

# Intelligent Assignment Plagiarism Detection System Using Machine Learning

N.Naga Bharghavi<sup>1</sup>, B.Sai Deepthi<sup>2</sup>,

<sup>1</sup>Final year MCA Student, Department of Computer Applications, GIET Engineering College, Andhra Pradesh, India

<sup>2</sup> Assistance Professor, GIET Engineering College, Andhra Pradesh, India

\*\*\*

**Abstract** – Plagiarism has really become a big deal in schools because of how easy it is to find stuff online and just copy and paste. Old ways of catching plagiarism take forever, don't work well with tons of papers, and are just plain ineffective. So, we're suggesting a new automatic system that uses AI to spot cheating. This system lets you upload all kinds of docs, like PDFs or texts. Once the files are uploaded, the text is grabbed and checked out. The main way it checks for theft? Using TF-IDF and cosine similarity to figure out if something's been copied, then it rates it as high plagiarism, partly ripped off, or not at all. Plus, there's an AI part of the app that looks to see if a paper was written by a person or another AI thing. The site is built on Flask and lets you sign up, track old uploads, see how well the system is doing, look at pretty graphs of matches, and even download the reports in various formats. All in all, this project should help keep things honest, save time, and make sure the job gets done right.

**Key Words:** *Plagiarism Detection, Machine Learning, Natural Language Processing(NLP), Artificial Intelligence(AI), TF-IDF(Term Frequency-Inverse Document Frequency), Cosine Similarity, Text Mining, Document Similarity Analysis, Feature Extraction, Text processing, Tokenization, Stop Word Removal, Vectorization, Similarity Score, Academic Integrity, AI Content Detection, Transformer models, Human-written Content Detection, AI-generated Content Detection, Document Classification, Flask Framework, User Authentication, PDF Processing, Text Extraction, Data analysis, Performance Visualization, Report Generation, Web-Based Application.*

## 1.INTRODUCTION

The problem of plagiarism has risen greatly in the context of academe as well as corporate settings owing to the fast-growing nature of content generated via digital media as well as the ease of availability of content over the internet. This has made it easier for people to plagiarize content. Detecting plagiarism manually is an ineffective approach as it is cumbersome and tedious and there is a high probability of error.

To solve this issue, the use of automated plagiarism detection tools based on machine learning and natural language processing has gained popularity. Automated plagiarism detection is more effective because the analysis of the text is done effectively and efficiently. For our

purpose, we design a web-based application that uses Flask. Users can submit documents in various file formats such as pdf and text.

The main functionalities of the system include the use of TF-IDF and cosine similarity, which will play an important role in calculating similarities between the documents. With the use of TF-IDF and cosine similarity, the system will be able to detect not only similar text but also partial and paraphrasing similarities. Other functionalities implemented in the system include user authentication, history, results visualization, and downloadable reports.

Through automation of the plagiarism detection process, the system ensures efficiency in the process.

## 2. Methodology

### 2.1 User Authentication

The User Authentication Module is used to control the processes of user account creation and user login into the system. It controls the access of users to the system and allows performing the plagiarism checks only for those who have valid user account and passwords. Users can register by entering their username and password. Such information will be saved to the database and will be used later during user login process. The User Authentication Module creates personalized access and saves the information of each particular user including the results of plagiarism checks and the history of the work in the system. Also, it increases the security level of the application.

### 2.2 File Upload and Management

The File Upload and Management Module allows users to upload files for the plagiarism check. The module supports files in various formats such as PDF files and text documents. Uploaded files will be stored in the certain folder on the server in order to perform further processing. Files' names will not be overwritten and appropriate file management will be ensured. The validation process of the uploaded files will be performed in order to make sure that the files are formatted correctly.

### 2.3 Text Extraction and Preprocessing

The function of the Text Extraction and Preprocessing Module is to extract text information from uploaded

documents and pre-process this text in a manner suitable for further analysis. To extract text information from PDF files, the PyPDF2 library can be used. In the case of plain text files, the process involves simply reading these files according to the proper encoding rules. Next, text data is processed through a number of operations which include removal of unwanted symbols, converting all letters into lower case, etc. This step is important because it allows to improve the quality of data before further processing operations are performed. Moreover, proper text preprocessing contributes to better performance of the similarity detection module.

## 2.4 Feature Extraction (TF-IDF)

To represent textual data as numerical information, the TF-IDF (term frequency – inverse document frequency) method can be used. By applying this method, more importance is assigned to certain words in a given text while commonly used words receive less attention. The end result is an array of values for each text document. These arrays (vectors) are used later to perform similarity comparisons.

## 2.5 Similarity Calculation (Cosine Similarity)

The Similarity Calculation Module employs cosine similarity in calculating the similarity between two document vectors. The calculation measures the cosine of the angle between the TF-IDF vectors of both documents, thereby determining their similarity. Higher cosine similarity values imply higher similarities between the documents. Cosine similarity performs well in not only detecting exact matches but also identifying modified and paraphrased copies. The module calculates the similarity score in percentages. Then based on the score, it classifies the degree of plagiarism. This module plays an important role in achieving the desired result from the system.

## 2.6 Result Visualization and Report Generation

The Result Visualization and Report Generation Module enables visualization of plagiarism detection outcomes. The module shows the similarity score, level of plagiarism, and confidence to users. Furthermore, it contains graphs, charts, and visual representation that aid users in interpreting the outcomes. Users have the ability to export results to various file formats such as TXT, JSON, and PDF. These file formats are helpful in documenting the plagiarism detection outcomes. The module helps in the easy visualization and understanding of results generated from the system.

## Results:



Fig-1: Home Page



Fig-2: Register Page



Fig-3: Uploading Files



Fig-4: Plagiarism Result



Fig-5: AI Detection Checking



Fig-6: AI Detection Result

### 3. DISCUSSION

The idea behind "Intelligent Assignment Plagiarism Detection System Using Machine Learning" is to make the identification process automated. From the review of existing systems, it became evident that the conventional way of detecting plagiarism involved checking for keyword matches and comparing texts. Unfortunately, such methods cannot detect paraphrasing and semantic similarities.

To solve this issues, machine learning and natural language processing techniques will be used. In particular, all uploaded documents will be processed with the help of TF-IDF (Term Frequency–Inverse Document Frequency) and then compared with the help of Cosine Similarity. It will allow finding both exact and partly modified copied content.

At the same time, apart from detecting plagiarized content, the system should identify AI-generated texts based on transformer models. The system will be coded in Python and will use Flask framework for its implementation.

### 4. CONCLUSIONS

In summary, the developed system is efficient in identifying similarity between two documents and checking whether any document contains AI-generated content. This is achieved using NLP and machine learning methods such as cosine similarity, TF-IDF vectorization, and AI detection. The system not only identifies similarity between documents but also classifies the degree of plagiarism in each category, hence offering a detailed output to help users understand the extent of plagiarism. Integration of different components such as user authentication, document uploading, and results visualization has improved the performance and usability of the system. In addition, the ability to generate reports in various formats makes the system highly useful in both academic and professional settings. The inclusion of AI content detection in the system makes it more relevant in today's era since the number of texts created using machines is increasing day by day. The project has effectively applied NLP and machine learning techniques in solving the problem.

### 5. REFERENCES

[1] J. Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries," in *Proc. 1st Instructional Conf. Machine Learning*, 2003, pp. 133–142.

- [2] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.
- [3] T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," in *Proc. ICLR*, 2013.
- [4] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proc. EMNLP*, 2014, pp. 1532–1543.
- [5] T. Kenter and M. de Rijke, "Short Text Similarity with Word Embeddings," in *Proc. CIKM*, 2015, pp. 1411–1420.
- [6] A. Radford et al., "Language Models are Unsupervised Multitask Learners," *OpenAI Technical Report*, 2019.
- [7] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL*, 2019, pp. 4171–4186.
- [8] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," 2019.
- [9] T. Brown et al., "Language Models are Few-Shot Learners," in *Proc. NeurIPS*, 2020.
- [10] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, O'Reilly, 2009.
- [11] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge Univ. Press, 2008.
- [12] K. Papineni et al., "BLEU: a Method for Automatic Evaluation of Machine Translation," in *Proc. ACL*, 2002.
- [13] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., Pearson, 2020.
- [14] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [15] A. Vaswani et al., "Attention Is All You Need," in *Proc. NeurIPS*, 2017.
- [16] Y. LeCun et al., "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE*, 1998.
- [17] T. Joachims, "Text Categorization with Support Vector Machines," in *Proc. ECML*, 1998.
- [18] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
- [19] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. KDD*, 2016.
- [20] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Comput. Surveys*, 2002.
- [21] E. Stamatatos, "Plagiarism Detection Using Character N-grams," in *Proc. PAN Workshop*, 2009.
- [22] M. Potthast et al., "Overview of the 1st International Competition on Plagiarism Detection," in *Proc. PAN*, 2009.
- [23] P. Clough, "Plagiarism in Natural and Programming Languages," *Dept. Computer Science, Univ. Sheffield*, 2000.
- [24] S. Alzahrani, N. Salim, and A. Abraham, "Understanding Plagiarism Linguistic Patterns," *IEEE Trans. Syst. Man Cybern.*, 2012.
- [25] M. Stein et al., "Strategies for Retrieving Plagiarized Documents," in *Proc. SIGIR*, 2007.
- [26] B. Gupta and N. Saxena, "A Survey on Plagiarism Detection Techniques," *Int. J. Comput. Appl.*, 2017.
- [27] S. Kakkonen and E. Sutinen, "Evaluation of Plagiarism Detection Systems," *J. Universal Computer Science*, 2004.
- [28] H. Maurer et al., "Plagiarism—A Survey," *J. Universal Computer Science*, 2006.
- [29] OpenAI, "Detecting AI-Generated Text," *OpenAI Research*, 2023.
- [30] HuggingFace, "Transformers: State-of-the-Art NLP Models," [Online]. Available: <https://huggingface.co>