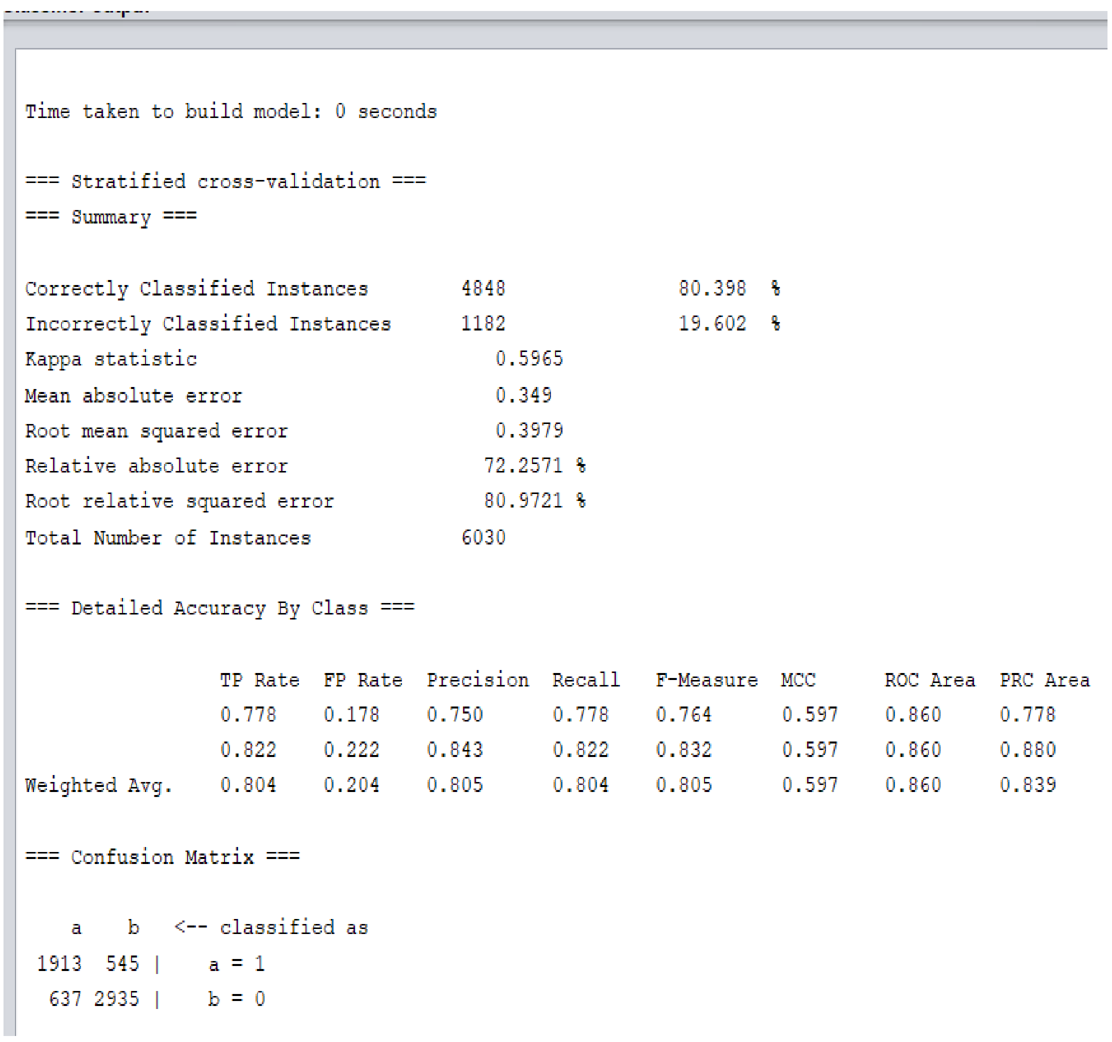
Results for DCLMs and Baselines on P4P Corpus

BERT



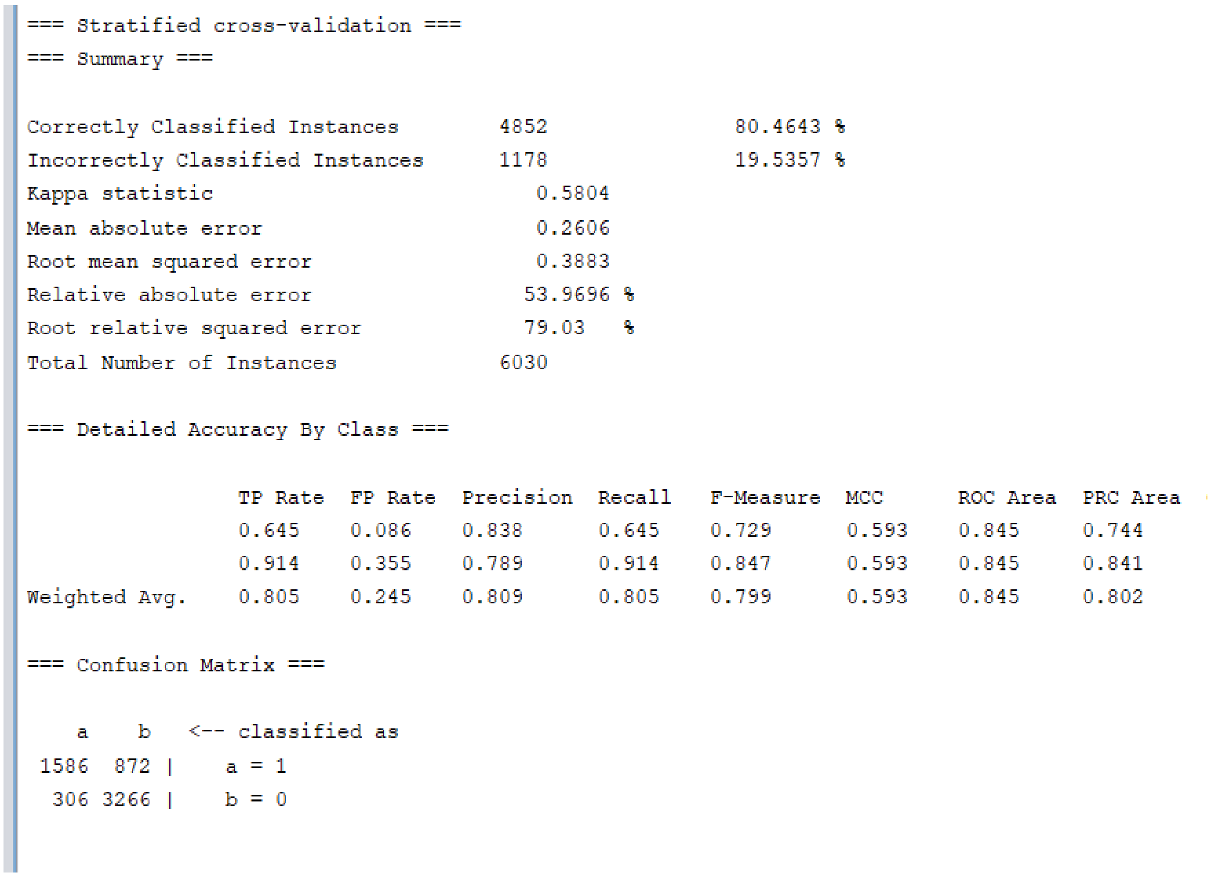
```
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        5087               84.3615 %
Incorrectly Classified Instances       943               15.6385 %
Kappa statistic                          0.672
Mean absolute error                      0.2336
Root mean squared error                  0.3429
Relative absolute error                 48.8808 %
Root relative squared error             70.1366 %
Total Number of Instances             6030

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.795    0.124    0.807      0.795    0.801      0.672    0.890     0.857
                 0.876    0.205    0.867      0.876    0.871      0.672    0.890     0.898
Weighted Avg.    0.844    0.173    0.843      0.844    0.843      0.672    0.890     0.882

=== Confusion Matrix ===

    a    b   <-- classified as
 1893  489 |    a = 1
  454 3194 |    b = 0
```

CoVe



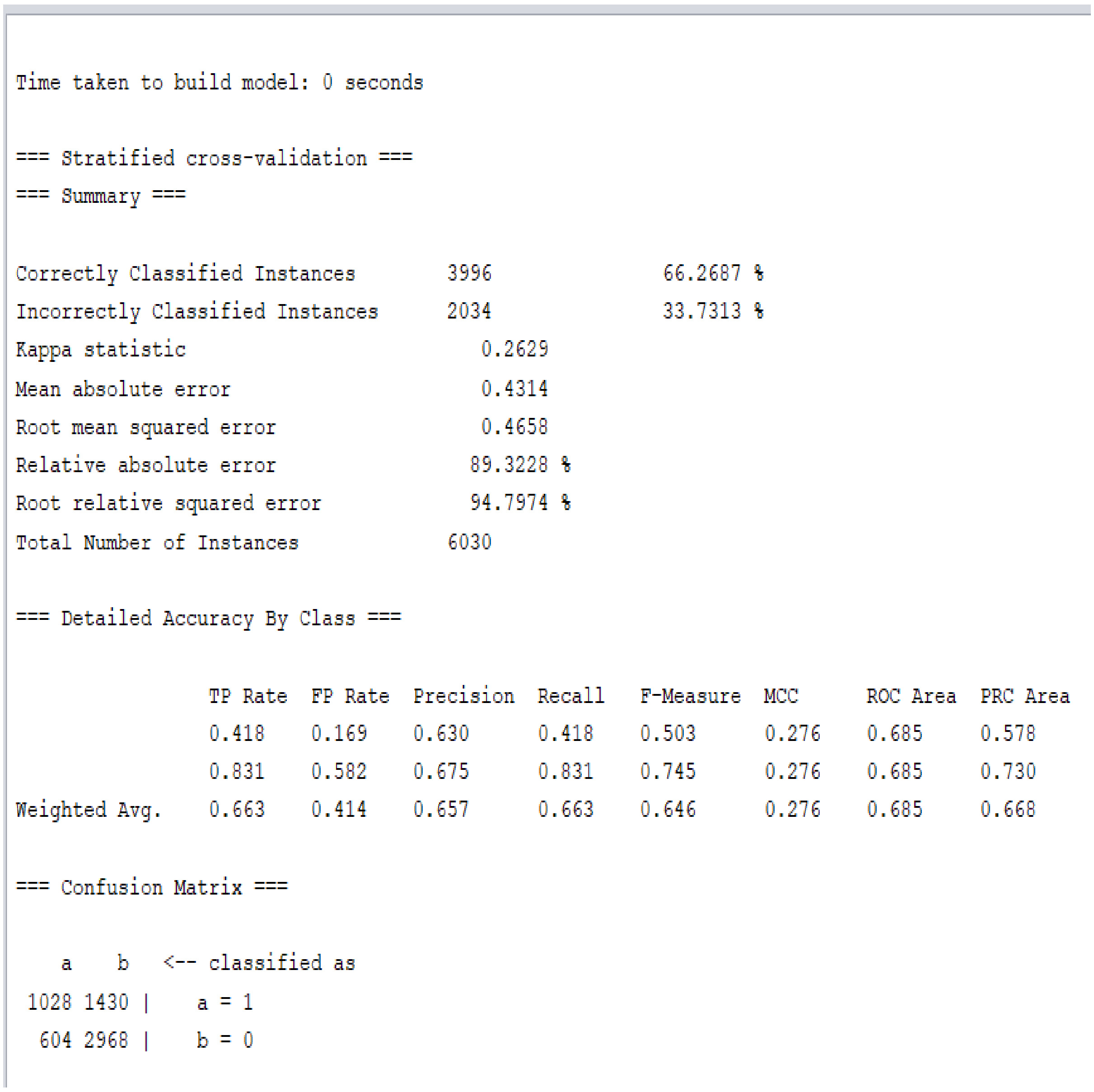
```
Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4848               80.398  %
Incorrectly Classified Instances      1182               19.602  %
Kappa statistic                          0.5965
Mean absolute error                      0.349
Root mean squared error                  0.3979
Relative absolute error                 72.2571 %
Root relative squared error             80.9721 %
Total Number of Instances             6030

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.778    0.178    0.750      0.778    0.764      0.597    0.860     0.778
                 0.822    0.222    0.843      0.822    0.832      0.597    0.860     0.880
Weighted Avg.    0.804    0.204    0.805      0.804    0.805      0.597    0.860     0.839

=== Confusion Matrix ===

    a    b   <-- classified as
 1913  545 |    a = 1
  637 2935 |    b = 0
```

GloVe



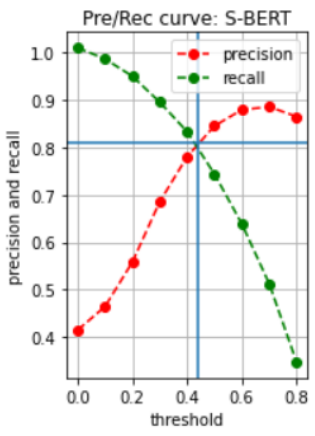
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4852               80.4643 %
Incorrectly Classified Instances      1178               19.5357 %
Kappa statistic                          0.5804
Mean absolute error                      0.2606
Root mean squared error                  0.3883
Relative absolute error                 53.9696 %
Root relative squared error             79.03   %
Total Number of Instances             6030

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.645    0.086    0.838      0.645    0.729      0.593    0.845     0.744
                 0.914    0.355    0.789      0.914    0.847      0.593    0.845     0.841
Weighted Avg.    0.805    0.245    0.809      0.805    0.799      0.593    0.845     0.802

=== Confusion Matrix ===

    a    b   <-- classified as
 1586  872 |    a = 1
  306 3266 |    b = 0
```

Cosine

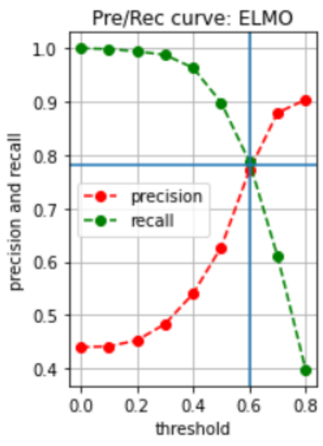


```
Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3996               66.2687 %
Incorrectly Classified Instances      2034               33.7313 %
Kappa statistic                          0.2629
Mean absolute error                      0.4314
Root mean squared error                  0.4658
Relative absolute error                 89.3228 %
Root relative squared error             94.7974 %
Total Number of Instances             6030

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.418    0.169    0.630      0.418    0.503      0.276    0.685     0.578
                 0.831    0.582    0.675      0.831    0.745      0.276    0.685     0.730
Weighted Avg.    0.663    0.414    0.657      0.663    0.646      0.276    0.685     0.668

=== Confusion Matrix ===

    a    b   <-- classified as
 1028 1430 |    a = 1
  604 2968 |    b = 0
```

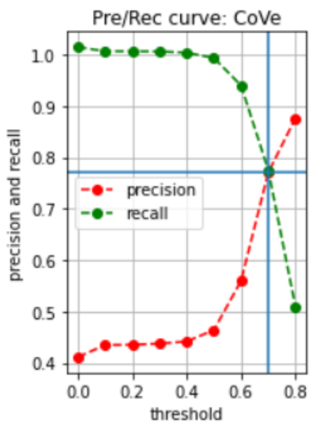
Detailed Results from the Experiment to Detect Paraphrase Types Embedded In Plagiarised Texts (P4P corpus).



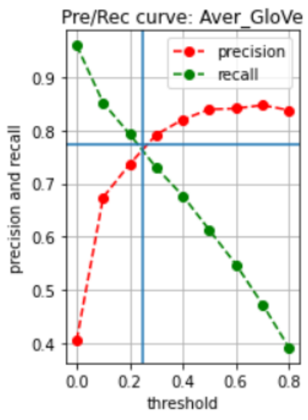
```
('lex_same_polarity', 0.913232104121475, 4631, 5071)
('syn_ellipsis', 0.9770114942528736, 85, 87)
('lex_opposite_polarity', 0.8461538461538461, 55, 65)
('semantic', 0.8294117647058824, 282, 340)
('dis_punct_format', 0.8234200743494424, 443, 538)
('lex_converse', 0.9090909090909091, 30, 33)
('lex_spelling_and_format', 0.8901601830663616, 389, 437)
('syn_subord_nesting', 0.9262981574539364, 553, 597)
('dis_direct_indirect', 0.5277777777777778, 19, 36)
('syn_dis_structure', 0.9488817891373802, 297, 313)
('syn_negation', 0.9696969696969697, 32, 33)
('mor_modal_verb', 0.853448275862069, 99, 116)
('identical', 0.9603960396039604, 97, 101)
('dis_sent_modality', 0.8571428571428571, 30, 35)
('non_paraphrases', 0.6857142857142857, 24, 35)
('addition_deletion', 0.8876903553299492, 1399, 1576)
('lex_synt_ana', 0.9372197309417041, 627, 669)
('order', 0.90625, 522, 576)
('mor_inflectional', 0.905511811023622, 230, 254)
('syn_coordination', 0.819047619047619, 172, 210)
('mor_derivational', 0.9425287356321839, 246, 261)
('syn_diathesis', 0.9538461538461539, 124, 130)
```



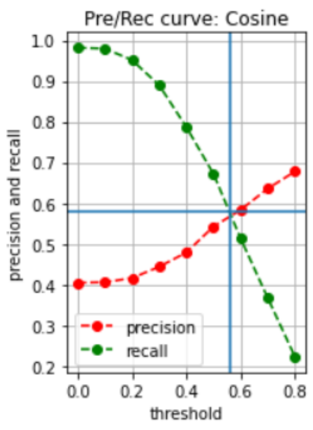
```
('mor_modal_verb', 0.8879310344827587, 103, 116)
('lex_converse', 0.8787878787878788, 29, 33)
('dis_sent_modality', 0.9142857142857143, 32, 35)
('addition_deletion', 0.8895939086294417, 1402, 1576)
('lex_same_polarity', 0.9264444882666141, 4698, 5071)
('syn_ellipsis', 1.0, 87, 87)
('syn_negation', 0.9393939393939394, 31, 33)
('lex_synt_ana', 0.9282511210762332, 621, 669)
('semantic', 0.8294117647058824, 282, 340)
('order', 0.9045138888888888, 521, 576)
('syn_dis_structure', 0.9456869009584664, 296, 313)
('non_paraphrases', 0.6571428571428571, 23, 35)
('syn_coordination', 0.8571428571428571, 180, 210)
('dis_direct_indirect', 0.6944444444444444, 25, 36)
('syn_diathesis', 0.9153846153846154, 119, 130)
('syn_subord_nesting', 0.9078726968174204, 542, 597)
('dis_punct_format', 0.8494423791821561, 457, 538)
('identical', 0.9306930693069307, 94, 101)
('lex_opposite_polarity', 0.8461538461538461, 55, 65)
('mor_derivational', 0.9348659003831418, 244, 261)
('lex_spelling_and_format', 0.8672768878718535, 379, 437)
('mor_inflectional', 0.9291338582677166, 236, 254)
```



```
('dis_direct_indirect', 0.5555555555555556, 20, 36)
('semantic', 0.7617647058823529, 259, 340)
('syn_negation', 0.8484848484848485, 28, 33)
('syn_subord_nesting', 0.9095477386934674, 543, 597)
('dis_sent_modality', 0.8285714285714286, 29, 35)
('non_paraphrases', 0.6285714285714286, 22, 35)
('lex_synt_ana', 0.9192825112107623, 615, 669)
('mor_inflectional', 0.9251968503937008, 235, 254)
('syn_diathesis', 0.8692307692307693, 113, 130)
('lex_same_polarity', 0.8844409386708736, 4485, 5071)
('lex_spelling_and_format', 0.9084668192219679, 397, 437)
('dis_punct_format', 0.8420074349442379, 453, 538)
('lex_converse', 0.9090909090909091, 30, 33)
('mor_modal_verb', 0.8620689655172413, 100, 116)
('mor_derivational', 0.9348659003831418, 244, 261)
('lex_opposite_polarity', 0.8153846153846154, 53, 65)
('identical', 0.9207920792079208, 93, 101)
('syn_ellipsis', 0.9540229885057471, 83, 87)
('syn_dis_structure', 0.9009584664536742, 282, 313)
('order', 0.8940972222222222, 515, 576)
('syn_coordination', 0.8333333333333334, 175, 210)
('addition_deletion', 0.8794416243654822, 1386, 1576)
```

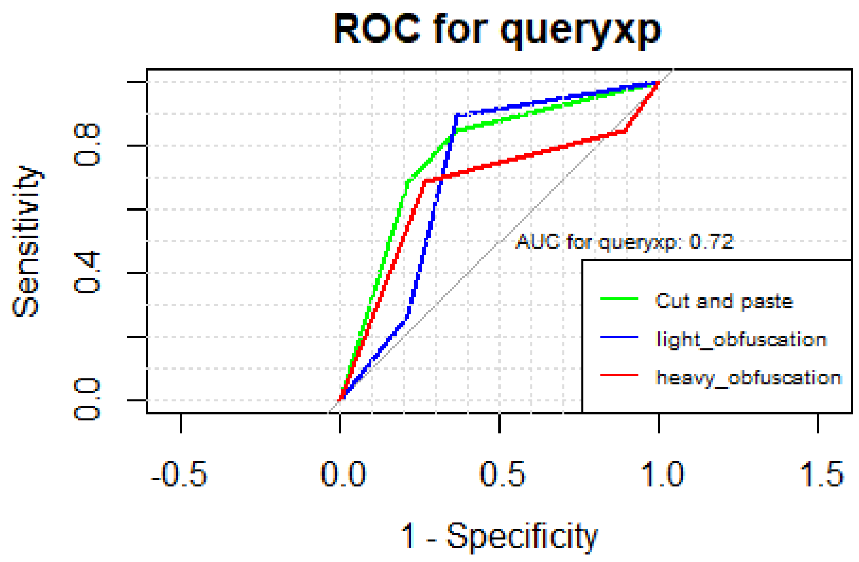


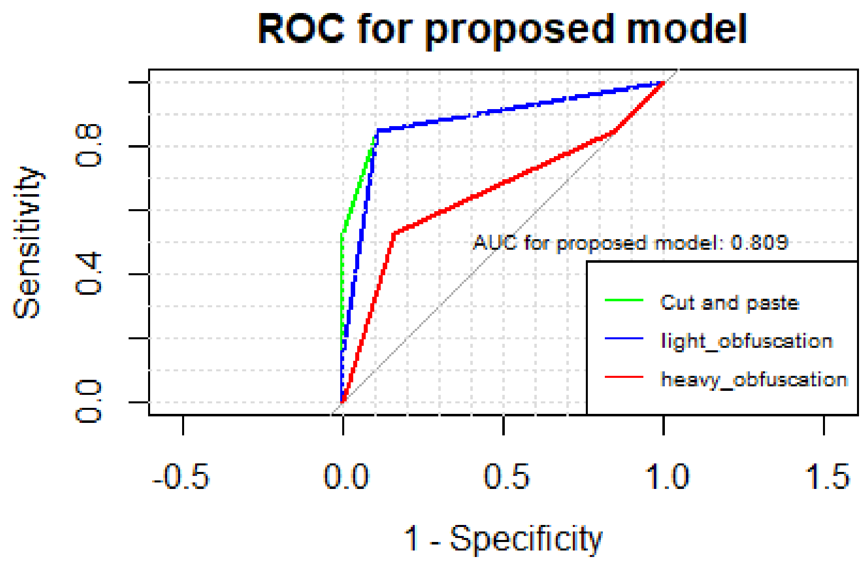
```
('syn_subord_nesting', 0.7839195979899497, 468, 597)
('lex_converse', 0.6666666666666666, 22, 33)
('mor_inflectional', 0.7795275590551181, 198, 254)
('semantic', 0.538235294117647, 183, 340)
('lex_same_polarity', 0.7722342733188721, 3916, 5071)
('syn_negation', 0.7878787878787878, 26, 33)
('syn_diathesis', 0.7692307692307693, 100, 130)
('dis_direct_indirect', 0.2777777777777778, 10, 36)
('mor_derivational', 0.7586206896551724, 198, 261)
('lex_synt_ana', 0.828101644245142, 554, 669)
('order', 0.8350694444444444, 481, 576)
('lex_spelling_and_format', 0.7459954233409611, 326, 437)
('syn_coordination', 0.6714285714285714, 141, 210)
('identical', 0.9603960396039604, 97, 101)
('dis_punct_format', 0.7546468401486989, 406, 538)
('mor_modal_verb', 0.646551724137931, 75, 116)
('syn_dis_structure', 0.7380191693290735, 231, 313)
('syn_ellipsis', 0.9195402298850575, 80, 87)
('non_paraphrases', 0.6285714285714286, 22, 35)
('addition_deletion', 0.7265228426395939, 1145, 1576)
('dis_sent_modality', 0.7142857142857143, 25, 35)
('lex_opposite_polarity', 0.6615384615384615, 43, 65)
```

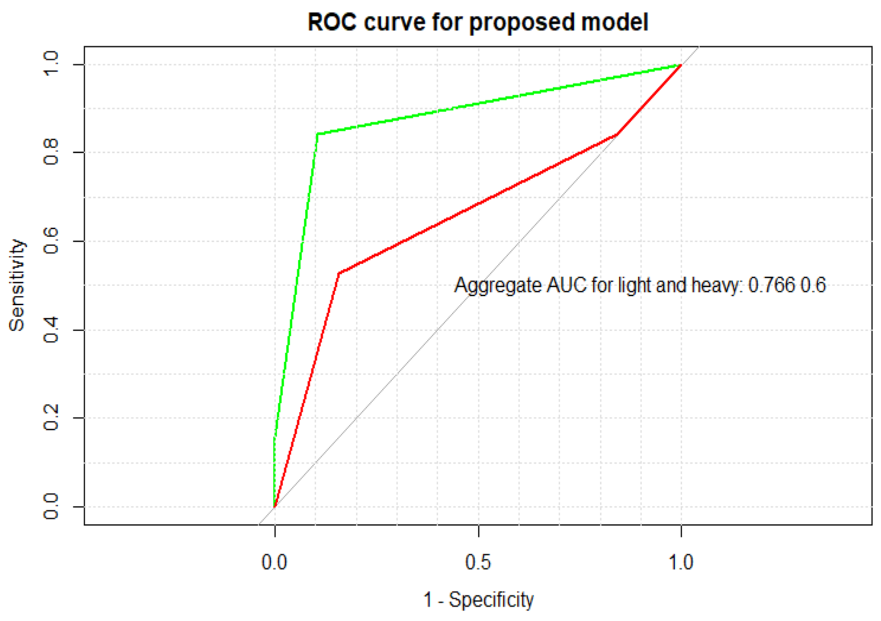


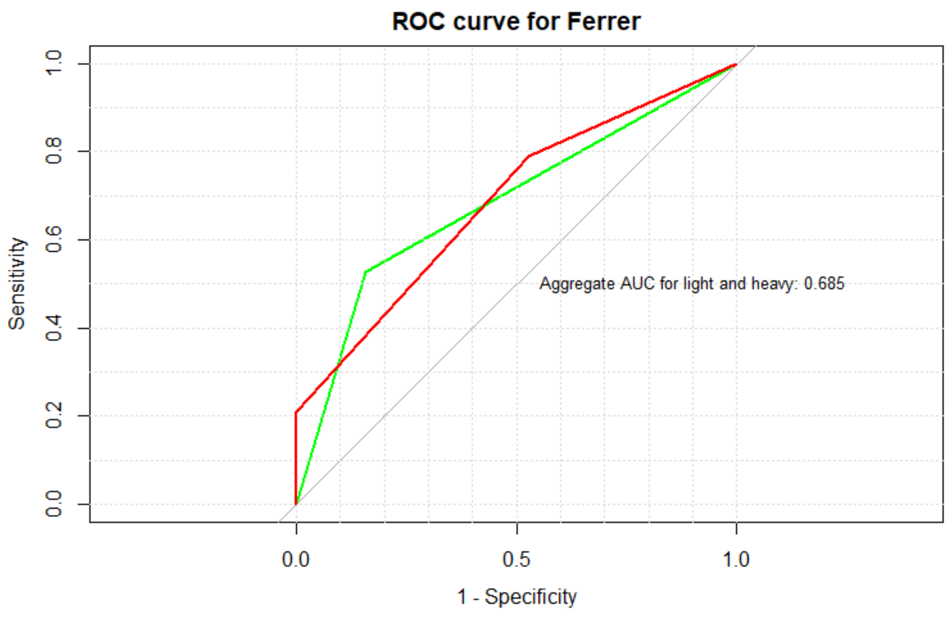
```
('syn_dis_structure', 0.65814696485623, 206, 313)
('lex_same_polarity', 0.6170380595543286, 3129, 5071)
('mor_modal_verb', 0.6120689655172413, 71, 116)
('order', 0.7673611111111112, 442, 576)
('lex_spelling_and_format', 0.7345537757437071, 321, 437)
('syn_negation', 0.5757575757575758, 19, 33)
('syn_subord_nesting', 0.6633165829145728, 396, 597)
('lex_opposite_polarity', 0.6615384615384615, 43, 65)
('identical', 0.9603960396039604, 97, 101)
('syn_coordination', 0.5952380952380952, 125, 210)
('lex_converse', 0.6060606060606061, 20, 33)
('syn_diathesis', 0.6615384615384615, 86, 130)
('dis_direct_indirect', 0.3611111111111111, 13, 36)
('semantic', 0.45294117647058824, 154, 340)
('mor_inflectional', 0.6299212598425197, 160, 254)
('mor_derivational', 0.5823754789272031, 152, 261)
('lex_synt_ana', 0.680119581464873, 455, 669)
('dis_punct_format', 0.6970260223048327, 375, 538)
('syn_ellipsis', 0.7816091954022989, 68, 87)
('addition_deletion', 0.618020304568528, 974, 1576)
('dis_sent_modality', 0.5714285714285714, 20, 35)
('non_paraphrases', 0.6285714285714286, 22, 35)
```

Aggregate AU-ROC Curves For Proposed Model (Best Performing Surface Combination) Baselines From The Experiments To Determine The Best Performing Combination Of Surface Tools

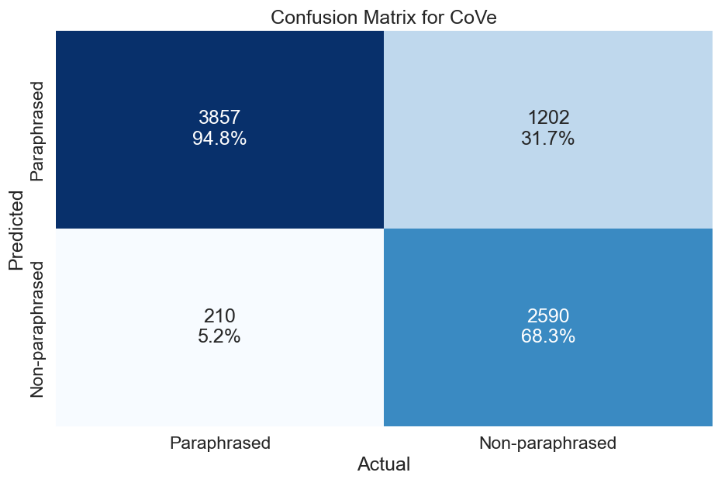




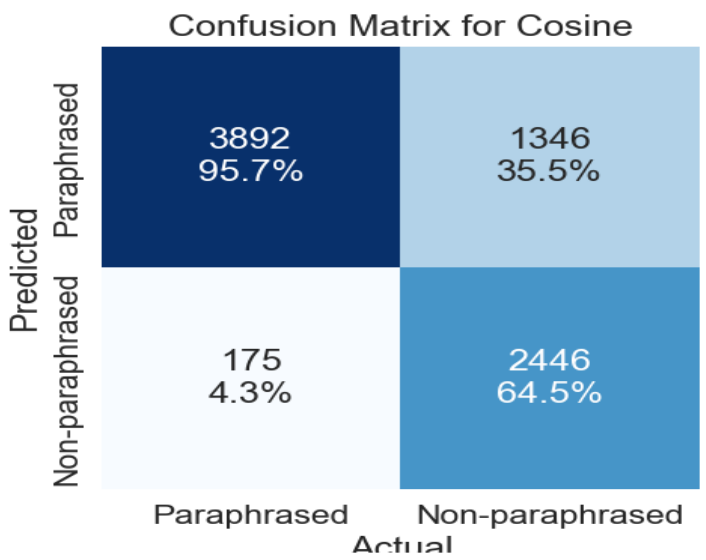

ROC curve for proposed model
Sensitivity
1 - Specificity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
Aggregate AUC for light and heavy: 0.766 0.6

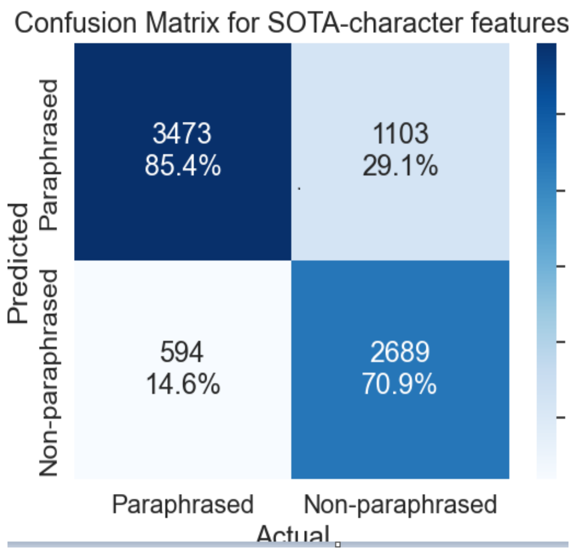

ROC curve for Ferrer
Sensitivity
1 - Specificity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
Aggregate AUC for light and heavy: 0.685

Confusion Matrices for Baselines on Crowdsourcing paraphrase Corpus (Experiments on Paraphrase Plagiarism Detection)

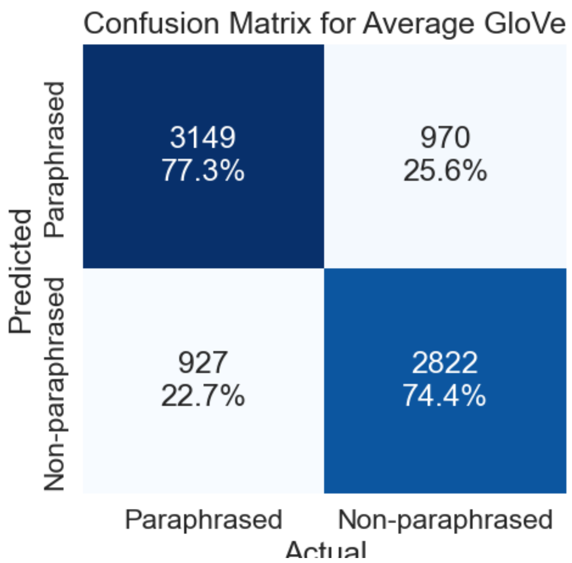


Confusion matrix for the CoVe model





Confusion Matrix for SOTA-character features
Predicted
Paraphrased
Non-paraphrased
3473
85.4%
1103
29.1%
594
14.6%
2689
70.9%
Paraphrased
Non-paraphrased
Actual



Confusion Matrix for Average GloVe
Predicted
Paraphrased
Non-paraphrased
3149
77.3%
970
25.6%
927
22.7%
2822
74.4%
Paraphrased
Non-paraphrased
Actual

Appendix B: Algorithms and Pseudocode

Algorithm for Threshold Determination

START

- For a given class of plagiarism, compute threshold range for all document pairs (suspect and source) that belongs to the class as follows;
- Move a sliding window with shifting upper and lower boundaries (threshold range) on the outputs (similarity scores) from document comparison (between 0 and 1), at an interval of 0.001.
- At each interval point, compute performance for all documents whose similarity scores fall within the sliding window (in precision, recall and f-measure).
- The threshold range at the point where performance is highest is taken as the threshold hold range for the class of plagiarism.
- Repeat above steps for all the classes.

END

Pseudocode for Text Alignment

```
Class text_alignment( ):
    /for a pair of plagiarised document and its source:
        Pre-process pair/
    /find overlapping terms in pre-processed pair/
    /for plagiarised doc:
        join neighbouring overlaps within certain distance into maximum overlapping sequence;
        if overlap < than length (x):
            remove overlap;
        else:  join maximum sequence overlaps within certain distance into one passage/
    /Repeat the above process on the source document/
    /If overlap not occurring in both suspect and source passage:
        Remove overlap;
    If passage size < than length(x):
        remove plagiarised passage and its sources /
    RETRUN plagiarised passages and their sources in  pairs/.
```

Pseudocode for Creating TFIDF

```
Class TFIDF( ):
    /for doc in a list of candidate source doc 1 to n:
        pre-process doc/
    /join all pre-processed document into a single dictionary with aggregated term frequency; call document frequency/
    /for doc in pre-processed candidate doc-list:
        for term  in doc:
            compute;[ (term/len-doc ) * (log(len(doc-list)/doc frequency)]
        append {output of compute to a dictionary}/
    /append [dictionary to a list]/
    RETURN [list of document vectors with TFIDF weighting]/.
```

#Pseudo-Code for Document Pre-Processing

```
Class pre-process():
    Open and read doc using ‘utf8’ encoding/
    /Convert doc to lowercase and tokenize to single words/
    /Remove stop-words from doc and stem each word to root form/
    /Transform doc to n-grams /
    /Transform doc to dictionary that contains n-grams and their weights i.e. frequency/.
    RETURN document vector as a dictionary of weighted n-grams/.
```

Pseudo-code for Candidate Selection

```
Class candidate_selection():
    /tokenise suspect doc into BOW/
    For term in suspect doc (BOW):
        query inverted_index_table with term
        retrieve list of candidate_doc IDs
        append each ID to a list in an array of  lists
          Sum [[each array list],[ ],[ ]…[ ]]
    Return [a list of 50 top ranking IDs as candidate set]
```

Algorithm Used To Combine Surface Tools for Plagiarism Detection

- START
- Pre-process text by applying case normalization, tokenization, white space removal and stemming.
- Transform preprocessed document into n-gram representation model;
- Vectorize ngram model by assigning weights to terms (n-grams) using a term weighting method (e.g. TF);
- Compare vectors for similarity using a similarity measure (i.e. cosine measure) and compare similarity score with threshold derived from corpus;
- Return text as potential plagiarized if similarity score >=threshold
- END

Algorithm for Building the Inverted Index Table

- START
- Pre-process source documents collection using methods described earlier.
- Collect the terms in the entire collection into a single list and convert the list into a dictionary that contains each term and its document frequency.
- Select terms with document frequency <=a specific threshold.
- Create a table in a database (inverted index table) with the following three columns; Terms, Doc_ID and Term frequency
- Load the term column of the inverted index table with the selected terms.
- For each term loaded in the inverted index table, collect all document IDs that contain the term and load them in the Doc_ID column.
- Load the term frequency of each term in the term frequency column of the inverted index table.
- Return inverted index table.
- END