Plagiarism Detection using Text Mining

A Project Report

submitted by

APPINI SURYA TEJA

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.
May 2014

THESIS CERTIFICATE

This is to certify that the thesis entitled **Plagiarism Detection using Text Mining**, submitted by **Appini Surya Teja**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Sutanu Chakraborti
Project Guide
Assistant Professor
Department of Computer Science and Engineering
IIT Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

ABSTRACT

KEYWORDS: Plagiarism, Plagiarism Detection, External Plagiarism, Natural Language Processing, WordNet, Wikipedia, Explicit Semantic Analysis, Fingerprinting, Winnowing

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ESA	Explicit Semantic Analysis
LCS	Longest Common Subsequence
NLP	Natural Language Processing
NLTK	Natural Language Took Kit
TF-IDF	Term Frequency - Inverse Document Frequency
WSD	Word Sense Disambiguation

NOTATION

CHAPTER 1

INTRODUCTION

Plagiarism is an act of using content or ideas of other people and presenting them as one's own work without proper acknowledgement of the original source. It is generally prominent in fields like academics, scientific research, journalism, etc.

Plagiarism detection is an important task for maintaining academic integrity and encouraging ethical research among students. Automatic plagiarism detection has become an increasingly popular task in the present scenario. The detection task involves the extraction of significant features from the suspected work and the probable sources in order to compare and establish a measure of similarity between them. In general, these features could be either lexical, semantic, syntactic or any combinations of these features. However, at the end of the detection process, human intervention is required to judge the suspicious cases and make the final decision regarding plagiarism.

Although many methods have been developed over the years for identifying intrinsic and extrinsic plagiarism, most of them are limited to surface level which attempt to compare the source and suspicious texts based on the overlap of strings or sequence of words. Most of the online plagiarism detection tools like Turnitin use these strategies (Maurer *et al.* [2006]). Little effort has been made to consider the semantics of words and the application of Natural Language Processing techniques in plagiarism detection (Chong *et al.* [2010]).

1.1 Chapter Overview

This chapter provides an introduction to the the process of plagiarism detection. The different types or levels of plagiarism are explained in Section 1.2. Section 1.3 provides an overview of the two classes of plagiarism detection techniques which are used to identify the various levels of plagiarism. Section 1.4 describes the motivation behind choosing this problem and Section 1.5 contains the description of the problem statement. The main contributions of this work are discussed in Section 1.6. Finally, the organisation of the thesis is summarized in Section 1.7.

1.2 Types of Plagiarism

There are different types of plagiarism depending on the complexity involved in the detection process. The basic form of plagiarism is verbatim copy which involves copying word for word without any effort to make changes with respect to the source(s). The next level includes changes made within a sentence. Changing the order of words and deletion or addition of certain words/phrases are some form of changes that belong to this category. Paraphrasing is the next higher form of plagiarism. Paraphrasing means to restate the meaning of a sentence or a piece of text using alternate words which are generally synonyms or words related to the actual words present in the text. Other forms of plagiarism at this level can include syntactic changes and changes pertaining to the form and structure of sentences. Changing active voice to passive voice is an example of such an instance. The highest level of plagiarism is to copy ideas or intent of others and present them as one's own work in his own style. This is generally prominent in academic research.

Text summarization can also be considered as a form of plagiarism in the absence of proper references.

Style plagiarism is another form of plagiarism in which an author may not copy the exact content but tends to follow the same style of writing from some external source. By style, we refer to a model based on statistical features like the average length of a sentence, number of sentences per paragraph and other such features. These features often provide significant information in cases where there is a well established correspondence between the sections and subsections in the source and the plagiarised works.

Cross-lingual plagiarism is also a form of plagiarism which is usually observed in texts that are translation of some original text in a different language. Online translation tools have also enabled to plagiarise texts in the same language by translating them into a different language and then back to the original language.

< insert image: different types of plagiarism >

1.3 Detection Techniques

In general, there are two classes of plagiarism detection methods which identify the different types of plagiarism as explained in the previous section. In this section, we discuss the intrinsic plagiarism detection methods in Subsection 1.3.1 and then extrinsic plagiarism detection methods in Subsection 1.3.2.

1.3.1 Intrinsic Plagiarism Detection

Intrinsic plagiarism detection is studied under the area *stylometry* or *authorship attribution*. These methods are used to detect plagiarism in the absence of a reference corpus containing the probable source documents.

In these methods, the writing style of the author is generally modelled using character n-gram profiles (Stamatatos [2009]), typically trigrams (Kestemont *et al.* [2011]). The variation in style is quantified by a style change function using a suitable dissimilarity measure. In intrinsic plagiarism detection, it is assumed that the original author has done atleast some part of the work. In case of failure of the assumption, the detection process would fail as there will be no deviation observed. This problem is known as the *Ghost Writer* problem.

1.3.2 Extrinsic Plagiarism Detection

Extrinsic plagiarism detection methods are used to detect plagiarism cases given a reference corpus containing the set of possible source documents. Most of the research related to plagiarism detection is focused in this area. The extrinsic plagiarism detection process involves various stages as shown in the figure below.

insert image: stages of plagiarism detection

Initially, the set of candidate documents are filtered out from the given corpus using methods based on fingerprinting (Manber *et al.* [1994]) and n-gram matching. A further detailed analysis is performed in order to establish similarities between the different pairs of documents. Finally, each pair is labelled as either plagiarised or non-plagiarised using an appropriate model which can perform the classification task. The classification could be binary classification or multi-class classification depending on the user's requirement. In general, binary classification can be successfully performed by learning a proper threshold for similarity. However, multi-classification is more challenging and it requires deeper Natural Language Processing techniques to distinguish the various levels of plagiarism.

In this work, we restrict ourselves to extrinsic plagiarism detection of text documents written in English language.

1.4 Motivation

With the increasing resources in web and increasing usage of electronic media, it is practically impossible for humans to manually test and detect unfair or unethical practices involving plagiarism. Currently we have some popular and commercial online plagiarism detection tools like Turnitin and Ferret. They help us in detecting plagiarism at a large scale with a high speed and a detailed summary or report of their analysis. These tools are well supported by large databases containing millions of webpages and academic records which are constantly being updated with the help of a web crawler.

The performance of these online plagiarism detection tools has not been satisfactory despite having such huge databases. Although they have been successful in identifying cases containing content copied directly from a variety of resources, they often fail in detecting cases involving higher forms of plagiarism. These tools detect cases which have exact matches of keywords, phrases or sequence of words between the source and suspicious document. In general academic settings, we are often required to compare each student's work with a reference corpus consisting of submissions of other students and the reference materials provided to them as part of courses. This is essentially the problem of external plagiarism detection. Students tend to plagiarise work by obfuscating it at different levels.

Direct string matching algorithms (Su *et al.* [2008]) which match exact sequence of words fail in detecting cases involving paraphrasing, addition or removal of words and other similar forms. Hence, we aim at building a tool using NLP concepts with text mining techniques to overcome these drawbacks.

1.5 Problem Statement

Our major objective is to build a robust tool for extrinsic plagiarism detection of text documents in English using text mining techniques and the semantics of the constituent words and phrases. We aim at a high recall centric system which can perform the binary classification task with a decent precision and hence high accuracy (f-measure).

The various parameters involved in the detection algorithm are abstracted out to give the users the flexibility to tune them according to their requirements. We also perform multi-class classification on the dataset (Clough and Stevenson [2011]) chosen and evaluate the validity of our approach on the basis of the accuracy values attained. We also analyse the effect of various parameter values on the performance.

1.6 Contributions of this work

The contributions of this work can be summarized in four points.

1. We integrate the information present in WordNet and Wikipedia knowledge sources with a modified k-gram matching strategy in order to compute similarities between document pairs. This helps in detecting cases which involve paraphrasing and changes in word and sentence order. Similarities between various topics and phrases which are not in the scope of NLTK are computed using the Explicit Semantic Analysis method (Gabrilovich and Markovitch [2007]). These provide support in matching content requiring background knowledge of the subject.
2. We provide the flexibility to focus detection on various levels of plagiarism by appropriately tuning the parameters involved and choosing the right threshold.
3. We implement the winnowing algorithm in plagiarism detection and prove its importance and efficiency in the process of detection.
4. We also compare the performance of our proposed approach with the two baseline detection systems - Ferret Detector (Lyon *et al.* [2004]) and the model developed by Chong *et al.* [2010].

1.7 Organisation of the thesis

Chapter 2 provides the background knowledge required to understand the various concepts used in the project. It also contains the details and analysis of the related work in the past and the current state-of-the-art. The background knowledge includes basic concepts in NLP and explanation of some methods used in text mining. In Chapter 3, we explain the methodology followed and provide detailed description of each step with relevant examples. In Chapter 4, we present the results and compare them with the baseline systems. We briefly explain the reasons for our observations and improvement achieved. Finally, in Chapter 5, we summarize our work and provide directions for some possible work in the future.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Chapter Overview

This chapter provides the required background knowledge to understand the various concepts and techniques used in different parts of the system. In Section 2.2 we present the overview of concepts used and then explain each of them with details in the subsections that follow. Concepts like text mining, fingerprinting, hashing, winnowing, ESA, clustering, etc. are explained with relevant examples. In section 2.3, we discuss the details of relevant work that had been done over the past along with some personal insights about the advantages and disadvantage of each approach. Finally, we conclude the chapter by stating how our approach is likely to perform better than the existing methods in certain aspects.

2.2 Background Knowledge

This project depends on concepts belonging to natural language processing and text mining. Some of the NLP concepts used include parsing, lemmatization, word sense disambiguation (WSD) and explicit semantic analysis (ESA). Post pre-processing, we apply text mining methods like fingerprinting to perform detailed analysis. We also experiment with winnowing, a fingerprinting algorithm used for efficient storage and analysis of preprocessed documents. We use the idea of clustering to group words related to each other in the taxonomy. In the following section, we briefly explain each of these concepts with relevant examples.

2.2.1 Text Mining

Text mining refers to the process of analysing data or text in natural language by extracting quality information from the text. It involves the process of conversion of unstructured input into structured data, deriving meaningful patterns from the structured data and interpreting them to obtain the output. General applications of text mining include text categorization, sentiment analysis, etc.

2.2.2 Fingerprinting

Document fingerprinting is essentially an algorithm to represent a document with a sequence of hashes for unique identification and comparison purposes. This helps in avoiding storage and comparison of large pieces of text. Fingerprinting is a widely used method for extrinsic plagiarism detection. In the detection process, the fingerprint of a suspicious document is compared against all the precomputed fingerprints of the documents in the given reference corpus.

The following subsections discuss the specific hashing algorithm used in this work and the concept of winnowing.

2.2.3 Rabin-Karp Hash Function

Karp and Rabin [1987] have proposed a rolling hash function which is used in computing the hash value of a sequence of elements (in general, character k-grams) for efficient string matching. The function enables us to calculate the hash value of $(i+1)^{st}$ k-gram from the hash value of the i^{th} k-gram. We represent each k-gram similar to a number in some base b. Let $w_1 w_2 .... w_k$ be a k-gram of words where each w_i represents the unique identity of a word. The hash value of the k-gram $H(w_1 w_2 .... w_k)$ is defined as follows:

$$H(w_1 w_2 .... w_k) = w_1 * b^{k-1} + w_2 * b^{k-2} + ... + w_{k-1} * b + w_k$$

The hash value of the next overlapping or $(i+1)^{st}$ k-gram is computed using the following identity.

$$H(w_2 w_3 .... w_{k+1}) = (H(w_1 w_2 .... w_k) - w_1 * b^{k-1}) * b + w_{k+1}$$

The base b is generally chosen to be a prime number to ensure uniform distribution of hash values and to prevent collisions among them.

2.2.4 Winnowing

Winnowing is an algorithm for selecting particular fingerprints from a sequence of hashes of k-grams representing a document (Schleimer *et al.* [2003]). Any fingerprinting algorithm must satisfy two important properties for matching substrings or k-grams between documents.

First, it should provide a *guarantee threshold, t,* such that any matching kgram sequence of length atleast t would necessarily contain a matching hash in fingerprints of both the documents. Second, it should also provide a *noise threshold, k,* which is the minimum length of the matching sequence to be detected. Any matching sequence of length less than k can be considered as noise.

The larger the value of k is, the lesser is the probability that matches between documents are coincidental. However, if the value of k is very large, the algorithm might not detect the reordering of substrings or k-grams of length less than k. This will affect the computation of similarity as we fail to detect some candidate matches. Hence, the value of k chosen is critical and we need to select an ideal minimum value to avoid coincidental matches.

In winnowing, we choose the minimum hash in every window of size w which contains w kgrams. If there is more than one hash with the same value as the minimum, we choose the rightmost occurrence. These selected set of hashes form the fingerprint of the document. For a selected noise threshold, k, if we desire a *guarantee threshold*, t, then we need to select the value of the window size $w = t - k + 1$. Any kgram sequence of length t will have $t - k + 1$ overlapping k-grams present in it. By selecting a window size of w as defined earlier, we ensure that at least one matching hash is present in some window corresponding to both the documents.

The expected length of the document after winnowing is equal to $\frac{2}{w+1}$ times its original length. This can be intuitively understood as there are only two cases out of the $(w + 1)$ possible cases that can contribute a new hash to the fingerprint while considering a new window. The two possible cases are:

- The previous minimum hash value chosen falls outside the new window.

- The new element added to the new window becomes the new minimum.

In all the other cases, the previous minimum is same as the current minimum and hence, does not contribute to the fingerprint. An example illustrating the winnowing algorithm ($k = 3$, $w = 4$) on preprocessed text is provided below.

- Text: "`inheritance basic concept object orient programming model reuse exist class code new class kind relationship`"
- Unique word identities: 2 9 5 6 1 11 13 12 17 4 3 10 7 8 14
- k-gram hash values: 98 36 105 503 83 84 38 37 125 25 143 93 32
- Fingerprint after winnowing: 36 83 38 37 25

2.2.5 Explicit Semantic Analysis

Semantic relatedness, also known as semantic similarity, is a similarity metric between two terms or documents based on their meaning and their information content. Generally, semantic relatedness between terms is estimated based on their positions in a taxonomy and using path based similarity measures.

Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch [2007]) is a method used to represent the meaning of text as a weighted vector of concepts in a high dimensional space of concepts derived from a corpus (in general, Wikipedia). Semantic relatedness between two texts is computed using standard similarity metrics (eg. cosine similarity) used for comparing vectors. In ESA, the column vectors of the tf-idf matrix represent the words. Each document is represented as the centroid of the its word vectors.

WordNet based similarities are also computed using concepts derived from some reference corpus. However, information in WordNet is limited to individual words which belong to the general taxonomy and it does not help in comparing longer pieces of texts. On the other hand, ESA is helpful in computing similarities between texts of any length which require domain-specific knowledge. Wikipedia is preferred as it is the largest knowledge repository available on the web with human defined natural concepts which are easy to understand.

< insert image: esa examples >

2.2.6 WordNet Based Semantic Similarities

The Natural Language Tool Kit (NLTK) is basically a set of libraries for statistical natural language processing in Python language. It provides interfaces to large corpora as well as lexical resources like WordNet. NLTK also has implementations of several algorithms for measuring semantic relatedness between words (Mihalcea *et al.* [2006] , Budanitsky and Hirst [2006]). Some of the examples are Wu-Palmer similarity (Wu and Palmer [1994]), Resnik similarity (Resnik [1995]), Leacock Chodorow similarity (Leacock and Chodorow [1998]), etc.

In our experiments, we use the Wu-Palmer similarity for measuring semantic similarity between words. The Wu-Palmer measure computes similarity by considering the depth of the two synset nodes in the WordNet taxonomy and the depth of their least common subsumer (LCS) node. The Wu-Palmer similarity between two synsets s1 and s2 is defined as in 2.1. Its range is (0,1].

$$wup\_similarity = \frac{2 * \text{depth(LCS)}}{\text{depth(s1)} + \text{depth(s2)}} \tag{2.1}$$

2.2.7 POS tagging and Word Sense Disambiguation

In NLP, POS tagging of words is the problem of assigning part of speech to a given word based on its definition and the context in which it occurs. On the other hand, WSD is the problem of identifying the correct sense of the given word based on its context. Both the problems are closely related to each other as they depend on each other. However, WSD is a more challenging task than POS tagging because the part of speech generally depends only on few words adjacent to the given word. But, the sense of a word might depend on words which are farther away.

POS tagging methods are generally statistics based or rule based. WSD is generally performed using knowledge-based methods which use resources like dictionary or knowledge bases like WordNet.

2.2.8 Agglomerative Hierarchical Clustering

Hierarchical clustering is a method of analysing elements by forming a hierarchy of clusters. Agglomerative hierarchical clustering is a bottom up approach to build large clusters incrementally. Initially, each element is considered as an individual cluster by itself. In each step, two closest clusters are merged together and this is repeated until all of the intermediate clusters are merged into a single large cluster. Agglomerative clustering allows us to terminate clustering after reaching the desired number of clusters. The proximity values between the clusters are calculated using the pairwise similarity of the elements in the cluster. There are several strategies to merge the intermediate small clusters. For example: MIN, MAX, Group average, etc. (Tan *et al.* [2014]). In our approach, we follow the MIN strategy to merge clusters repeatedly. In MIN strategy, we select the two closest points belonging to two different clusters and merge those clusters.

< insert image : agglomerative hierarchical clustering example >

2.3 Related Work

Plagiarism detection has been an important concern in many academic institutions and other fields. Significant amount of research has gone into this problem over the time to detect plagiarism of various forms and levels. Lukashenko *et al.* [2007] presented an overview of the various computer based methods used for plagiarism detection and Meuschke and Gipp [2013] discussed the state-of-the-art methods in this area. Here, we present details and personal insights about some related work which were focused on extrinsic plagiarism detection of text documents.

One of the earliest attempts in plagiarism detection was made by Lyon *et al.* [2004]. Their implementation of the Ferret electronic plagiarism detection system was based on identifying short strings that were matching in both the documents. They used trigram matching in their systems driven by statistical observations which showed that the number of distinct trigrams in any text is generally higher than the corresponding unigrams or bigrams and the number of matching trigams between two different documents is significantly low. Although it has been successful in identifying simple plagiarism cases and can serve as a strong baseline system, it cannot detect higher forms of plagiarism involving paraphrasing or syntactic changes.

Nahnsen *et al.* [2005] developed a system to detect pairs of chapters in books that were translations of the same original chapter. Their system was based on n-grams of lexical chains as the main feature and used similarity measures such as cosine similarity of vectors of lexical chain n-grams and vectors of tf-idf weighted keywords. Lexical chains represent list of words or items that are conceptually related to each other through relations like hyponymy, synonym, etc. The lexical chains were computed using the algorithm proposed by Silber and McCoy [2002]. Their results indicate that unordered lexical chain n-grams of length more than four are effective in identifying similar content. However, their approach was limited to nouns, verbs, and adjectives that were present in WordNet. According to the algorithm, they consider only the contribution of relations between words separated by less than or equal to six sentences. Moreover, they do not consider the order of n-grams and could also fail to detect matching keywords if the n-grams containing them do not match exactly. Also, their approach does not make distinction between keywords and stopwords as it lacks proper preprocessing techniques.

Clough and Stevenson [2009] designed a corpus containing different types of plagiarism. The suspicious documents belong to various levels of plagiarism ranging from verbatim plagiarism to non-plagiarised documents. Detailed explanation about the corpus can be found in Chapter **??** as we use the same dataset for our experiments. The authors used n-gram overlap and longest common subsequence to classify the suspicious documents into different types of plagiarism (Clough and Stevenson [2011]). They performed classification task using both the individual features (different sized n-grams) and combination of these features. They observed highest accuracy for bigrams and trigrams when used independently as individual features. They achieved an accuracy of 94.7% in the binary classification using Naive Bayes classifier with 10-fold cross validation. However, they did not consider the meaning of words or phrases while comparing n-grams. Also, there was no attempt made to consider the re-ordering of words within those k-grams.

Chong *et al.* [2010] demonstrated the importance of preprocessing and NLP based techniques in detecting plagiarism. They performed their experiments on the corpus created by Clough and Stevenson [2009]. Techniques like sentence segmentation, stop-word removal, stemming, part-of-speech tagging, etc. were used as a part of preprocessing. Further, they used comparison methodologies like trigram similarity, language model probability (perplexity), longest common subsequence and dependency relations matching to classify the documents into various categories. Their work showed significant improvements in performance compared to the state-of-the-art methods at that time. They achieved an accuracy of 70.53% as compared to 66.32% obtained using Ferret baseline Lyon *et al.* [2004]. However, as they do not consider the semantics, it is possible to fail even in simple cases of paraphrasing using synonyms.

Chong and Specia [2011] further extended their work by going a step deeper. In addition to all the preprocessing and NLP techniques applied in their previous work, they generalised words using their WordNet synsets for groups of synonym words. They performed experiments on a subset of PAN'10 corpus and proved that their approach yielded better results when compared to the standard n-gram techniques. They used the overlap coefficient as defined by (Clough and Stevenson [2011]). Results showed a significant improvement in recall with almost similar precision as before. However, their approach is limited to only synonyms of words which are present in the WordNet and do not capture higher order relations between words.

Later, Adeel Nawab *et al.* [2012] proposed a strategy to detect text reuse by creating modified and weighted n-grams from the original n-grams. They performed their experiments on the METER corpus Gaizauskas *et al.* [2001] and reported improvement in performance compared to the existing approaches at that time. In their approach, they create new n-grams from original ones by deleting a single word in an n-gram or by substituting synonyms for all possible senses of a word in the original n-gram. However they do not consider the actual meaning of words due to which it might not be possible to handle complex paraphrasing cases. Ignoring the order of words while matching and using only n-grams could result in losing important matches of length less than n. Such evidences are crucial in differentiating the different levels of plagiarism among the detected cases.

Recently, Kumar [2014] has proposed a graph based plagiarism detection technique to handle word reordering and paraphrasing in text. He experimented with the PAN'12 corpus and the corpus created by Clough and Stevenson (2009) and showed that his approach performed better than the state-of-the-art methods. The idea is to create a semantic network of terms as nodes by using a sliding window of size two for creating links and then using co-occurrence frequency and normalise Pointwise Mutual Information for assigning weights to links. Further, he used the concept of minimum weighted bipartite graph clique cover to identify the plagiarised text patterns in the target and the source texts.

Other related works include approaches based on syntactic information, structures and rules combined with similarity measures using tf-idf of keywords and lexical semantic features. These aim at handling re-ordering of words and related changes. Uzuner *et al.* [2005] and Ram *et al.* [2014] have made significant contributions using such approaches. However, they did not take the semantic relatedness between words into consideration.

In our approach, we consider generalising words or phrases based on their semantic relatedness and also use the LCS to account for the order of matching sequence of words or concepts. Fingerprinting also preserves order as the selection of hashes in each window of k-grams depends on the order in which the k-grams occur. We also integrate ESA which allows us to extend our scope beyond individual words presents in WordNet and relate words which require domain specific knowledge.

CHAPTER 3

METHODOLOGY

3.1 Chapter Overview

This chapter explains our approach and the methodology followed in detail. In Section 3.2, we describe the experimental setup consisting of the description of corpus in subsection 3.2.1. In Section 3.3, we explain each step of the detection process in order. We discuss the three phases of our approach in the corresponding subsections. The three phases are preprocessing, application of NLP techniques and the classification task.

3.2 Experimental Setup

3.2.1 Corpus

In this experiment, we use the corpus created by Clough and Stevenson (2009). The corpus was manually developed using 5 Wikipedia articles as the source or original documents. It contains a total of 95 suspicious documents with different levels of plagiarism. These were written by 19 students as responses to 5 questions, each question based on one of those original articles. The students wrote answers based on the original articles with varying levels of plagiarism according to the authors instructions. The suspicious documents were classified into four classes:

1. Near copy: Text was simply copied and pasted from the original text.
2. Light revision: Copying by substituting words and phrases with synonyms, addition or deletion of some phrases and changes in grammatical structure. Order of the sentences was preserved without any major changes.
3. Heavy revision: Rephrasing and restructuring the sentences of the original text using their own words. This also involved splitting and combining sentences.
4. Non plagiarism: Answers written based on student's own knowledge and learning from sources like text book chapters and notes without access to the original Wikipedia articles.

The suspicious documents were also annotated with factors like the nativity, knowledge of the student and difficulty experienced in answering. Each answer was about 200-300 words in length. There were 38 non-plagiarised answers and 19 answers in each of the other three categories. Although there are other corpus like the METER corpus (Gaizauskas *et al.* [2001]) or the standard PAN corpus, we used this corpus as it is well annotated and easy to analyse the different cases. This corpus reflects a more realistic situation than other corpora where artificial plagiarism is introduced through automation.

3.2.2 Multiclass Classification

The goal is to classify the suspicious documents into the four categories described above. The challenge involves identifying features that help in detecting plagiarism and also in distinguishing between the various levels of plagiarism. More importantly, we desire high accuracy with respect to the binary classification task.

3.3 Detection Process

We follow three stages in the process of plagiarism detection. First, we apply standard pre-processing techniques to remove noise and bring the text into a uniform format for easy analysis. Next, we use WordNet and Wikipedia based semantic similarities to cluster words related to each other. Finally, we use fingerprinting (with winnowing) to represent the document and extract k-grams and longest common subsequence.

3.3.1 Pre-processing

We use the following pre-processing steps:

Sentence splitting: Split the text into sentences as they are required for parsing, lemmatization and word sense disambiguation.

Convert to lowercase: All the characters are converted to lower case for uniform representation and easy comparison.

Punctuations removal: We remove all the extra characters and punctuation marks treating them as noise.

Replacing numbers: We replace all the numbers with a dummy character for uniform comparison.

3.3.2 Application of NLP techniques

We use the Stanford CoreNLP package (version 3.3.1) publicly available at the Stanford NLP Group which provides a set of natural language tools for analysing English text. The package consists of tools like the part-of-speech tagger, the parser, the lemmatizer, etc. We use some of these tools in our next steps of the detection process.

POS tagging: We use the Stanford part-of-speech tagger to assign tags as used in the Penn Treebank Project. These tags are appropriately mapped to the WordNet sense tags for comparing the part-of-speech of the synsets of different senses of the words.

Word sense disambiguation Generally part-of-speech tagging and word sense disambiguation are problems that are very closely related to each other. The current state-of-the-art methods have achieved a very high accuracy of approximately 95% in POS tagging as compared to a relative low accuracy of around 75% in WSD. We use the word sense disambiguation module available in NLTK in Python language. Also, we compare the POS tag of the synset obtained after WSD with the actual POS tag assigned by the POS tagger while considering the similarity of words for clustering.

POS tagging and WSD are required for forming coherent clusters of related words. It is possible that the same word might be highly similar to two unrelated different words with respect to two different senses of the word. This would cause all the three words to fall in the same cluster which is not desired. Hence, we need to find the correct senses of the words while comparing similarities to form clusters.

Lemmatization We use the Stanford Lemmatizer to reduce the words to their basic forms, also known as lemmas. This helps in grouping and matching different inflected forms of the same word.

Tokenization We use the Stanford Parser to parse the original sentences (before converting to lowercase) and extract the various noun phrases present in it. We apply certain heuristics like selecting bigrams and trigrams starting with upper case letters to identify proper nouns, concept names, etc. The remaining words are treated as individual tokens.

At this stage, we remove all the stopwords present in the list of stopwords which is provided as an input file to our system. This is required as the overlap of commonly occurring stopwords is not so significant in identifying plagiarism.

Semantic Similarity After extracting the tokens, we compute the pairwise similarities between all the tokens. For words that are present in the WordNet taxonomy, we use the Wu & Palmer semantic similarity measure implemented in Python as *wup_similarity* in NLTK. Pairwise semantic similarities for words that are not present in WordNet and for phrases extracted using parser are computed using the explicit semantic analysis technique. We use the publicly available implementation of ESA based on the original implementation by Gabrilovich and Markovitch [2007].

Clustering The agglomerative hierarchical clustering technique is applied to form clusters of words and phrases. We set a lower threshold on the similarity based on WordNet or Wikipedia, below which we do not merge the clusters any further. This threshold enables us to control the number of clusters formed and the level of abstraction at which we desire to generalise words or phrases. The pseudo-code of the clustering algorithm used is presented in Algorithm 1. All the tokens belonging to the same cluster are represented by their cluster identity.

Fingerprinting and Winnowing After clustering, we represent the pre-processed documents by replacing their constituent words and phrases with their respective cluster identities. We conduct two sets of experiments in which we enable the option of winnowing in one and disable in the other. Fingerprints are used to extract k-grams and longest common subsequence between the pairs of suspicious document fingerprint and the corresponding source document fingerprint.

3.3.3 Comparison and Classification

This phase involves extracting relevant features from the fingerprint and using a suitable metric to establish similarities between the document pairs.

k-grams and LCS We prefer to use these two features as they complement each other in identifying plagiarism and distinguishing the different levels. We extract all possible k-grams from the fingerprint and sort each k-gram based on the cluster identities of its constituents. This enables us to detect both the cases of re-ordering of words within a sentence as well as re-ordering of sentences within the document as kgrams do not consider the order.

Further, in order to differentiate between the various levels within the plagiarised cases, we compute the longest common subsequence between the fingerprint which serves as an important feature as it considers the order while matching elements and also captures matches of length less than k. LCS is also used to account for addition or deletion of one or more words which can lead to incomplete matches among kgrams.

< **insert image: illustration why lcs is needed** >

Comparison methodology We use the containment measure as defined by Clough and Stevenson [2011] in order to establish similarities for comparing the document pairs. Let A be the suspicious document and B be the source document. S(A) and S(B) represent the set of trigrams in the respective documents.

$$Containment\ C_k(A,B) = \frac{|S(A,k) \cap S(B,k)|}{|S(A,k)|}$$

Containment measure gives the number of matching kgrams normalised by the number of kgrams in the suspicious document which helps in comparing documents with varying lengths and is also suitable for this corpus as the source documents are always larger in size than the corresponding suspicious documents.

We also use the normalised LCS as the other feature. Normalised LCS measure is defined as follows.

$$LCS_{norm} = \frac{LCS(A,B)}{Length(A)}$$

Classification We use the simple Naive Bayes classifier with 10-fold cross validation which is implemented in Weka (Hall *et al.* [2009]) for the classification task using the two features. Simple Naive Bayes is chosen as we consider that both the features as independent. Also, it is not necessarily true that the features have a linear dependency with respect to the classes where we can classify using simple threshold intervals of similarities for the different classes. So, Naive Bayes model is used to build and train the classifier.

Algorithm 1 Detection process algorithm

```
1: Input files: Source document = src_doc, Suspicious document = sus_doc, Stop-
   words file = stop_file
2: Input parameters: n-gram size = $k$, NLTK similarity threshold = $t_n$, ESA
   similarity threshold = $t_e$, winnowing window size = $w$
3: Output: Label - Plagiarism type
4: procedure DETECTION
5:     src_file ← Preprocess(src_doc)
6:     sus_file ← Preprocess(sus_doc)
7:     pos_tagged_src ← POS tagging(src_file)
8:     pos_tagged_sus ← POS tagging(sus_file)
9:     lemmatized_src ← Lemmatization(pos_tagged_src)
10:    lemmatized_sus ← Lemmatization(pos_tagged_sus)
11:    src_tokens ← Parse(lemmatized_src)
12:    sus_tokens ← Parse(lemmatized_sus)
13:    synsets ← WSD(lemmatized_src, lemmatized_sus)
14:                                   ▷ synsets = Hashmap< token, synset >
15:    Remove stopwords(lemmatized_src, lemmatized_sus)
16:    all_tokens ← src_tokens + sus_tokens
17:    Initialise sim_matrix
18:    for every pair (token1, token2) in all_words do
19:        if token1 and token2 present in WordNet then
20:            synset1 ← synsets.get(token1)
21:            synset2 ← synsets.get(token2)
22:            sim_matrix(token1, token2) = synset1.wup_similarity(synset2)
23:        else
24:            sim_matrix(token1, token2) = ESA(token1,token2)
25:    cluster_identities ← do Agglomerative Hierarchical Clustering(sim_matrix)
26:                       ▷ cluster_identities = Hashmap< token, cluster_id >
27:    src_fingerprint ← Fingerprinting(src_tokens, cluster_identities )
28:    sus_fingerprint ← Fingerprinting(sus_tokens, cluster_identities)
29:
30:    $LCS_{norm} \leftarrow \frac{LCS(src\_fingerprint, sus\_fingerprint)}{length(sus\_fingerprint)}$
31:
32:    src_kgrams ← getKgrams(src_fingerprint)
33:    sus_kgrams ← getKgrams(sus_fingerprint)
34:    sort(src_kgrams)
35:    sort(sus_kgrams)
36:
37:    $Containment \leftarrow \frac{|src\_kgrams \cap sus\_kgrams|}{|sus\_kgrams|}$
38:
39:    Build the Naive Bayes Classifier (NBC) model with 10-fold cross validation
40:    label ← NBC(Containment, $LCS_{norm}$)
41:    return label
```

CHAPTER 4

EXPERIMENT RESULTS

4.1 Chapter Overview

In this chapter we present the various results obtained through our experiments on the dataset described in the methodology chapter. We compare our model against the baseline, the Ferret electronic plagiarism detection tool Lyon *et al.* [2004] and also against the model proposed by Chong *et al.* [2010]. We also discuss the effect of variation of various parameters on the performance of the system in terms of precision, recall and accuracy.

4.2 Baseline Models

Ferret uses Jaccard similarity coefficient to measure the similarity between the source and suspicious documents by matching trigrams in both the documents. Let A and B be the two documents and S(A) and S(B) be the set of trigrams respectively.

$$\text{Jaccard Coefficient J(A,B)} = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|}$$

The model proposed by Chong *et al.* [2010] uses the containment measure which is a slight modification of the Jaccard similarity defined in terms of sets. It uses

LCS, NLP techniques like language models and dependency parsing to obtain various features for comparison and classification.

4.3 Evaluation

We evaluate our system in two different ways. First, we perform a binary classification task to label every suspicious document as either plagiarised or not. Later, we also perform a multi-class classification task in which we build a general classifier to classify each document into one of the four levels of plagiarism which were explained in subsection 3.2.1. We use the following evaluation metrics as defined in the PAN plagiarism detection evaluation framework (Potthast *et al.* [2010]). Each of these measures is defined with respect to a particular class.

$$Precision = \frac{\text{Number of documents correctly classified as belonging to a class}}{\text{Total number of documents classified as that class}}$$

$$Recall = \frac{\text{Number of documents correctly classified as belonging to a class}}{\text{Total number of documents belonging to that class}}$$

$$F\ Measure = \frac{(2\ *\ Precision\ *\ Recall)}{(Precision\ +\ Recall)}$$

4.4 Classification Task

In our experiments, we used the Naive Bayes Classifier in Weka to perform the classification tasks using 10-fold cross validation to learn the model. Table 1.1

shows the results obtained by our model using containment measure and longest common subsequence as the two features. On performing experiments with varying set of parameter values, we report the values corresponding to our best set of parameters.

- k gram size = 3
- Threshold similarity in NLTK for clustering = 0.95
- Threshold similarity in ESA for clustering = 0.4

Class	Precision	Recall	F-Measure
Non	0.905	1.0	0.95
Cut	0.733	0.579	0.647
Heavy	0.522	0.632	0.571
Light	0.6	0.474	0.529
Average	0.733	0.737	0.73

Table 4.1: Precision, Recall and F-Measure values obtained for each class in multi-class classification task using Naive Bayes Classifier and the two features

Class	Precision	Recall	F-Measure
Non-plagiarised	0.905	1.0	0.95
Plagiarised	1.0	0.93	0.964
Average	0.962	0.958	0.958

Table 4.2: Precision, Recall and F-Measure values obtained for each class in binary classification task using Naive Bayes Classifier and the two features

		Predicted			
		Non	Cut	Heavy	Light
Original	Non	**38**	0	0	0
	Cut	2	**11**	4	2
	Heavy	2	1	**12**	4
	Light	0	3	7	**9**

Table 4.3: Confusion matrix for classification using the proposed model

Table 1.1 shows the precision, recall and f-measure obtained for each class using our model. The last row contains the weighted average values of each metric with the number of instances in each class as the weights. We observe that the non-plagiarised documents are very well distinguished from the plagiarised documents but the f-measure values for within the plagiarised categories are practically low. Therefore, we understand that the multi-class classification task is a difficult task as such and requires more enhanced techniques. On the other hand, the binary classification task has yielded a high f-measure of 0.958 as it can be observed in Table 1.2. In general, we require to distinguish between the plagiarised and the non-plagiarised documents and might not require the extent of plagiarism involved. Hence we can observe that our model performs the binary classification task significantly well and proves useful.

Table 1.3 shows the confusion matrix for multi-classification task. Majority of the misclassified instances occur between the heavy and light categories. The classifier has achieved an accuracy of 73.68% where accuracy is defined as the ratio of the number of instances correctly classified to the total number of instances (95).

We also compare our results against the results obtained with the Ferret Baseline and the model proposed by Chong *et al.*. Table 1.4 and Table 1.6 show the precision, recall and f-measure values obtained for each class using Chong *et al.* model and Ferret Baseline respectively.

Class	Precision	Recall	F-Measure
Non	0.881	0.974	0.925
Cut	0.667	0.526	0.588
Heavy	0.55	0.579	0.564
Light	0.5	0.474	0.486
Average	0.696	0.705	0.698

Table 4.4: Precision, Recall and F-Measure values obtained for each class using Naive Bayes Classifier and best set of features in Chong *et al.* model

Predicted

		Non	Cut	Heavy	Light
Original	Non	**37**	0	1	0
	Cut	2	**10**	2	5
	Heavy	3	1	**11**	4
	Light	0	4	6	**9**

Table 4.5: Confusion matrix for classification using best set of features in Chong *et al.* model

Class	Precision	Recall	F-Measure
Non	0.884	1.0	0.938
Cut	0.615	0.421	0.5
Heavy	0.419	0.684	0.52
Light	0.5	0.211	0.296
Average	0.66	0.663	0.639

Table 4.6: Precision, Recall and F-Measure values obtained for each class using Naive Bayes Classifier and Ferret Baseline

On comparing our result with the model proposed by Chong *et al.*, we observe that our model has a better recall (equal in some cases) than their model. This is especially significant in the heavy revision category where we have a higher recall but a lower precision. We can understand this intuitively based on the idea that words are being generalised to a common representative based on their semantic similarity. This might lead to a few false positives. Overall, we are able to achieve a better f-measure score than both the baseline models. We also observe that their accuracy (as computed from the confusion matrix in Table 1.5) is 70.52% which is slightly lower than our accuracy. The major reason could be the consideration of semantics which helped in handling acts like paraphrasing using synonym replacement especially in cases belonging to light and heavy revision categories.

4.5 Results with Winnowing

Class	Precision	Recall	F-Measure
Non	0.905	1.0	0.95
Cut	0.714	0.526	0.606
Heavy	0.522	0.632	0.571
Light	0.563	0.474	0.514
Average	0.722	0.726	0.718

Table 4.7: Values obtained for each class in multi-class classification task using Naive Bayes Classifier and the two features obtained after winnowing

Class	Precision	Recall	F-Measure
Non-plagiarised	0.905	1.0	0.95
Plagiarised	1.0	0.93	0.964
Average	0.962	0.958	0.958

Table 4.8: Precision, Recall and F-Measure values obtained for each class in binary classification task using Naive Bayes Classifier and the two features obtained after winnowing

		Predicted			
		Non	Cut	Heavy	Light
Original	Non	**38**	0	0	0
	Cut	2	**10**	4	3
	Heavy	2	1	**12**	4
	Light	0	3	7	**9**

Table 4.9: Confusion matrix for classification using our model with winnowing

Table 1.7 and Table 1.8 show the precision, recall and f-measure for each class obtained using features derived after winnowing the trigrams with a window size of 3. We can observe that the results obtained are almost similar to the previous experiments without winnowing. Although winnowing considers order, values with winnowing are lower mainly due to the loss of few matching n-grams of size less than w + k - 1 (*guarantee threshold)* . From Table 1.9, we observe that the accuracy value for multi-class classification obtained from confusion matrix is 72.63% which is slightly lower than the value obtained from best set of parameters without winnowing.

The similarity using winnowing is usually lower than the direct similarity as we only sample a few trigrams in winnowing and estimate the similarity assuming a similar distribution of matching trigrams over the entire document which might not hold true in general. By choosing a window of size w, we are effectively sampling atleast one trigram from every matching n-gram with a miniumum value of $n = w + 2$. Therefore, in winnowing, we observe that few cases of heavy and light revision have been misclassified as non-plagiarised cases which were properly classified in the previous setting.

Results for binary classification are the same with winnowing which proves that winnowing is an effective technique for increasing the efficiency of the system.

4.6 Effect of Parameters

The effect of varying each individual parameter on the overall performance of the system was observed based on the f-measure values obtained in binary classification task for different parameter settings.

4.6.1 Size of k grams

Based on the experiments performed, the ideal value of k was found out to be 3. Trigrams generally provide a balance to achieve a good precision without losing on recall. The accuracy values obtained for trigrams and bigrams were almost similar. For values of k higher than 3, the accuracy values were low as expected as we fail to match sequences of length less or equal to 3 which are very significant features indicating plagiarism.

4.6.2 Threshold similarity to merge clusters

The values of accuracy tend to have a maximum around a particular value of similarity threshold less than 1. As the threshold decreases, the clusters formed will not be coherent and the average similarity value over all pairs of documents tends to increase due to large sizes of clusters. This generally leads to an increase in threshold used to distinguish the plagiarised cases from the non-plagiarised cases. Therefore plagiarised documents with a slightly low similarity would be labelled as non-plagiarised leading to decrease in recall and overall accuracy. If the threshold is close to 1, the system effectively accounts for replacement of words with common synonyms and this setting would fail to capture cases involving paraphrasing using related words which may not exactly synonymous to the actual word.

4.6.3 Size of window for winnowing

The size of window used in winnowing effectively determines the minimum length of kgram match between the pair of documents that we are interested in. Hence, its effect is similar to the value of k in kgrams.

< Tables or Graphs showing effect of parameters >

CHAPTER 5

CONCLUSIONS AND FUTURE DIRECTIONS

5.1 Conclusions

In this project, we proposed a new algorithm which uses various NLP concepts and knowledge sources (WordNet, Wikipedia) along with text mining methods for extrinsic plagiarism detection. We performed our experiments on the short answers corpus created by Clough and Stevenson (2009) which have varying levels of plagiarism. Results show a significant improvement over the baseline systems which are based on direct string matching methods.

Through this experiment, we realise the importance of considering semantics of words and the order in which they occur. By generalising all the synonyms and related words to their common identities (clusters), we observe that there is an increase in recall which can be trivially understood considering the effect caused on the similarities between the documents. Winnowing has also been proved as an effective technique to improve the efficiency without compromising on the performance. We make use of the Data Mining software, Weka, to build the classifier for combining the similarity scores obtained from different features. However, we realise that the multi-class classification task is still a difficult challenge and requires deeper NLP techniques to obtain richer features which can distinguish the different levels clearly.

5.2 Future Work

Further, we can integrate some techniques using syntactic features from parsing to detect some forms of text rewritten by making changes to the structure of the sentences. Also, we could include *stylometric* analysis for initial prediction of the plagiarised cases and use shallow detection techniques such as n-gram matching using winnowing for efficient retrieval of the candidate source documents. Such strategies in implementation help in increasing the efficiency of the system and in optimising the detection process.

We have restricted our experiments to a small corpus containing fairly less number of documents belonging to Computer Science topics alone. However, we would expect similar kind of results with larger datasets. We can perform our experiments on other large and standard datasets like the PAN corpora and evaluate with their standard evaluation framework.

REFERENCES

Adeel Nawab, R. M., **M. Stevenson**, and **P. Clough**, Detecting text reuse with modified and weighted n-grams. *In Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2012.

Budanitsky, A. and **G. Hirst** (2006). Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, **32**(1), 13–47.

Chong, M. and **L. Specia**, Lexical generalisation for word-level matching in plagiarism detection. *In RANLP*. Citeseer, 2011.

Chong, M., **L. Specia**, and **R. Mitkov**, Using natural language processing for automatic detection of plagiarism. *In Proceedings of the 4th International Plagiarism Conference (IPC 2010), Newcastle, UK*. 2010.

Clough, P. and **M. Stevenson**, Creating a corpus of plagiarised academic texts. *In Proceedings of Corpus Linguistics Conference, CL09 (to appear)*. 2009.

Clough, P. and **M. Stevenson** (2011). Developing a corpus of plagiarised short answers. *Language Resources and Evaluation*, **45**(1), 5–24.

Gabrilovich, E. and **S. Markovitch**, Computing semantic relatedness using wikipedia-based explicit semantic analysis. *In IJCAI*, volume 7. 2007.

Gaizauskas, R., **J. Foster**, **Y. Wilks**, **J. Arundel**, **P. Clough**, and **S. Piao**, The meter corpus: a corpus for analysing journalistic text reuse. *In Proceedings of the Corpus Linguistics 2001 Conference*. 2001.

Hall, M., **E. Frank**, **G. Holmes**, **B. Pfahringer**, **P. Reutemann**, and **I. H. Witten** (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, **11**(1), 10–18.

Karp, R. M. and **M. O. Rabin** (1987). Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, **31**(2), 249–260.

Kestemont, M., **K. Luyckx**, and **W. Daelemans** (2011). Intrinsic plagiarism detection using character trigram distance scores.

Kumar, N., A graph based automatic plagiarism detection technique to handle artificial word reordering and paraphrasing. *In Computational Linguistics and Intelligent Text Processing*. Springer, 2014, 481–494.

Leacock, C. and **M. Chodorow** (1998). Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, **49**(2), 265–283.

Lukashenko, R., **V. Graudina**, and **J. Grundspenkis**, Computer-based plagiarism detection methods and tools: an overview. *In Proceedings of the 2007 international conference on Computer systems and technologies*. ACM, 2007.

Lyon, C., **R. Barrett**, and **J. Malcolm** (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*.

Manber, U. *et al.*, Finding similar files in a large file system. *In Usenix Winter*, volume 94. 1994.

Maurer, H. A., **F. Kappe**, and **B. Zaka** (2006). Plagiarism-a survey. *J. UCS*, **12**(8), 1050–1084.

Meuschke, N. and **B. Gipp** (2013). State-of-the-art in detecting academic plagiarism. *International Journal for Educational Integrity*, **9**(1).

Mihalcea, R., **C. Corley**, and **C. Strapparava**, Corpus-based and knowledge-based measures of text semantic similarity. *In AAAI*, volume 6. 2006.

Nahnsen, T., **O. Uzuner**, and **B. Katz** (2005). Lexical chains and sliding locality windows in content-based text similarity detection. *CSAIL Memo*.

Potthast, M., **B. Stein**, **A. Barrón-Cedeño**, and **P. Rosso**, An evaluation framework for plagiarism detection. *In Proceedings of the 23rd international conference on computational linguistics: Posters*. Association for Computational Linguistics, 2010.

Ram, R. V. S., **E. Stamatatos**, and **S. L. Devi**, Identification of plagiarism using syntactic and semantic filters. *In Computational Linguistics and Intelligent Text Processing*. Springer, 2014, 495–506.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*.

Schleimer, S., **D. S. Wilkerson**, and **A. Aiken**, Winnowing: local algorithms for document fingerprinting. *In Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 2003.

Silber, H. G. and **K. F. McCoy** (2002). Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, **28**(4), 487–496.

Stamatatos, E. (2009). Intrinsic plagiarism detection using character n-gram profiles. *threshold*, **2**, 1–500.

Su, Z., **B.-R. Ahn**, **K.-Y. Eom**, **M.-K. Kang**, **J.-P. Kim**, and **M.-K. Kim**, Plagiarism detection using the levenshtein distance and smith-waterman algorithm. *In Innovative Computing Information and Control, 2008. ICICIC'08. 3rd International Conference on*. IEEE, 2008.

Tan, P., **M. Steinbach**, and **V. Kumar**, *Introduction to Data Mining.* Pearson Education, Limited, 2014. ISBN 9780273769224. URL `http://books.google.co.in/books?id=T0dzMAEACAAJ`.

Uzuner, Ö., **B. Katz**, and **T. Nahnsen**, Using syntactic information to identify plagiarism. *In Proceedings of the second workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics, 2005.

Wu, Z. and **M. Palmer**, Verbs semantics and lexical selection. *In Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1994.