Plagiarism detection and Grading of Documents

Aakash Bhalla
Information Technology
National Institute Of Technology Karnataka
201IT201
kashbhalla.201it201@nitk.edu.in

Sanket Hanagandi
Information Technology
National Institute Of Technology Karnataka
201IT154
sankethanagandi.201it154@nitk.edu.in

Mayur Jinde
Information Technology
National Institute Of Technology Karnataka
201IT135
jindemayur.201it135@nitk.edu.in

Shaulendra Kumar
Information Technology
National Institute Of Technology Karnataka
201IT159
shaulendragmailcom.201it159@nitk.edu.in

***Abstract*—In current times professors are busy with projects and different academic works checking the answer sheet of the individuals and detection the plagiarism among answer sheets would be a more tedious task to complete which takes too much time. This brings the call for us to come up with a project to grade the answer sheets and find plagiarism among the students using semantic similarity. If the students write the synonyms it can be easily detected for grading and plagiarism in the project with help of NLP.**

I. Introduction

Text-based comparison, in which the text of two papers is examined to find overlaps or similarities, is one of the most widely used techniques for detecting plagiarism. Plagiarism detection systems compare the text in the documents, frequently using complex algorithms to find matches that are identical or almost identical as well as paraphrased or rephrased information. These technologies analyse a variety of linguistic elements, including word choice, sentence structure, and even formatting, to assist find possible instances of plagiarism.

Document submission to a plagiarism detection programme or piece of software initiates the plagiarism detection process. The programme then evaluates the provided document against a vast database of sources, which might include academic journals, published papers, publicly accessible records, and other web sources. The tool searches for matches, highlighting comparable material and showing how similar the two papers are to one another. Some plagiarism detection software also include thorough reports that list the sources of the text that was matched and indicate certain parts that could be instances of plagiarism.

In addition to academic writing, professional and creative writing, such as journalism, content production, and copyright infringement lawsuits, also require the ability to identify plagiarism. For instance, plagiarism checking software is used in journalism to make sure that articles are unique and free of plagiarised text. In the area of content production, plagiarism detection technologies are essential for spotting situations when information has been lifted verbatim from other websites and preventing the development of and publication of plagiarised content.

It might be difficult to identify plagiarism. The incapability of plagiarism detection technologies to pick up on subtle kinds of plagiarism, such paraphrase, as well as false positives and false negatives, are just a few of its drawbacks. A programme may, for instance, indicate a piece of writing as possibly copied based on linguistic similarities, but it might not take into consideration idioms or specialised terminology used in a particular industry, which could result in false positives. On the other hand, if the text has been considerably paraphrased or rephrased, a tool might not indicate a document as possibly plagiarised, leading to false negatives.

In conclusion, plagiarism detection is an essential procedure for safeguarding written work's originality, quality standards, and academic integrity. As internet material becomes more widely available, plagiarism detection programmes are becoming increasingly important in spotting possible instances of plagiarism by contrasting the content of two or more publications. Notwithstanding their drawbacks, plagiarism detection systems are important tools for academics, educators, and content producers who want to encourage ethical writing habits and make sure that original work is acknowledged and cited properly.

II. Dataset

The Data Set is imported from the Brown Corpus library of Python.

The Brown Corpus is a corpus of English-language writings that has served as a standard for work in computer linguistics and natural language processing. Around one million words of written and spoken language from a variety of sources, including novels, newspapers, magazines, official documents, and conversations, made up this collection, which was put together in the 1960s. The Brown Corpus is a useful tool for researching the prevalence and distribution of words and grammatical constructions in English literature since it is annotated with part-of-speech tags. It has been applied in several research to examine linguistic variations, evaluate linguistic patterns, and create computer models for tasks involving natural language processing, including sentiment analysis, text categorization, and machine translation.

III. Methodology

A. Data Preprocessing

1) **Lower Casing**:Lowercasing is a typical data preparation method used in Python's Natural Language Toolkit (NLTK) package for natural language processing (NLP). To maintain uniformity in word representations and minimise the dimensionality of the data, it entails changing all text to lowercase letters.

2) **Tokenization**: Tokenization is the procedure used to separate a written document into tokens, or discrete words. It aids in the transformation of unstructured text data into a more organised format that can be easily processed by machines, which is a crucial step in natural language processing (NLP) and text analysis jobs.

3) **Punctuation Mark Removal**: Punctuation mark removal is a typical data preparation step, especially when utilising Python's Natural Language Toolkit (NLTK) for NLP applications. Punctuation marks are unique symbols used to break up sentences in written language. These symbols include commas, periods, exclamation points, question marks, and quote marks. Yet, when carrying out activities like text analysis or sentiment analysis, they are frequently superfluous and can introduce noise to text data.

4) **Stop Word Removal**: Data pretreatment for natural language processing (NLP) applications frequently includes the elimination of stop words. Stop words are frequently used terms in a language that are frequently seen as having minimal value in text analysis since they are quite prevalent and don't have a lot of meaning on their own. The words ”a,” ”an,” ”the,” ”is,” ”in,” ”and,” ”of,” ”that,” ”it,” and ”with” are examples of stop words in English.

5) **Stemming**: Natural language processing (NLP) and text mining employ the stemming approach to condense words to their fundamental or basic form, sometimes referred to as the ”stem” or ”root” word. Normalizing words and reducing variances in text data is a frequent step in data preparation, which can enhance text analysis and information retrieval tasks.

6) **Lemmatization**:Lemmatization is a method used in natural language processing (NLP) to break down words into their lemma, which is the basic or root form. It is a crucial stage in the preparation of data since it normalises text data by reducing words to their dictionary or canonical forms, which can help with tasks like text classification, sentiment analysis, and information retrieval.

B. Model Training: Word2Vec

Word2Vec is a well-liked natural language processing (NLP) method for extracting word embeddings, sometimes referred to as word representations, from massive amounts of text. It was created in 2013 at Google by Tomas Mikolov and others. The main concept of Word2Vec is to map related words to adjacent places in a high-dimensional space by representing words as continuous dense vectors.

Word2Vec uses the Continuous Bag of Words (CBOW) and Skip-gram designs as its primary building blocks. Although in Skip-gram, the model predicts the context words given a target word, in CBOW, the model predicts the target word given its context words (words in its surrounding window). Both architectures entail training a neural network using a sizable corpus of text data, and word embedding training is an unsupervised learning method used to determine the weights of the neural network.

The taught word embeddings from the Word2Vec model may be used to a variety of NLP applications, including text classification, named entity identification, sentiment analysis, and others. The model can grasp word connections and similarities thanks to the word embeddings, which also help it perform better on downstream NLP tasks.

By offering a potent approach to represent words as continuous vectors, which can be employed in many downstream tasks to increase their performance, Word2Vec has been widely used in numerous applications and considerably improved the area of NLP.

Testing the Model with sample documents:

- A set of 15 sample documents along with one source document is tested using the Model, to check for plagiarism and also grade the documents.
- The documents are initially preprocessed and then fed to the Model after vectorizing the words in the documents using word2Vec library.

The documents are then compared for the results as follows:

- *Plagiarism detection*: All the documents are compared with the one another for the check of any plagiarism and the result is given as output.
- *Grading*: The sample documents are compared with one source document for the grading.

Finally all the sample documents are graded and checked for plagiarism and the result is shown as the percentage as output. The flowchart below depicts the flow of our Model:

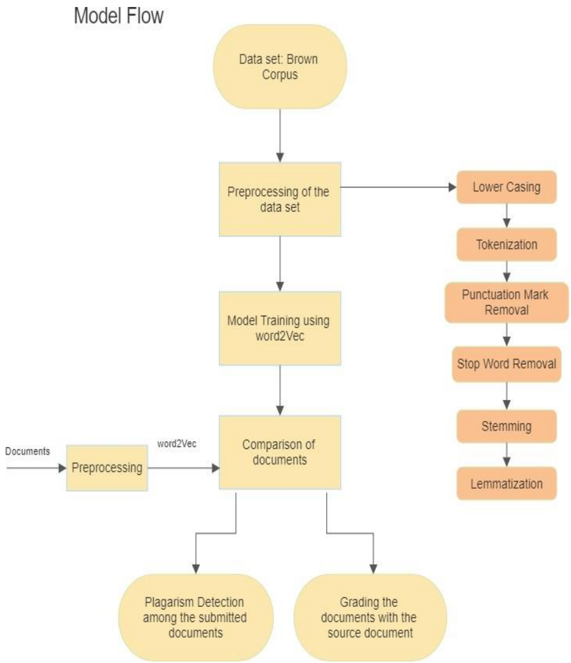
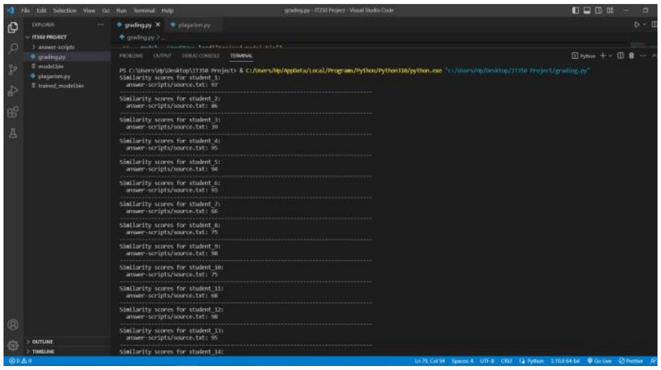


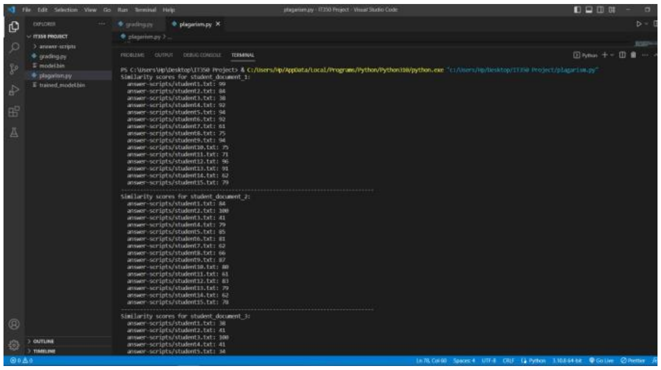
Fig. 1. Flow of Model

IV. Results And Analysis

The trained model was run for 15 sample documents. The results as below shown in fig 2 and fig 3. The sample documents when tested for plagiarism by comparing with one another gives the output in a form of a matrix that shows how similar any two documents are to each other as percentage. Also, the grading of the sample documents was done by comparing with the source document and the results were given in the form of percentage as shown in the figures.



```
PS C:\Users\Hp\Desktop\IT350 Project> & C:/Users/Hp/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/Hp/Desktop/IT350 Project/grading.py"
Similarity scores for student_1:
  answer-scripts/source.txt: 97
--------------------------------------------------------------------------------
Similarity scores for student_2:
  answer-scripts/source.txt: 86
--------------------------------------------------------------------------------
Similarity scores for student_3:
  answer-scripts/source.txt: 39
--------------------------------------------------------------------------------
Similarity scores for student_4:
  answer-scripts/source.txt: 95
--------------------------------------------------------------------------------
Similarity scores for student_5:
  answer-scripts/source.txt: 94
--------------------------------------------------------------------------------
Similarity scores for student_6:
  answer-scripts/source.txt: 93
--------------------------------------------------------------------------------
Similarity scores for student_7:
  answer-scripts/source.txt: 66
--------------------------------------------------------------------------------
Similarity scores for student_8:
  answer-scripts/source.txt: 75
--------------------------------------------------------------------------------
Similarity scores for student_9:
  answer-scripts/source.txt: 98
--------------------------------------------------------------------------------
Similarity scores for student_10:
  answer-scripts/source.txt: 75
--------------------------------------------------------------------------------
Similarity scores for student_11:
  answer-scripts/source.txt: 68
--------------------------------------------------------------------------------
Similarity scores for student_12:
  answer-scripts/source.txt: 98
--------------------------------------------------------------------------------
Similarity scores for student_13:
  answer-scripts/source.txt: 95
--------------------------------------------------------------------------------
Similarity scores for student_14:
```



```
PS C:\Users\Hp\Desktop\IT350 Project> & C:/Users/Hp/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/Hp/Desktop/IT350 Project/plagarism.py"
Similarity scores for student_document_1:
  answer-scripts/student1.txt: 99
  answer-scripts/student2.txt: 84
  answer-scripts/student3.txt: 38
  answer-scripts/student4.txt: 92
  answer-scripts/student5.txt: 94
  answer-scripts/student6.txt: 92
  answer-scripts/student7.txt: 61
  answer-scripts/student8.txt: 75
  answer-scripts/student9.txt: 94
  answer-scripts/student10.txt: 75
  answer-scripts/student11.txt: 71
  answer-scripts/student12.txt: 96
  answer-scripts/student13.txt: 91
  answer-scripts/student14.txt: 62
  answer-scripts/student15.txt: 79
--------------------------------------------------------------------------------
Similarity scores for student_document_2:
  answer-scripts/student1.txt: 84
  answer-scripts/student2.txt: 100
  answer-scripts/student3.txt: 41
  answer-scripts/student4.txt: 79
  answer-scripts/student5.txt: 85
  answer-scripts/student6.txt: 81
  answer-scripts/student7.txt: 62
  answer-scripts/student8.txt: 66
  answer-scripts/student9.txt: 87
  answer-scripts/student10.txt: 80
  answer-scripts/student11.txt: 61
  answer-scripts/student12.txt: 83
  answer-scripts/student13.txt: 79
  answer-scripts/student14.txt: 62
  answer-scripts/student15.txt: 78
--------------------------------------------------------------------------------
Similarity scores for student_document_3:
  answer-scripts/student1.txt: 38
  answer-scripts/student2.txt: 41
  answer-scripts/student3.txt: 100
  answer-scripts/student4.txt: 41
  answer-scripts/student5.txt: 34
```

V. Conclusion

With the use of plagiarism detection software, technology has recently made it simpler to identify plagiarism. To find any instances of plagiarism, these programmes utilise algorithms to compare a manuscript to a vast database of other texts. It is crucial to remember that these technologies are not perfect, and human judgement is still required to assess the seriousness and intentionality of any alleged plagiarism. Nowadays, technology has made it easier to detect plagiarism with the use of plagiarism detection tools. These programmes use algorithms to compare a document to a huge database of other texts in order to detect any instances of plagiarism. It is important to keep in mind that these technologies are not flawless and that human judgement is still needed to determine the gravity and purpose of any claimed copying. Automated grading systems can be useful for giving students immediate comments on their papers, but they cannot take the place of a human grader's knowledge and perspective. Automatic programmes can grade a student's writing in terms of grammar and spelling, but they are unable to comment on the content of the writing or the calibre of the arguments made. In conclusion, while technology can help with document grading and plagiarism detection, human judgement and experience are still crucial for guaranteeing academic integrity and giving students useful feedback.