APPLIED MACHINE LEARNING FOR TEXT ANALYSIS PROJECT REPORT

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

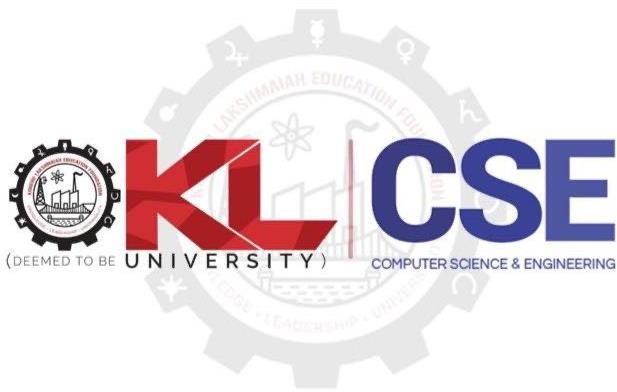
**BACHELOR OF TECHNOLGY SUBMITTED BY**

**B.JAYANTH KUMAR (2300039077)**

**M.MANDEEP CHAKRAVARTHY (2300039108)**

**V. CHAITANYA (2300039116)**

**SK. VALIYA BABJI (2300039117)**

****

**Department of Computer Science and Engineering KONERU LAKSHMAIAH EDUCATION FOUNDATION**

**Green Fields, Vaddeswaram.**

# ACKNOWLEDGEMENTS

We would like to express my sincere gratitude to **Dr. Malladi Ravisankar, Professor**, Department of Computer Science and Engineering (CSE-2), **KL Deemed to be University**, for his valuable guidance, encouragement, and continuous support throughout the completion of my **Applied Machine Learning for Text Analysis Project Report** titled *“Plagiarism Detection Mini-Project.”*

His insightful suggestions and constant motivation greatly contributed to the successful completion of this project.

I would also like to thank my university, **KL Deemed to be University**, for providing a conducive learning environment and encouraging students to undertake practical and research-oriented projects that enhance technical knowledge and hands-on skills in emerging technologies like **Machine Learning** and **Artificial Intelligence**.

# INDEX

|  |  |  |
| --- | --- | --- |
| **S. No** | **Topics** | **Page No.** |
| 1 | ABSTRACT | 3 |
| 2 | INTRODUCTION | 4 |
| 3 | Software Development Life Cycle (SDLC) | 5 |
| 4 | System Architecture and Design | 8 |
| 5 | System Flowcharts | 10 |
| 6 | SETUP AND INSTALLATION | 13 |
| 7 | CODE EXPLANATION  APPENDICES | 14 |
| 8 | GUI IMPLEMENTATION WITH GRADIO | 17 |
| 9 | TESTING | 18 |
| 10 | CONCLUSION AND FUTURE INFERENCES | 19 |
| 11 | REFERENCES | 20 |

ABSTRACT

Plagiarism detection is a fundamental aspect of maintaining academic integrity and promoting originality in written content. This project, titled “Applied Machine Learning for Text Analysis – Plagiarism Detection,” presents an intelligent system designed to automatically identify textual similarities between documents using both traditional and advanced Natural Language Processing (NLP) techniques. The system is implemented in Python and employs a hybrid approach that combines TF-IDF (Term Frequency–Inverse Document Frequency) for lexical similarity assessment with SBERT (Sentence-BERT) for semantic similarity evaluation. This dual-model strategy enables the system to effectively capture both surface-level and contextual similarities between texts.

A user-friendly web-based interface, developed using Gradio, allows users to input suspect and reference texts, perform real-time analysis, and visualize detailed similarity scores. The development process adheres to the Software Development Life Cycle (SDLC) methodology, encompassing key phases such as requirement analysis, system design, implementation, testing, deployment, and maintenance. System architecture diagrams and data flow representations illustrate the logical progression of data—from initial text input and preprocessing to model-based similarity computation and result presentation.

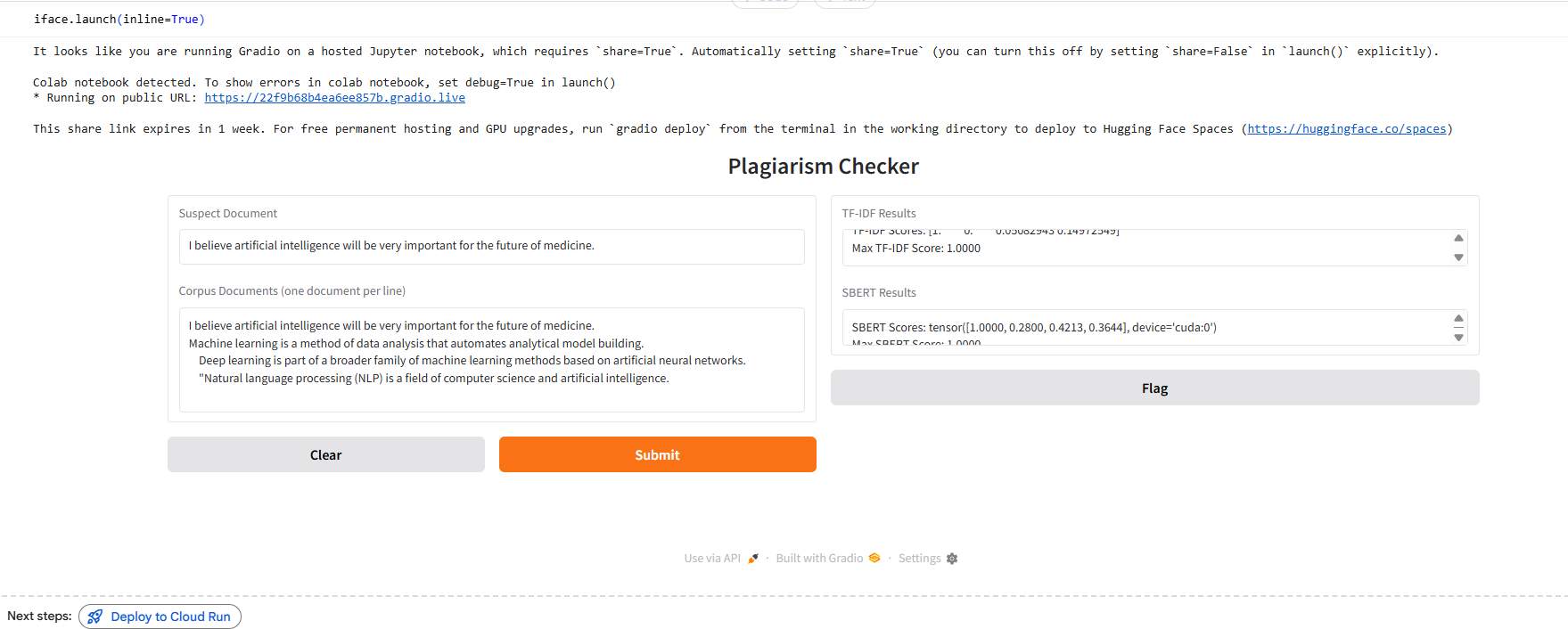
Extensive testing was conducted using various document pairs, including exact copies, paraphrased versions, and unique content, to ensure the accuracy, robustness, and reliability of the system. Experimental results demonstrate that the combined TF-IDF and SBERT approach significantly improves detection performance compared to traditional methods. The integration of semantic similarity modeling enables the detection of subtle rewording and paraphrasing, addressing limitations of earlier lexical-based approaches.

Beyond academic use, the proposed system holds significant potential for real-world applications in content verification, publishing, and online education platforms where originality is essential. By leveraging modular architecture and scalable machine learning models, the system can be extended to process large datasets, detect cross-lingual plagiarism, and integrate with institutional databases or Learning Management Systems (LMS). Future enhancements may include support for document file uploads, API integration, and fine-tuning transformer-based models for domain-specific plagiarism detection. Ultimately, this project demonstrates the transformative potential of machine learning and NLP in ensuring ethical authorship and fostering a culture of originality in the digital age.

PLAGIARISM DETECTION MINI-PROJECT

1. **Introduction**

This document provides a detailed overview of the plagiarism detection mini-project, implemented in Python. The project utilizes both traditional TF-IDF (Term Frequency-Inverse Document Frequency) and advanced SBERT (Sentence-BERT) models to identify potential plagiarism between a suspect document and a corpus of documents. A web-based graphical user interface (GUI) has been developed using the Gradio library to facilitate easy interaction with the tool.

****

**2. Software Development Life Cycle (SDLC)**

**2.1 Introduction to SDLC**

The Software Development Life Cycle (SDLC) is a systematic process used for developing high-quality software that meets or exceeds customer expectations. It defines a structured sequence of stages, including requirement analysis, system design, implementation, testing, deployment, and maintenance.

By applying the SDLC approach to this Plagiarism Detection System using Machine Learning, the development process becomes organized, efficient, and measurable, ensuring reliability and accuracy in the system’s output.

**2.2 SDLC Phases Applied to the Project**

a) Requirement Analysis

This is the initial phase where the system requirements were identified and documented.  
The main requirements for the plagiarism detection system include:

* Detecting similarities between two or more textual documents.
* Supporting traditional and modern NLP models such as TF-IDF and SBERT.
* Providing a user-friendly interface using Gradio for interaction.
* Delivering fast and accurate similarity scores.

Both functional and non-functional requirements were considered:

* Functional: Input text processing, similarity calculation, and result visualization.
* Non-functional: Performance efficiency, accuracy, and ease of use.

b) System Design

The design phase transforms requirements into a blueprint for implementation. The architecture was designed using modular components:

1. Data Input Module: Takes the suspect and reference documents.
2. Preprocessing Module: Tokenizes text, removes stop words, and prepares the data.
3. Feature Extraction Module: Generates embeddings using TF-IDF and SBERT.
4. Similarity Engine: Computes cosine similarity between documents.
5. GUI Module: Displays similarity results through a Gradio-based web interface.

Each module interacts sequentially, ensuring a clean and maintainable structure.

c) Implementation

The system was implemented using Python in a Google Colab environment. The following tools and libraries were used:

* NLTK: For preprocessing and stop-word removal.
* Scikit-learn: For implementing TF-IDF and cosine similarity.
* Sentence-Transformers: For generating SBERT embeddings.
* Gradio: For creating the user interface.

Code modules were developed independently and integrated gradually for better debugging and testing.

d) Testing

Testing was performed to ensure correctness and reliability. Various cases were evaluated:

* Comparing identical documents (expected: high similarity).
* Comparing paraphrased documents (moderate similarity).
* Comparing completely different documents (low similarity).

Manual and automated test cases were created to verify the accuracy of similarity outputs. Bugs identified during testing were fixed iteratively.

e) Deployment

The Gradio web interface enables simple and lightweight deployment.  
Users can access the plagiarism detection tool through a local URL or via the public Gradio link.  
Since it runs in a Colab notebook, it does not require local installation, improving accessibility for students and researchers.

f) Maintenance

The project’s modular design allows for easy updates. Future maintenance may include:

* Adding file upload options (PDF, DOCX).
* Integrating new embedding models (e.g., BERT variants).
* Enhancing the GUI for improved visualization.

Continuous feedback and retraining of models can maintain accuracy as language data evolves.

**2.3 Advantages of Using SDLC**

* Provides a clear framework for development.
* Enhances project visibility and tracking.
* Facilitates better testing and maintenance.
* Ensures systematic quality control and documentation.

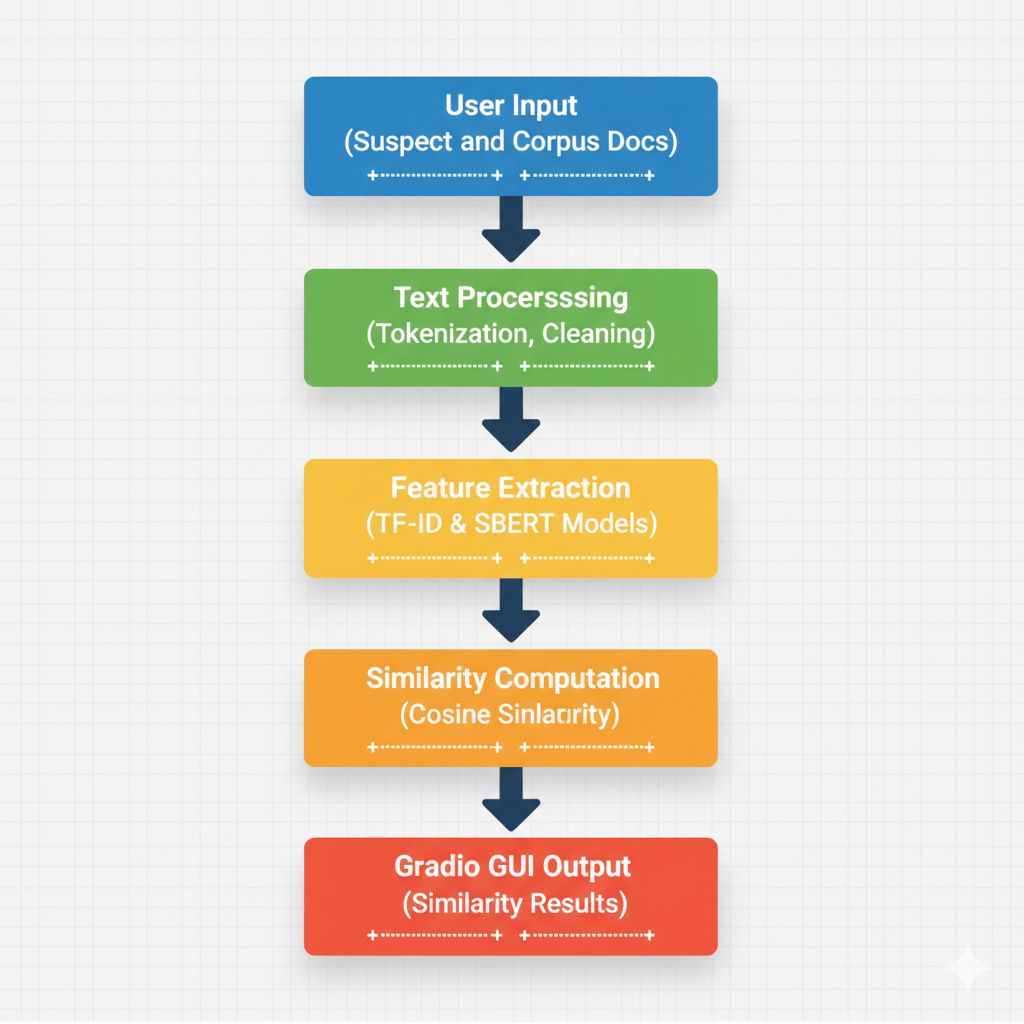
**3. System Architecture and Design**

**3.1 System Architecture Overview**

The system follows a modular architecture, consisting of five main layers:

1. User Layer: Allows users to input text through a Gradio interface.
2. Processing Layer: Handles text preprocessing using NLTK.
3. Feature Extraction Layer: Converts processed text into numerical vectors via TF-IDF or SBERT.
4. Similarity Computation Layer: Calculates cosine similarity to find matches.
5. Output Layer: Displays similarity scores and potential plagiarism levels.

**3.2 System Architecture Diagram**

****

**3.3 Data Flow Diagram (DFD)**

Level 0: High-Level DFD

User → [Plagiarism Detection System] → Similarity Report

Level 1: Detailed DFD

User Input → Preprocessing → Feature Extraction → Similarity Calculation → GUI Display

Each stage represents a logical operation transforming raw text into interpretable results.

**3.4 Use Case Diagram**

* Actor: User
* Use Cases: Enter document, check plagiarism, view similarity result.
* System: Plagiarism Detection Tool

Relations:

* User interacts with the GUI.
* The system processes input and returns a similarity score.

**3.5 Sequence of Operations**

1. User enters suspect and corpus text.
2. System cleans and preprocesses input.
3. Feature extraction occurs using TF-IDF and SBERT.
4. Cosine similarity is computed.
5. Results are displayed on the GUI interface.

**4. System Flowcharts**

**4.1 Overall Flowchart**

A diagram of a computer process

AI-generated content may be incorrect.

**4.2 TF-IDF Workflow Flowchart**

A diagram of a software

AI-generated content may be incorrect.

**4.3 SBERT Workflow Flowchart**

**A diagram of a process

AI-generated content may be incorrect.**

**4.4 Explanation**

These flowcharts visualize the logic of the plagiarism detection system.  
TF-IDF focuses on lexical similarity, while SBERT captures semantic similarity using deep learning models.  
The combination of both approaches improves detection accuracy and robustness.

**5. Setup and Installation**

This section outlines the necessary steps to set up and run the project in a Google Colab environment.

2.1 Prerequisites

* Google Account to access Google Colab.

2.2 Installing Libraries

The project requires the following Python libraries:

* nltk: For natural language processing tasks like tokenization and stop word removal.
* scikit-learn: For implementing the TF-IDF model and cosine similarity.
* sentence-transformers: For implementing the SBERT model.
* gradio: For building the web-based GUI.

These libraries can be installed using pip in a code cell within your Colab notebook.

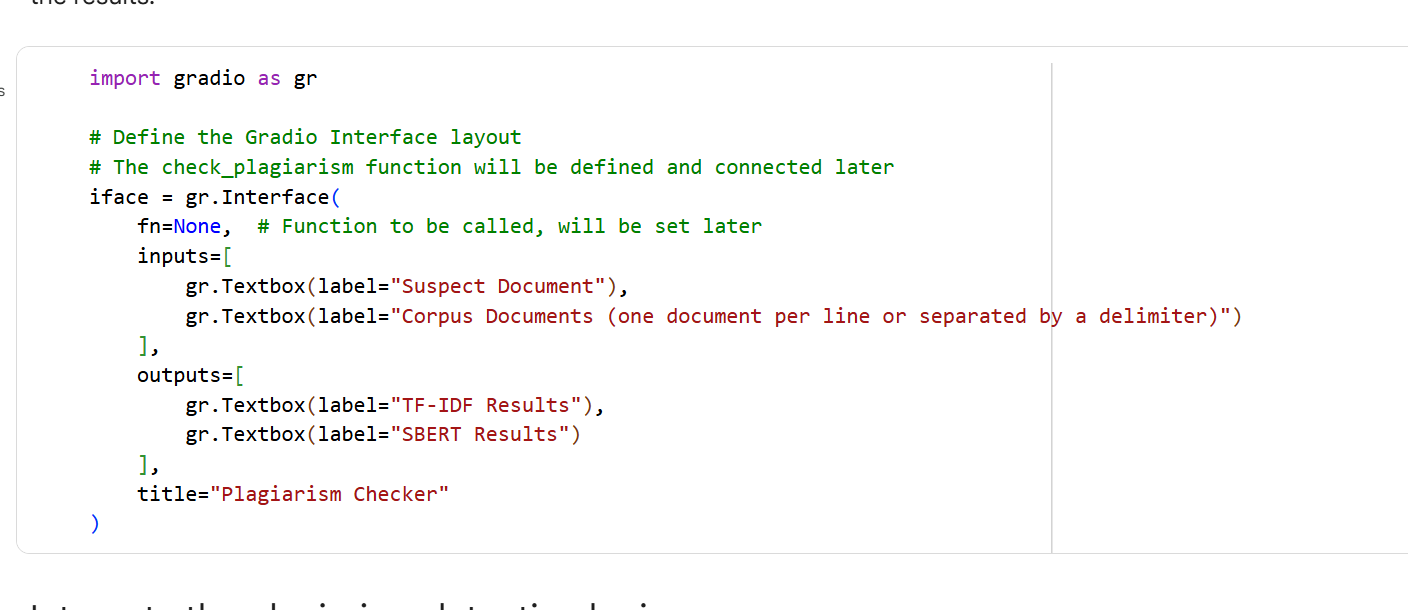
****

1. **Code Explanation**

This section explains the core components of the Python code used for plagiarism detection.

6.1 Importing Libraries and Helper Functions

This part of the code imports all necessary libraries and defines helper functions, such as text preprocessing for TF-IDF.

****

6.2 TF-IDF Model

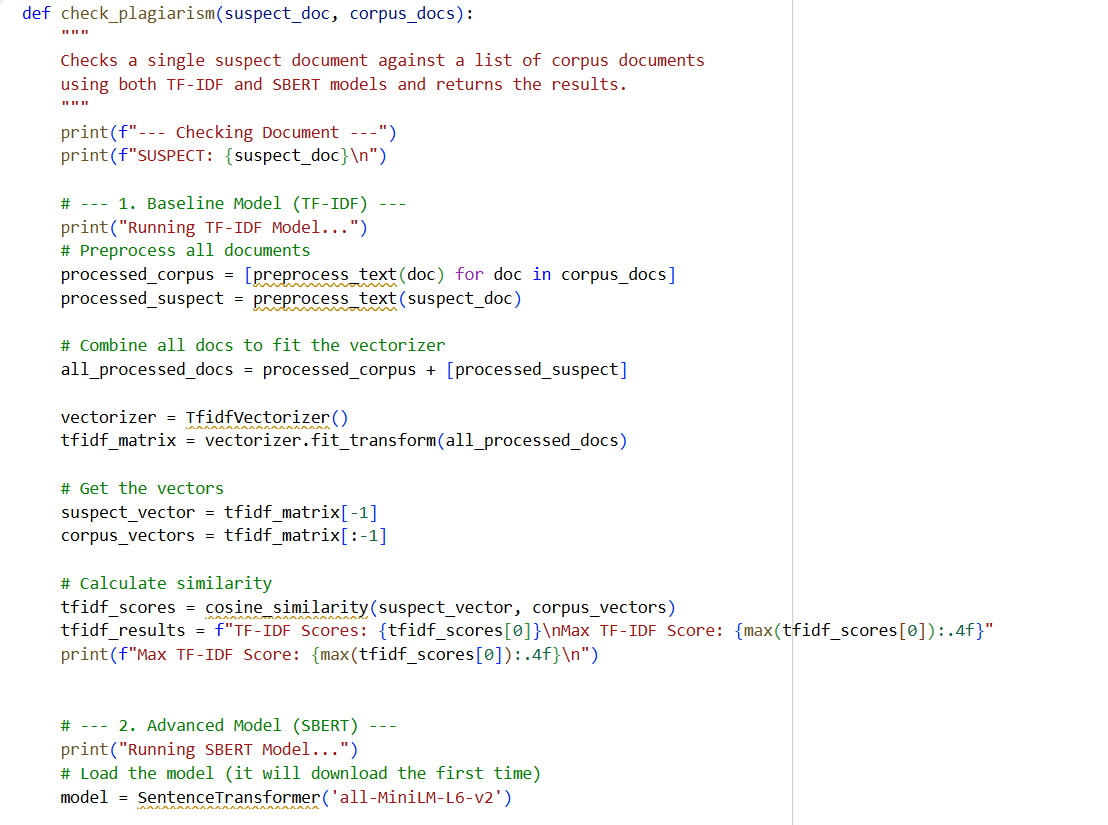
Explain how the TF-IDF model is used to represent documents as vectors and how cosine similarity is calculated to find similarities between the suspect document and corpus documents.

6.3 SBERT Model

Explain how the SBERT model is used to generate sentence embeddings and how cosine similarity is applied to measure the semantic similarity between documents.

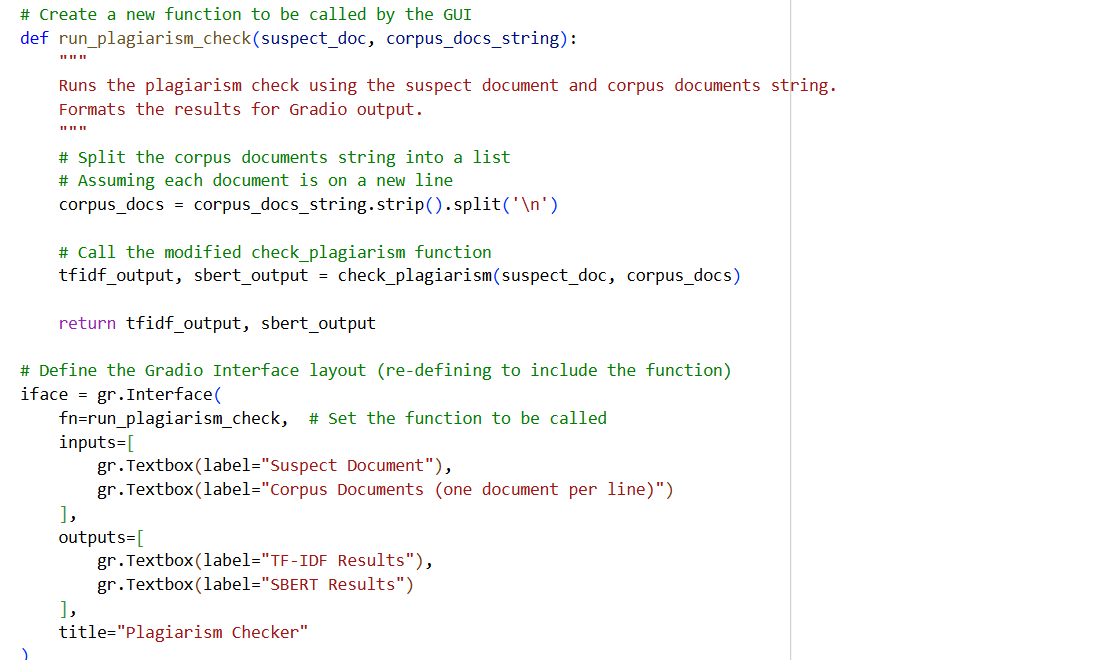
6.4 check\_plagiarism Function

Detail the check\_plagiarism function, which takes the suspect document and corpus documents as input, applies both TF-IDF and SBERT models, and returns the similarity results.

****

6.5 run\_plagiarism\_check Function

Explain the run\_plagiarism\_check function, which acts as an intermediary between the Gradio GUI and the check\_plagiarism function. It handles input formatting from the GUI and output formatting for the GUI.

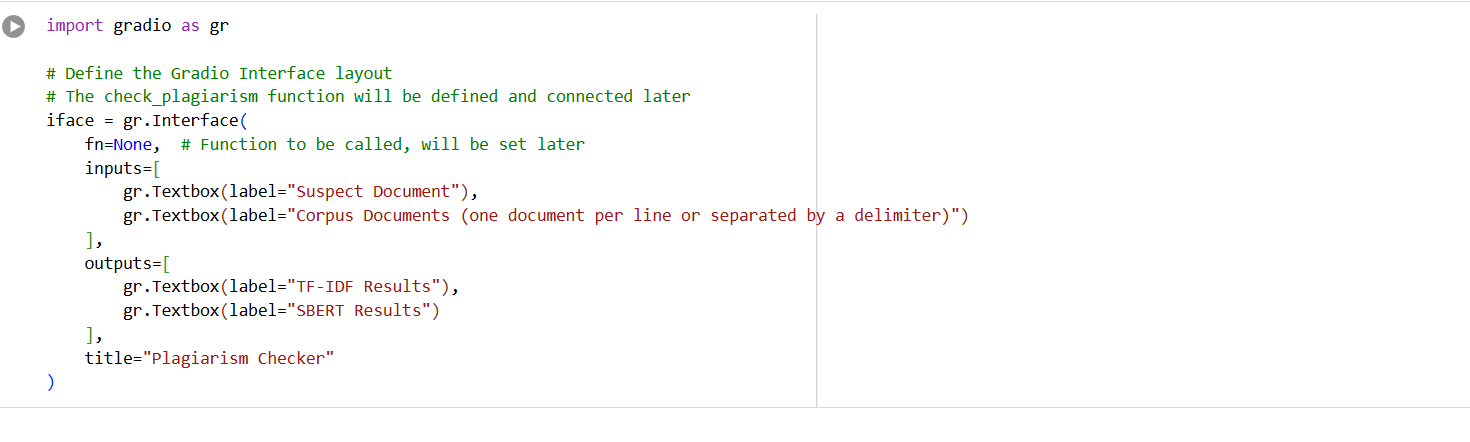
****

1. **GUI Implementation with Gradio**

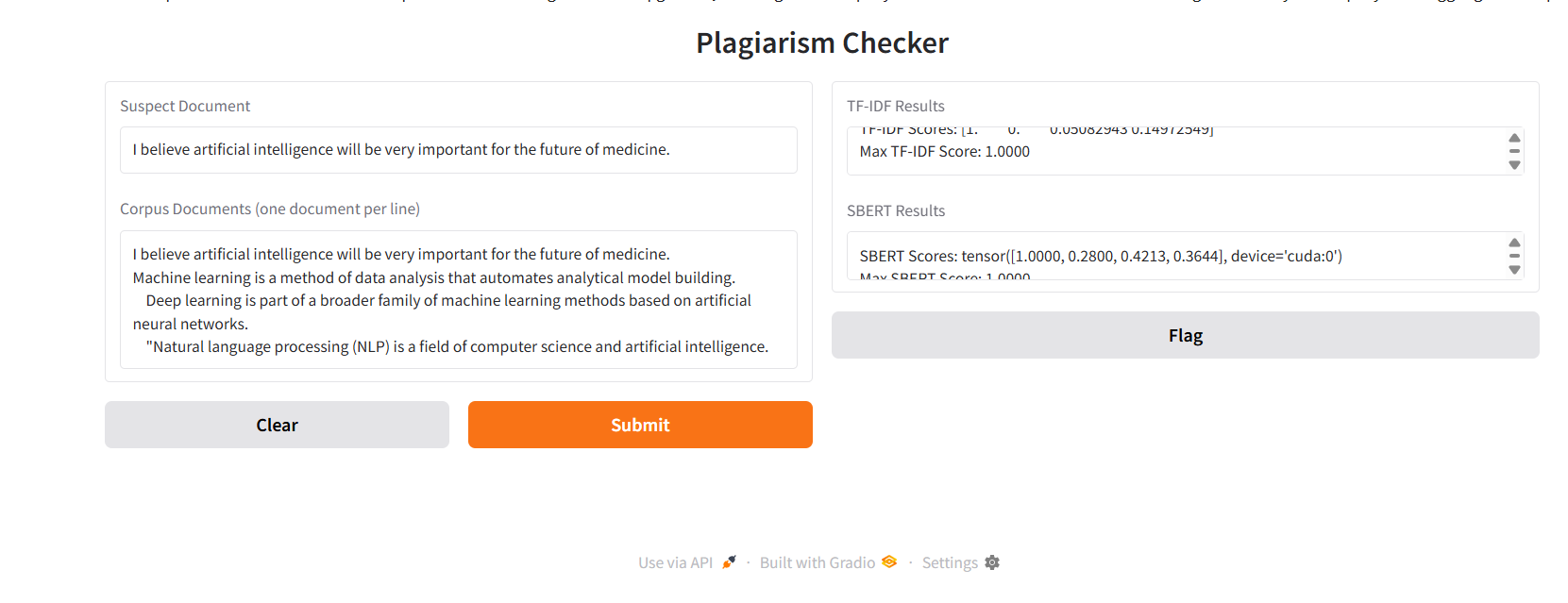
This section describes how the Gradio library was used to create the interactive GUI.

4.1 Designing the Interface Layout

Explain how the gradio.Interface class was used to define the layout of the GUI, including the input textboxes for the suspect document and corpus documents, and the output textboxes for the TF-IDF and SBERT results.

****4.2 Launching the GUI

Show the code used to launch the Gradio interface, making it accessible via a local or public URL.

****

1. **Testing**

Describe the testing process used to verify the functionality of the plagiarism detection tool and the GUI.

5.1 Test Cases

Outline the different test cases used, such as:

* Checking an original document against the corpus.
* Checking a direct copy of a corpus document.
* Checking a paraphrased version of a corpus document.

****

**9. Conclusion and Future Improvements**

Summarize the key findings of the project and discuss potential areas for future improvement, such as:

* Adding support for different input formats (file uploads).
* Implementing progress indicators for long processing times.
* Exploring other plagiarism detection techniques.
* Improving the user interface and user experience.

**10 . References**

1. Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks.* Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP).  
   Available at: [https://arxiv.org/abs/1908.10084](https://arxiv.org/abs/1908.10084?utm_source=chatgpt.com)
2. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space.* arXiv preprint arXiv:1301.3781.  
   Available at: https://arxiv.org/abs/1301.3781
3. Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.  
   Available at: https://nlp.stanford.edu/IR-book/
4. “TF–IDF (Term Frequency–Inverse Document Frequency).” Wikipedia, Wikimedia Foundation.  
   Available at: [https://en.wikipedia.org/wiki/Tf%E2%80%93idf](https://en.wikipedia.org/wiki/Tf%E2%80%93idf?utm_source=chatgpt.com)
5. Simha, A. (2021, October 6). *Understanding TF-IDF for Machine Learning.* Capital One Tech Blog.  
   Available at: [https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/](https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/?utm_source=chatgpt.com)
6. “Understanding TF-IDF (Term Frequency-Inverse Document Frequency).” *GeeksforGeeks*, August 2025.  
   Available at: [https://www.geeksforgeeks.org/machine-learning/understanding-tf-idf-term-frequency-inverse-document-frequency/](https://www.geeksforgeeks.org/machine-learning/understanding-tf-idf-term-frequency-inverse-document-frequency/?utm_source=chatgpt.com)
7. GitHub – UKPLab. *Sentence-Transformers: State-of-the-Art Text Embeddings.*  
   Available at: [https://github.com/UKPLab/sentence-transformers](https://github.com/UKPLab/sentence-transformers?utm_source=chatgpt.com)
8. *NLTK: Natural Language Toolkit – Official Documentation.*  
   Available at: <https://www.nltk.org/>
9. *Scikit-learn: Machine Learning in Python – Official Documentation.*  
   Available at: <https://scikit-learn.org/stable/>
10. *Gradio: Build & Share Machine Learning Demos – Official Documentation.*  
    Available at: <https://www.gradio.app/docs>
11. *Sentence-Transformers: Documentation & API Reference.*  
    Available at: <https://www.sbert.net/>
12. *Kaggle Plagiarism Detection Dataset.*  
    Available at: <https://www.kaggle.com/datasets>
13. *Google Research: Text Similarity Datasets.*  
    Available at: <https://research.google/resources/datasets/>
14. *Analytics Vidhya – A Guide to TF-IDF and Word Embeddings.*  
    Available at: <https://www.analyticsvidhya.com/>
15. TutorialsPoint. *Python NLP and Text Mining Tutorial.*  
    Available at: https://www.tutorialspoint.com/python\_text\_mining/index.htm