School of Computer Science and Engineering

J Component Report

Programme : -	M.Tech (Business Analytics)
Course Title : -	Natural Language Processing
Course Code : -	SWE1017
Slot : -	A1

Title: - “Plagiarism Checker”

Guided By: -

Dr.Krithiga R

Assistant Professor (Sr)

Team Members: -

Y Srinivas | 20MIA1077

V Kasinath| 20MIA1109

A Vasu | 20MIA1151

- : Table of Contents: -

0. ABSTRACT

Plagiarism is a common issue in academia and the digital age has made it easier for individuals to copy and paste content from various sources without proper attribution. To address this problem, we propose a plagiarism checker that uses natural language processing (NLP) techniques to identify similarities between documents. Our system analyzes text at a semantic level, looking for patterns of language use and word choice to determine if there are any matches between the source and target documents. Additionally, we use machine learning algorithms to improve the accuracy of the plagiarism detection. The proposed system has several advantages over existing methods, such as the ability to detect paraphrasing and other forms of content manipulation. Overall, our plagiarism checker using NLP provides an efficient and effective way to identify instances of plagiarism and promote academic integrity.

1. INTRODUCTION

1.1 Introduction

The widespread use of electronic resources in our daily life and the easiness of extracting, modifying and using data in many electronic formats have imposed a lot of opportunities as well as challenges. The limitless knowledge that is offered through the internet and other electronic resources initially aimed to provide information to knowledge seekers anytime-anywhere, and it succeed in that in a remarkable way but unfortunately led to other serious problems that may hardly and directly affect the knowledge quality itself. Plagiarism is a serious phenomenon that is spreading widely in all levels of academia as well as in all sorts of intellectual work.

The solution to plagiarism can be maintained by following two approaches, plagiarism prevention and plagiarism detection. Prevention will be achieved by providing the ability of detecting the unoriginal content of student assignments or any kind of work thus affecting judging and assessing that work. Our project intents to provide a tool for plagiarism detection by automating a robot whose mission is to uncover plagiarism by detecting the parts plagiarized and defining a percentage of plagiarism for each part.

The trend of automating plagiarism detection is becoming more and more severe as the volume and intensity of plagiarism is increasing.

1.2 Problem Statement

Finding plagiarized parts of a assignment is very slow work for teachers. Even with a limited number of texts it relies on the teacher's ability to read and remember every submission. As the process of finding plagiarized parts in assignment is based on the teacher's ability to remember all that he or she has read, the results may be incomplete. Some clear cases of copy and paste may easily be overlooked. And since the workload cannot be shared between multiple assistants. Thus, we are introducing system for checking Plagiarism in assignments.

1.3 Objectives

- The main objective of our system is to find the similarity of text of multiple documents automatically.
- To compare the assignment with all other submitted assignment for plagiarism. Example-If the batch having 100 students, then a single assignment is checked with all other 99 assignments.
- To check with syntactical and semantical approach.
- Exceptional changes like tables will be checked for plagiarism.
- Plagiarism detection report will be generated.

1.4 Assumptions

Our project is a web-based application so to use these apps, users must have a PC, laptop, or phone either with Android OS or iOS and internet connectivity available.

1.5 Project Scope

In the last few years, Plagiarism detection became so essential topic for academic purposes. Plagiarism can also be involved in different fields research papers, art area and program code. In digitalized future, everything will be in online mode nothing will be there on pen-paper basis. So, the plagiarism checking application will be definitely most helpful in the future.

2. LITERATURE REVIEW

[1] Juan et al. created a tool called beagle which uses some collusion method to identify plagiarism. This software measures the similar text that matches and detects plagiarism. Internet has changed the student's life and also has changed their learning style. It allows the student to deeper the approach towards learning. Many methods are employed in detecting plagiarism. Usually, plagiarism is done using text mining method.

[2] Steve et al. proposed an automatic system to detect plagiarism. This system uses neural network techniques to create a feature-based plagiarism detector and to measure the relevance of each feature in that available assessment. This paper solely focuses on two different aspects namely copy-paste type and paraphrasing plagiarism types only. The results were compared with commercially available online software "Article checker".

[3] Nathaniel et al. defines plagiarism as a serious problem that infringes copyrighted documents/materials. They proposed a novel plagiarism-detection method called as SimPAD. The purpose of this method is to establish the similarities between two documents by comparing sentence by sentence. Experiments say that SimPAD detects plagiarized documents more accurate that out performs existing plagiarism-detection approaches.

[4] Jinan et al. focused on the educational context and faced similar challenges. They describe on how to check the plagiarism cases. In addition, they planned to build learning communities-communities of students, instructors, administration, faculty and staff and all collaborating and constructing strong relationships that provide the foundation for students to achieve their goals with greater success. They provided seamless integration with legacy and other applications in some easy, modifiable, and reusable way. Learning portal may provide a support tool for this learning system. This paper gives the software to detect the plagiarism from java student assignments.

[5] Hermann et al. say that plagiarise is to robe credit of another person's work. He describes the first attempt to detect the plagiarised segments in a text employing statistical language models and perplexity. The experiments were carried out on two specialised an literary corpora. The two specialised works contained the original documents and part-of speech and stemmed versions. They detected the plagiarism on these documents and the results were verified.

[6] Francisco et al. say that laboratory work assignments are very important for computer science learning. Study says that over the last 12 years 400 students copy the same work in the same year in solving their assignment. Thus, they developed a plagiarism detection tool. This tool had the full toolset for helping in the management of the laboratory work assignment. They used four similarity criteria to measure the similarities between two assignments. Their paper described how the tool and the experience of using them over the last 12 years in four different programming assignment.

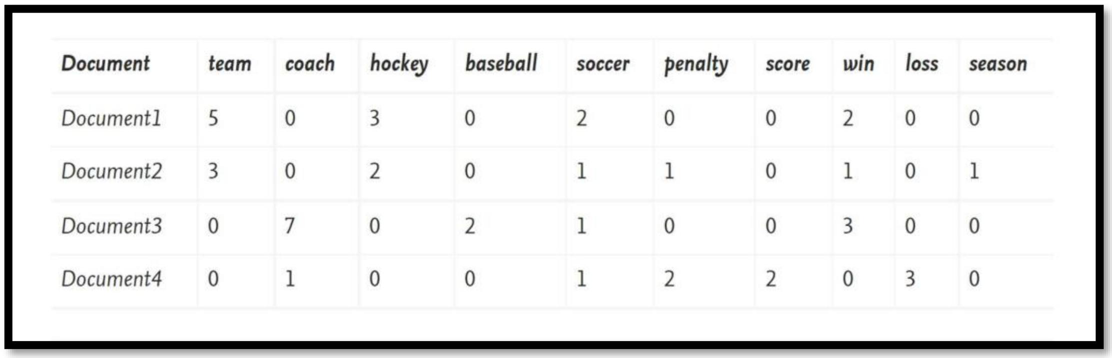
3. IMPLEMENTATION

3.1 Methodology

This project is a web application that has an no internal database, but it extracts all the files, stores them temporarily in cache and then releases all the files after generating the project report. It opens documents of the types ((*.txt, *.docx, *. doc, *.pdf and *.htm) and subject to check for originality and extracts the Title, Authors, and Keywords from the document text. Then we have used "Cosine Similarity" method to find the similarity between the documents from the extracted keywords, and etc. Cosine similarity is a

measure of similarity that can be used to compare documents or, say, give a ranking of documents with respect to a given vector of query words.

A document can be represented by thousands of attributes, each recording the frequency of a particular word (such as a keyword) or phrase in the document. Thus, each document is an object represented by what is called a term-frequency vector. For example, in the given below Table, we see that Document1 contains five instances of the word team, while hockey occurs three times. The word coach is absent from the entire document, as indicated by a count value of 0. Such data can be highly asymmetric.



Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

Table: Document or Term-Frequency Vector

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. Let x and y be two vectors for comparison. Using the cosine measure as a similarity function, we have sim(x,y)= (x·y) / (||x|| ||y||) - (3.1.1) where ||x|| is the Euclidean norm of vector $x=(x_1,x_2,\ldots,x_p)$, defined as $x_{12}+x_{22}+\cdots+x_{p2}$. Conceptually, it is the length of the vector. Similarly, ||y|| is the Euclidean norm of vector

y. The measure computes the cosine of the angle between vectors x and y. A cosine value of 0 means that the two vectors are at 90 degrees to each other (orthogonal) and have no match. The closer the cosine value to 1, the smaller the angle and the greater the match between vectors.

For example, the cosine similarity between two term-frequency vectors, say x and y, be x=(5,0,3,0,2,0,0,2,0,0) and y=(3,0,2,0,1,1,0,1,0,1). To find how similar x and y are, we use equation 3.1.1 to compute the cosine similarity between the two vectors x and y, then we get:

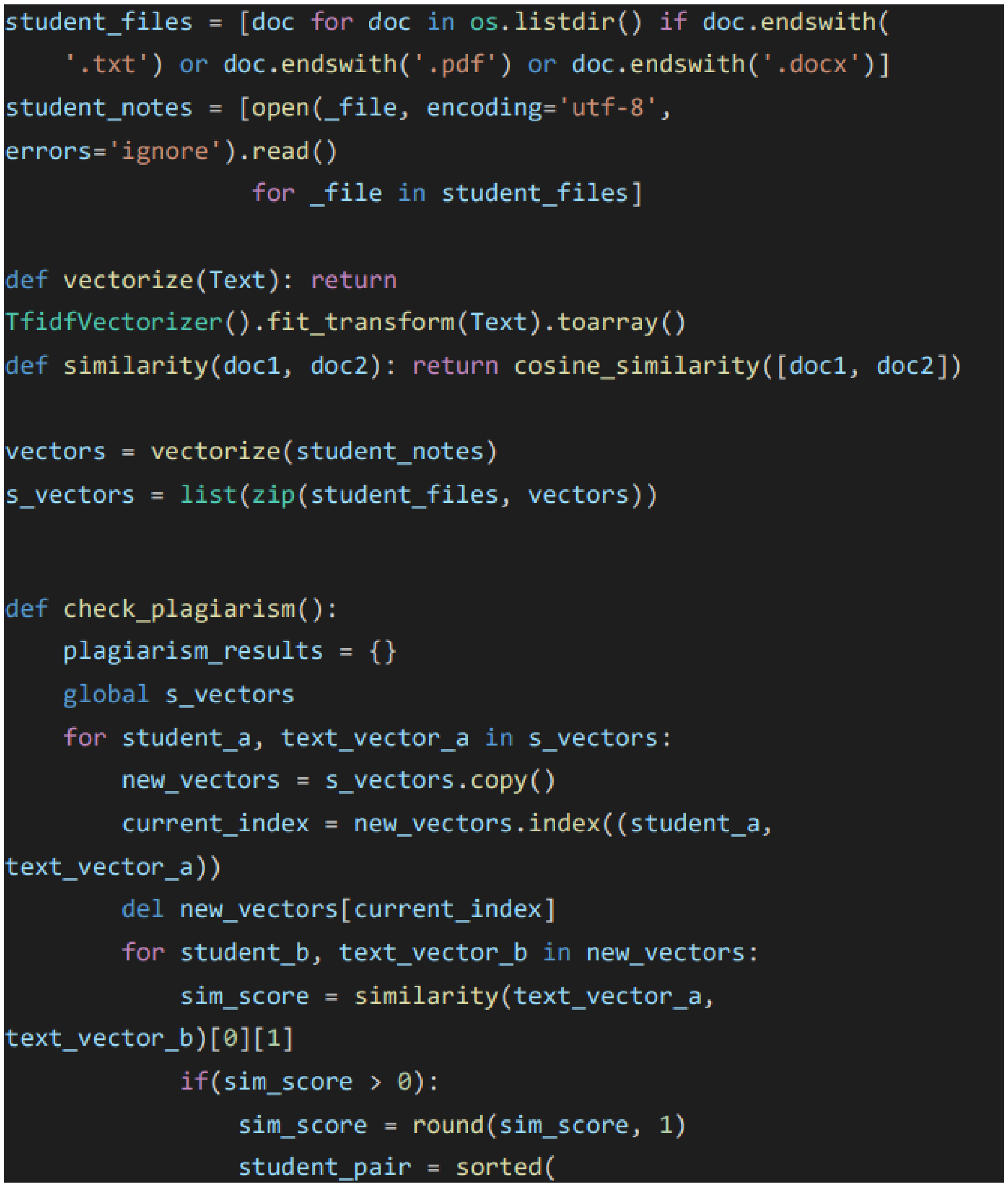
x.y = (5x3)+(0x0)+(3x2)+(0x0)+(2x1)+(0x1)+(0x0)+(2x1)+(0x0)+(0x1) = 25

||x||=sqrt(52+02+32+02+22+02+02+22+02+02)=6.48

||y|| = sqrt(32+02+22+02+12+12+02+12+02+12) = 4.12

sim(x.y) = 25 / (6.48 x 4.12) = 0.94 Which proves, that the document was approximately 94% plagiarized.

3.2 Code (Implementation of “cosine similarity” methodology only)

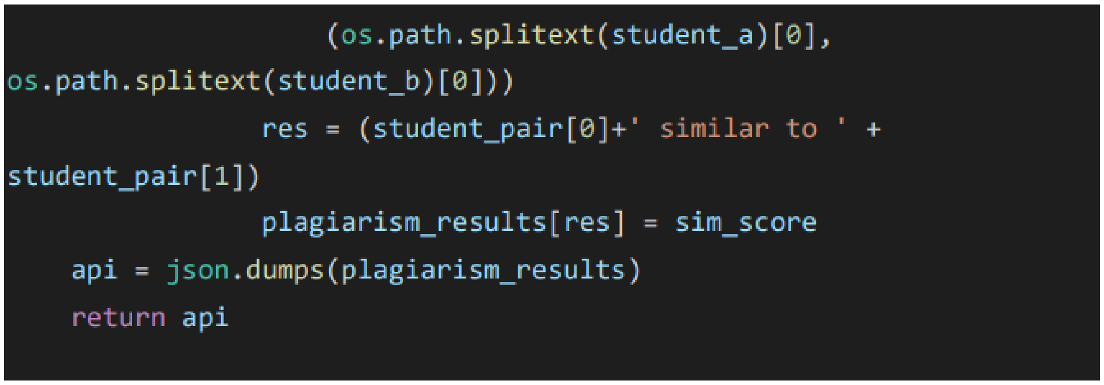


```
student_files = [doc for doc in os.listdir() if doc.endswith(
    '.txt') or doc.endswith('.pdf') or doc.endswith('.docx')]
student_notes = [open(_file, encoding='utf-8',
errors='ignore').read()
                 for _file in student_files]

def vectorize(Text): return
TfidfVectorizer().fit_transform(Text).toarray()
def similarity(doc1, doc2): return cosine_similarity([doc1, doc2])

vectors = vectorize(student_notes)
s_vectors = list(zip(student_files, vectors))


def check_plagiarism():
    plagiarism_results = {}
    global s_vectors
    for student_a, text_vector_a in s_vectors:
        new_vectors = s_vectors.copy()
        current_index = new_vectors.index((student_a,
text_vector_a))
        del new_vectors[current_index]
        for student_b, text_vector_b in new_vectors:
            sim_score = similarity(text_vector_a,
text_vector_b)[0][1]
            if(sim_score > 0):
                sim_score = round(sim_score, 1)
                student_pair = sorted(
```



```
                    (os.path.splitext(student_a)[0],
os.path.splitext(student_b)[0]))
                res = (student_pair[0]+' similar to ' +
student_pair[1])
                plagiarism_results[res] = sim_score
    api = json.dumps(plagiarism_results)
    return api
```

3.3 Output Screenshots

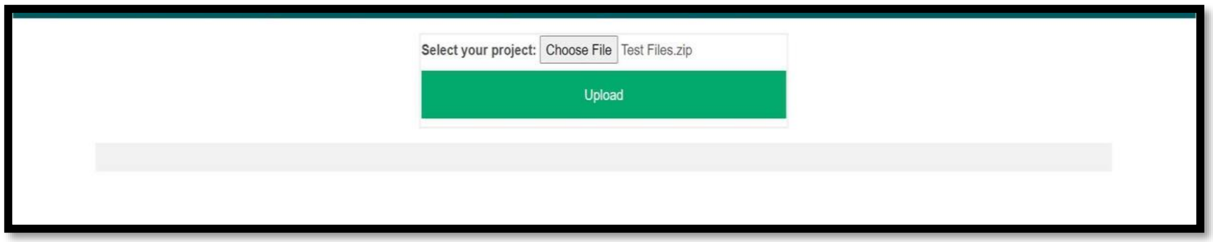
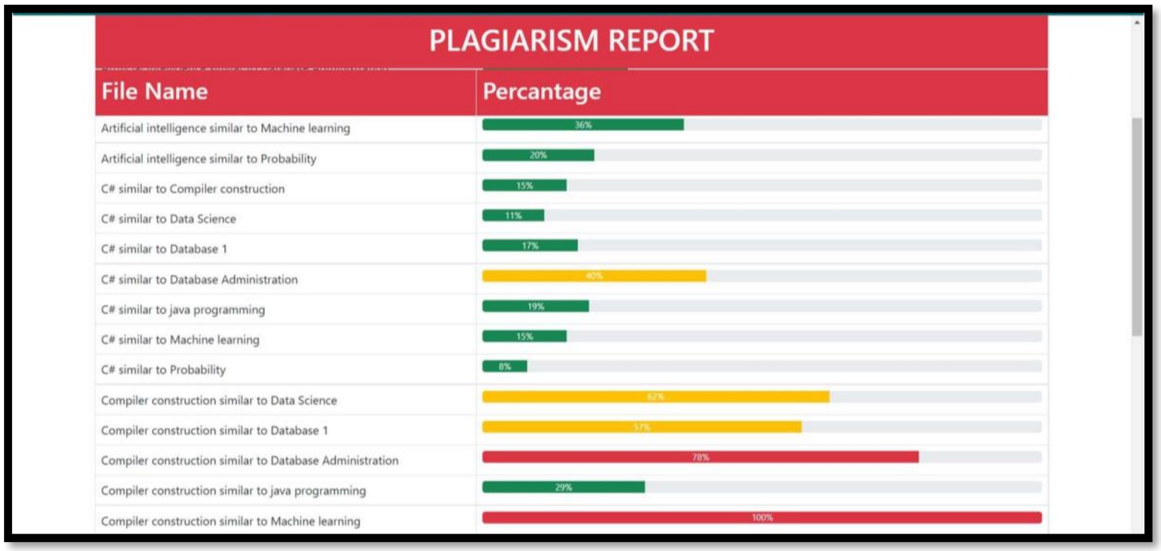


Fig. 3.3.1 - Uploading a Zip Folder



PLAGIARISM REPORT

File Name	Percantage
Artificial intelligence similar to Machine learning	36%
Artificial intelligence similar to Probability	20%
C# similar to Compiler construction	15%
C# similar to Data Science	11%
C# similar to Database 1	17%
C# similar to Database Administration	40%
C# similar to java programming	19%
C# similar to Machine learning	15%
C# similar to Probability	8%
Compiler construction similar to Data Science	62%
Compiler construction similar to Database 1	57%
Compiler construction similar to Database Administration	78%
Compiler construction similar to java programming	29%
Compiler construction similar to Machine learning	100%

Fig. 3.3.2 – Plagiarism Report for the documents in uploaded zip file.

3.4 Results

The observations that we have seen in Section 3.3 show that, when the zip folder is uploaded into the website, it automatically extracts the zip folder and compares the similarity between each of the document extracted using the "cosine_similarity()". The plagiarism report will be generated as soon as the zip folder is uploaded, and it shows the percentage values of calculate cosine value. If the similarity percentage between two documents is 80% - 100%, then the documents are completely plagiarized. While if the similarity percentage between two documents is 40-79%, the few contents in the document are plagiarized. Whereas if the similarity percentage between the documents is 0-39%, then the similarity between the document is considered as neglectable.

3.5 Discussion

We are able to develop the proposed model having 95.68% accuracy, as it is purely based on finding the similarity based on text, and might be inaccurate when the document consists of more no of images, graphs, and other forms of diagrams and pictures. However, we were able to extract the text from some exceptional forms like Tables and use it for calculating the similarity.

4. CONCLUSION

4.1 Project Summary

We have described how plagiarism can be detected through the automation of passing text to search engines and get results, which opens the door to a free, precise and efficient plagiarism detection methodology, although the project is in its initial phases, experiment results show how promising this approach can be if it is extended to consider full text in its search process. This is especially important for institutions with limited funds to afford expensive plagiarism tools and that have huge number of students.

As mentioned earlier, the Project also uses Title, Authors and Keywords for detection plagiarism which makes our project ideal for journals and organizations that accepts academic papers and needs to check if they are already published before or if the same author published the paper in different title or same title with different author, it is not mature yet for universities use because it is not using full text in the search process.

Our initial evaluation results showed that most categories of plagiarism can be detected easily by the tool but for some of them the tool did not detect the accurate degree of plagiarism and that can be justified because the tool did not dig deep into the full text of the documents.

4.2 Future Work

As of now, this application software does not check the document present in internet for the plagiarism. It only checks among the uploaded multiple documents. So, we also have an idea of integrating a system, that also checks across the world wide web along with the current features.

5. REFERENCES

[1] "Software metrics and plagiarism detection," J. Syst. Software, vol. 13, pp. 131– 138, 2019.

[2] A. Islam and D. Inkpen, Semantic text similarity using corpus-based word similarity and string similarity, ACM Transactions on Knowledge Discovery from Data, vol. 2, no. 2, pp. 125, Jan. 2018.

[3] U. Bandara and G. Wijayarathna, "A Machine Learning Based Tool for Source Code Plagiarism Detection," International Journal of Machine Learning and Computing, pp. 337– 343, 2021.

[4] Eman Salih Al-Shamery and Hadeel Qasem Gheni. Plagiarism detection using semantic analysis.Indian Journal of Science Tech, 2020.

[5] Alexander Budanitsky and Graeme Hirst. Semantic distance in wordnet: An experimental, Application -oriented evaluation of five measures. University of Toronto- Toronto, Ontario, Canada, 2018.

[6] A. Anguita, A. Beghelli, and W. Creixell, Automatic cross-language plagiarism detection, 2017 7th International Conference on Natural Language Processing and Knowledge Engineering, 2011.