DEVELOPING A CLASSIFICATION MODEL AND SEMANTIC SIMILARITY DETECTION OF SYSTEM AND SOFTWARE REQUIREMENTS USING SENTENCE TRANSFORMERS

Dissertation Phase 2 Report

Submitted by

Ms. V R MANASI MANASAN

REG NO : TKM22MEAI16

to

In partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY
IN
Artificial Intelligence

Under the guidance of
Dr. ADARSH S

CENTRE FOR ARTIFICIAL INTELLIGENCE

TKM College of Engineering, Kollam

JUNE 2024

Thangal Kunju Musaliar College of Engineering
Centre for Artificial Intelligence

C E R T I F I C A T E

This is to certify that, this report titled ***DEVELOPING A CLASSIFICATION MODEL AND SEMANTIC SIMILARITY DETECTION OF SYSTEM AND SOFTWARE REQUIREMENTS USING SENTENCE TRANSFORMERS*** is a bonafide record of the **Dissertion Phase-2 report** presented by **V R MANASI MANASAN (TKM22MEAI16)**, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, **M. Tech in Artificial Intelligence** in **APJ Abdul Kalam University**.

Internal Supervisor
Dr. Adarsh S
Professor
Dept. of Civil Engg.
TKMCE

External Supervisor
Yuvavani Priya S
Senior Technical Lead
Continental Automotives
Banglore

Project Coordinator
Dr. Sumod Sundar
Associate Professor
Centre for AI
TKMCE

Head of the Department
Dr. Imthias Ahamed T P
Professor
Centre for AI
TKMCE

ACKNOWLEDGEMENT

A successful project is a fruitful culmination of efforts by many people, some directly involved and some others indirectly, by providing support and encouragement. Firstly I would like to thank the almighty for giving me the wisdom and grace for making my project a memorable one. I thank him for steering me to the shore of fulfillment under his protective wings.

I express my sincere gratitude to **Dr. T A Shahul Hameed**, Principal of T.K.M College of Engineering for allowing me to present my project. I would like to thank **Dr. Imthias Ahamed T P**, Professor and Head of the Department, Centre for Artificial Intelligence, TKM College of Engineering, Kollam, for his constant support and encouragement throughout the work.

With a profound sense of gratitude, I would like to express my heartfelt thanks to my Internal supervisor, **Dr.Adarsh S**, Professor, Department of Civil Engineering, TKM College of Engineering, Kollam and Project Coordinator, **Dr.Sumod Sundar**, Associate Professor, Centre for Artificial Intelligence(AI), TKM College of Engineering for their expert guidance, cooperation, and immense encouragement. I also extend my thanks to the entire faculty and staff members of the Centre for AI, TKMCE, who have encouraged me throughout this work.

I express my deepest gratitude to **Ms. Yuvavani Priya S**, Senior Technical Lead, Continental Automotive Components Pvt.Ltd, Banglore,for mentoring me with the project idea and giving me the opportunity to work on it. Sincere thanks to **Ms.Sanjeena Menezes**, Senior Manager,Continental Automotive Components Pvt.Ltd, Banglore, for her guidance throughout the project. I also extend my thanks to everyone at Continental Automotive Components Pvt.Ltd, Banglore, for their support and help throughout this project.

I also express my thanks to my loving parents and friends, for their support and encouragement in the successful completion of this work.

V R MANASI MANASAN

Abstract

Reading and understanding similar type of requirements using human interventions can be a laborious and time-consuming task. However, the complexity of natural language makes it difficult to accurately identify semantic similarities, which makes more difficulty to the task. The majority of algorithms for detecting similarities rely on matching words to words, paragraphs to paragraphs, or the entire page to the matching.. In this project, a novel approach is proposed which uses large language model (LLM) such as Sentence Transformers for detecting semantically similar requirement.The Sentence Transformer models used in this project includes all-MiniLM-L6-v2 and paraphrase-MiniLM-L6-v2. The dataset consists of 10,500 software and system requirements vital for SCANIA's (German Company) operations. The aim of the project is to find semantically similar system and software requirements and categorize those requirements based on similarity score and compare the performance of Sentence Transformers with traditional algorithms such as Word2vec and TF-IDF Vectorizer to find semantically similar requirements. In addition to it a GUI(Graphical User Interface) is built so that user can interact easily and find similar requirements. The project employs several measures to evaluate the performance of the model, including F1 score, precision , accuracy, euclidian distance , cosine similarity.Among 'all-MiniLM-L6-v2 ' and 'paraphrase-MiniLM-L6-v2', 'all-MiniLM-L6-v2' outperformed better with an accuracy of 92%,precision of 95%,Recall of 82% and F1 score of 88%. Ultimately, the project aspires to revolutionize requirement analysis, driving efficiency and productivity in software and system development processes through the seamless integration of cutting-edge technologies and intuitive interfaces.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sentence similarity measures are becoming increasingly more important in text-related research and other application areas. Simply reading and understanding similar type of requirements from different companies using human interventions can be a laborious and time-consuming task.However, the complexity of natural language makes it difficult to accurately identify semantic similarities, which makes more difficulty to the task.The large volume of documentation generated by numerous companies adds to the complexity, which makes manual categorization impractical and susceptible to error.

This research is important because it addresses critical challenges within the realm of software engineering and requirements specification. The detection of semantic similarity in requirements plays a pivotal role in enhancing communication and collaboration among team members and stakeholders, fostering a clearer understanding of project goals. By identifying similarities, the research enables the reuse of requirements across different projects, thereby optimizing resource utilization and reducing redundancy. Moreover, the ability to analyze the impact of changes on interconnected requirements aids in effective change management. The research contributes to the establishment of consistency and completeness in requirements, mitigating the risk of oversights and ensuring a comprehensive specifications document.With the advancements in deep learning and natural language processing (NLP) techniques, it has become possible to recognize the underlying semantically similar requirements present between the data and helps to categorize the requirements according to similarity level.

The aim of the work is to extract the necessary information (requirement) from the requirement document as well as the DNG (Requirement Management Database), convert those extracted information into a suitable format (xlsx), find semantically similar requirements and categorize those requirement according to similarity level.The method proposed an automated methods, specifically using large language model such as Sentence Transformers for detecting semantically similar requirements.

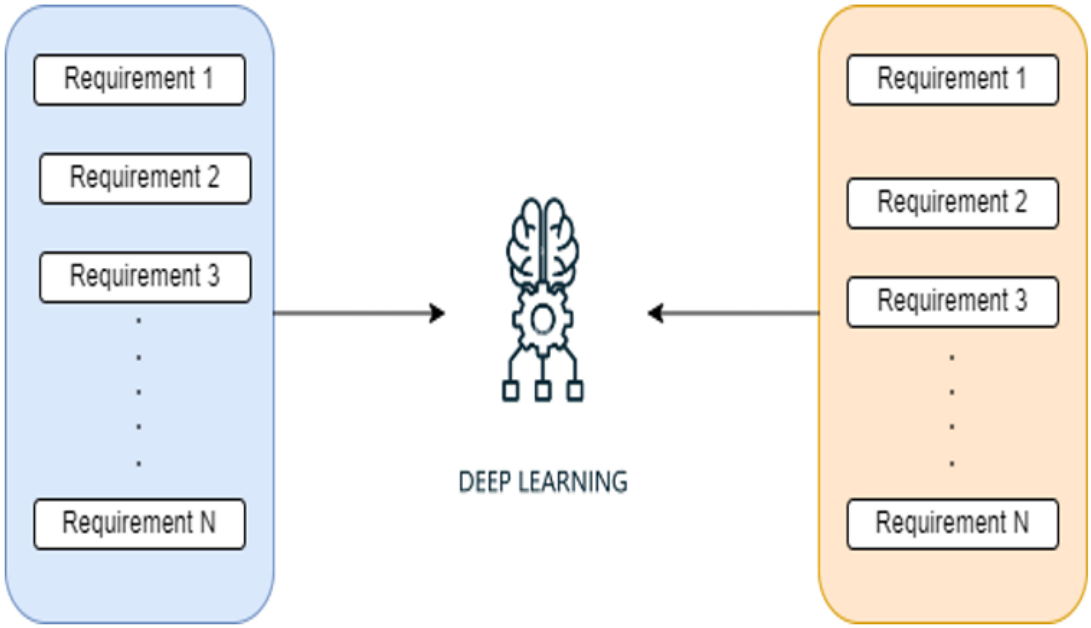


Figure 1.1: Problem Statement

Several studies have been conducted on detecting semantic similarity between texts, documents, articles, and paragraphs. However, there is a lack of research in finding semantically similar system and software requirements in the automotive industry. Notably, there is a gap in the integration of natural language processing (NLP) and requirements engineering (RE). While significant advancements have been made in both NLP and RE individually, there is a notable absence of studies that effectively integrate these two domains. Existing approaches often focus solely on linguistic analysis or formal requirement specifications, neglecting the potential synergies between these fields. Bridging this gap could lead to novel approaches that enhance requirement similarity analysis in the automotive industry by combining NLP techniques for understanding natural language texts with RE methodologies for defining and managing requirements. Additionally, there is limited research on the specific methods and techniques for extracting and detecting semantic similarity among requirements, particularly in the context of the automotive industry. Current studies often focus on general-purpose text similarity measures, which may not capture the nuances and complexities of automotive requirements, including domain-specific terminology and context. Dedicated research efforts are needed to explore specialized algorithms and models tailored to the unique characteristics of automotive requirements, such as safety-critical features, regulatory compliance, and system integration challenges. The lack of attention to semantic similarity detection in automotive requirements hampers the development of automated tools and systems capable of effectively analyzing, managing, and evolving large-scale automotive systems and software projects. Addressing this gap requires interdisciplinary collaboration between researchers in NLP, RE, and automotive engineering to develop innovative approaches that combine

linguistic analysis with domain-specific knowledge and requirements engineering principles.

This project is expected to produce results that are not limited to academics.The practical applications of detecting semantic similarity in requirements are manifold and offer tangible benefits across various stages of the software development life-cycle. By identifying similarities among requirements, this capability facilitates more effective requirements prioritization, allowing development teams to focus on common functionalities and streamline project planning. Moreover, it enables the efficient reuse of requirements across projects or within the same project, promoting consistency and optimizing resource allocation.

1.1 Objectives

- To implement Sentence Transformers to extract sentence embeddings and identify semantically similar requirements between documents from System and Software Requirements.
- To compare the performance of Sentence Transformers with traditional algorithms such as Word2Vec,TF-IDF Vectorizer to find semantically similar requirements.
- To build a Graphical User Interface to interact and find semantically similar requirements.

Chapter 2

Literature Survey

In the associated embedding space, words that are semantically comparable also have similar embedding vectors. Additionally, if a text can be thought of as a bag of words, word embeddings can be utilized to extract the text's embedding.Words in a collection or corpus can be given weights based on their importance to the document using TF-IDF (Term Frequency–Inverse Document Frequency).The number of papers in the corpus that contain a word offsets the TF-IDF value, which rises proportionately to the number of times the term appears in the document. Using the embeddings of the words that make up a text, TF-IDF can be utilized as a weighting approach to get the embedding matching to the text. Word embeddings are used by Kusner et al. [5] to introduce Word Mover's Distance (WMD), a distance function that determines how similar two texts are. Greater frequency of a word signifies greater importance. It is assumed that the text documents are represented as normalised bag-of-words vectors, d Rn for a finite size vocabulary of n words. When utilizing Euclidean distance to calculate the travel cost between two words in a semantic embedding space, the semantic similarity between individual word pairs can be included. The least weighted cumulative cost needed to transfer every word from the first text to the second is known as the net distance between two texts. Word ordering is not taken into account by WMD because it uses a bag-of-words technique and requires a lot of processing time.Even the improved version has more complexity, whereas the vanilla version's complexity is O(p3 log p).Using word embeddings of the constituent words, Arora et al. [5] presented the smooth inverse frequency (SIF) method for generating text embeddings.

In next set of paper[2] illustrates Weights that were applied to the individual words in an input text, and the resulting single vector is obtained by taking the average. The weight allotted to the word embedding is aa+p(w), where p(w) is the predicted frequency of the word in the reference corpus and an is a parameter that is typically set to 0.001[1]. For a given set of components, the major components of the resulting embeddings are calculated. To eliminate the variation in frequency and syntax that is less significant from a semantic standpoint, the projection of the sentence embeddings on their first principal component is deducted from the original embeddings. SIF downgrades words that appear frequently and is also fairly robust to the weighting method; that is, performance is not negatively impacted by employing word frequencies estimated from various corpora. Near-best outcomes can be obtained by varying the parameters widely, and a larger range can yield a notable improvement above the unweighted average. A drawback of the algorithm is its slowness. Additionally,

since it uses a bag of words approach, it creates the sentence embedding without considering the sentence's word order. Neural networks that have been supervisedly trained to build phrase embeddings with a focus on word order are known as pre-trained encoders. With regard to sentences, they seek to fulfill the same function as Word2vec and GloVe. Pre-trained encoders are designed to learn on a variety of supervised and unsupervised tasks in order to extract as much semantic information as feasible.Words with similar semantic meanings also have similar embedding vectors in the corresponding embedding space.Additionally, word embeddings can be used to extract the embedding of a text if it can be conceptualized as a bag of words.

The word embedding suggested in paper [5] is TF-IDF,its value increases in direct proportion to the number of times a term appears in a document, and is offset by the number of papers in the corpus that contain that word.TF-IDF (Term Frequency–Inverse Document Frequency) is a tool for assigning word weights that considers a word's relevance inside a document within a corpus or collection. TF-IDF can be used as a weighting technique to obtain the embedding matching to the text by using the word embeddings.The distance function known as Word Mover's Distance was initially introduced by Kusner et al. [3].A method that was first presented by Buscaldi et al.[1] combines two modules: one that uses WordNet and a concept similarity measure to determine the similarity between concepts in the sentences, and another that determines the similarity between sentences based on N-grams. The automatic and manual similarity results show a good correlation.A system called UKP (Ubiquitous Knowledge Processing) was introduced by Bar et al. [2]. It utilizes a basic log-linear regression model based on training data that combines various text similarity measures, including string and semantic similarity, as well as measures pertaining to structure, style, and text expansion mechanisms.The final UKP model, which generated rather acceptable correlation findings, is a log-linear mixture of roughly twenty characteristics out of the 300 potential features that could have been incorporated.Aggarwal et al.[4] proposed a method that combines a corpus-based semantic relatedness measure over the complete sentence with knowledge-based semantic similarity scores computed for the terms sharing the same syntactic roles in both sentences.

One such way of determining semantic similarity between documents within large collections, specifically for an Arabic search engine. It investigates the use of the MapReduce framework for managing distributed processing and calculating semantic similarity. The study reviews various state-of-the-art approaches for computing document similarity and proposes a novel method based on a parallel algorithm using MapReduce and WordNet, following a translation phase. This approach aims to improve the retrieval of relevant documents in response to Arabic queries.[6].The proposed method leverages the MapReduce programming model to handle big data efficiently and employs WordNet for semantic similarity measures post-translation. The experimental results demonstrate the technique's efficiency and performance, showing its effectiveness in detecting relevant documents. The research highlights the potential of integrating advanced semantic similarity measures with distributed processing frameworks to enhance the accuracy and relevance of search engine results for Arabic language queries.

Slimani [8] provides an overview of existing semantic similarity methods, highlighting

approaches based on structure, information content, and features. These methods quantify similarity between concepts using various metrics, with one prominent example being the Path length-based measure. This approach calculates similarity by measuring the path distance that separates two concepts in a hierarchical structure. The similarity degree is determined by the shortest path length, indicating that closer concepts have higher similarity. The Path length-based measure focuses on the direct distance, emphasizing the shortest route between concepts to assess their relatedness.

In addition to the Path length-based measure, the Depth relative measure is another method used to determine semantic similarity. While it also relies on the shortest path principle, it incorporates the depth of the concepts within the hierarchy. This means that the measure takes into account not just the direct distance but also the hierarchical level at which the concepts reside. By considering depth, this approach provides a more nuanced similarity assessment, recognizing that concepts located deeper in the hierarchy might be more specific and thus have a different similarity context compared to those higher up. These methods collectively enhance the ability to quantify and understand semantic relationships between concepts, aiding in various applications such as information retrieval and knowledge management.In recent studies, researchers have explored methods that combine corpus-based semantic relatedness measures with knowledge-based semantic similarity scores to enhance the understanding of text relationships. Aggarwal et al. [4] proposed a novel approach that integrates a corpus-based semantic relatedness measure computed over entire sentences with knowledge-based semantic similarity scores derived from terms sharing similar syntactic roles in both sentences. This hybrid method leverages both the statistical patterns present in large text corpora and the structured semantic information encoded in knowledge bases. By combining these two sources of information, the approach aims to capture a more comprehensive understanding of text similarity, incorporating both lexical and conceptual similarities. This integrative approach shows promise in improving the accuracy of text similarity assessments, offering a nuanced perspective that goes beyond surface-level lexical overlaps.

Advancements in text similarity modeling have also been propelled by the integration of diverse features and techniques within unified frameworks. Bar et al. [9] introduced the UKP (Ubiquitous Knowledge Processing) system, which employs a comprehensive log-linear regression model trained on various text similarity measures. These measures encompass not only lexical and semantic similarities but also structural, stylistic, and text expansion characteristics. Through the fusion of over twenty distinctive features from a pool of 300 potential attributes, the UKP model achieves notable correlation outcomes. This amalgamation of diverse features underscores the importance of holistic text representation, accommodating the multifaceted nature of language comprehension. As such, these integrated systems offer a glimpse into the future of text similarity assessment, where nuanced combinations of features yield richer insights into textual relationships[8].

There is an innovative mechanism for novelty detection to enhance current web crawlers, addressing the issue of redundant information.[7] The approach involves several key steps: summarizing the text using an ontology to capture essential information, calculating semantic similarity with WordNet 3.0, generating a hash value of the document using the winnowing algorithm, and comparing hash values using the Dice coefficient to measure similarity. Based on a similarity threshold, documents are classified as novel or redundant. Implemented with

SQL as the backend and Visual Studio 2012 as the frontend, the results show that this approach reduces memory consumption and decreases the number of resultant documents, saving user time. This method can also be integrated with search engines like Google, Yahoo, Bing, and AltaVista to filter out redundant documents, enhancing web search efficiency.In the realm of natural language processing, advancements continue to evolve toward more nuanced and contextually aware techniques for understanding text similarity. One such approach, introduced by Buscaldi et al. [10], integrates multiple modules to assess similarity at both the conceptual and syntactic levels. By leveraging resources like WordNet for concept similarity and N-grams for syntactic analysis, this method encapsulates a broader spectrum of linguistic cues. Automatic evaluations alongside manual assessments have demonstrated promising correlations, indicating the effectiveness of this multifaceted approach. As text analysis demands deeper comprehension beyond surface-level overlaps, these integrated methods pave the way for more sophisticated text understanding systems capable of capturing intricate linguistic nuances.

Chapter 3

Methodology

3.1 NLP Pipeline

Figure 3.1 shows the proposed NLP Pipeline.

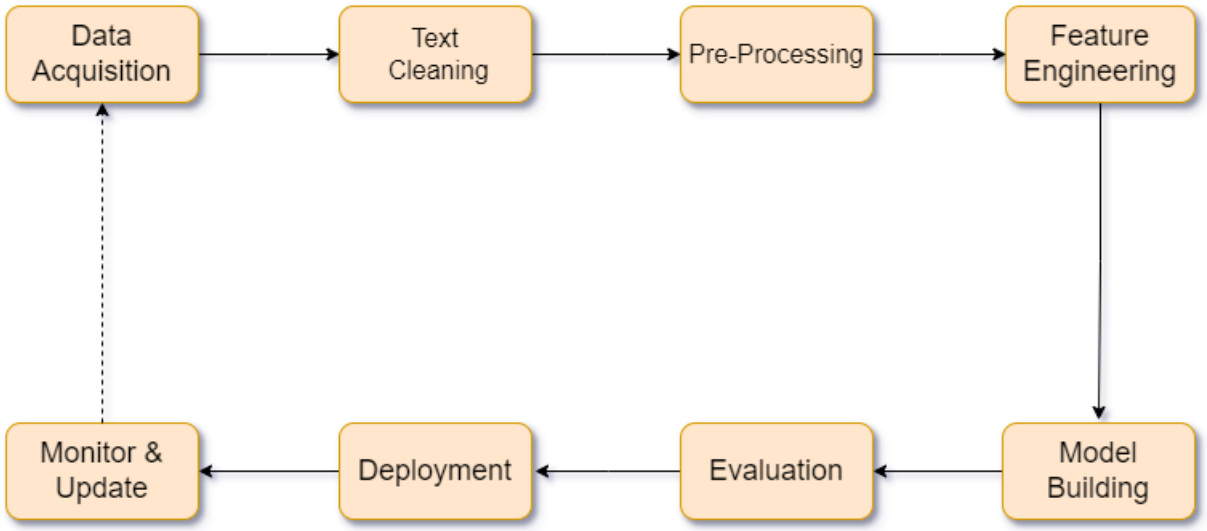


Figure 3.1: NLP Pipeline

The exploration of semantic similarity detection using Sentence Transformers commences with the careful collection of textual data from two distinct sources: the DNG (Requirement Database) and the essential requirements of the Volkswagen Group. The DNG document, housing a substantial volume of 10,500 requirements, presents a rich repository of software requirements alongside their corresponding requirement IDs. Concurrently, the Volkswagen Group's essential requirements outline fundamental specifications crucial for the automotive industry. These documents serve as the primary raw data for subsequent semantic similarity analysis. The diverse origins of the textual data, their integration provides a comprehensive and varied dataset, enriching the analysis and ensuring a robust evaluation of semantic similarity across different domains.Following data collection, the gathered textual data undergoes rigorous preprocessing to ensure uniformity and cleanliness. This preprocessing stage

is pivotal for standardizing the format of the textual data, making it amenable to subsequent analysis. Tasks such as tokenization, which breaks down text into individual words or tokens, and character normalization, ensuring consistency in text encoding, are performed meticulously. Through these preprocessing steps, potential inconsistencies or irregularities in the textual data are addressed, laying the groundwork for more effective analysis downstream. By establishing a consistent and standardized format, preprocessing enhances the quality and reliability of the data, facilitating more accurate semantic similarity analysis.

The vectorization stage constitutes a critical component of the semantic similarity analysis pipeline, wherein Sentence Transformers are employed to generate embeddings for the textual data. Sentence Transformers are specialized models trained to convert sentences or paragraphs into numerical representations known as embeddings. These embeddings capture the semantic content of the sentences, enabling subsequent mathematical operations and comparisons essential for semantic similarity analysis. By representing text numerically, Sentence Transformers facilitate the exploration of semantic relationships between requirements, providing insights into their similarities and differences. This stage plays a pivotal role in transforming the textual data into a format conducive to computational analysis, thereby laying the foundation for more sophisticated semantic similarity analysis techniques.

Subsequent to vectorization, the process may incorporate feature engineering to refine the model's performance for the specific task at hand. Feature engineering involves selecting or transforming features in the data to enhance the performance of machine learning models. In the context of semantic similarity analysis, this may entail adapting embeddings or incorporating additional information, such as metadata associated with the requirements. By tailoring the features to the semantic similarity task, the model can better capture the nuances within the requirements data, thus improving the accuracy and efficacy of similarity analysis. Through meticulous data preparation and feature engineering, this methodology aims to unlock deeper insights into the semantic relationships between requirements, ultimately contributing to more informed decision-making in software and automotive engineering domains.

3.2 Proposed Methodology

The proposed methodology for identifying semantically similar requirements between documents initiates with data acquisition from diverse sources like the DNG (Requirement Database) and essential requirements of prominent entities such as the Volkswagen Group. Following data collection, a series of preprocessing steps are employed, encompassing tasks like lower case conversion and stop words removal, ensuring uniformity and enhancing the quality of the textual data. Subsequently, the text is tokenized to facilitate further analysis, and a transformer model, like BERT or RoBERTa, is selected to generate embeddings, representing the semantic content of the words. These embeddings are then aggregated to derive a mean feature vector for each requirement, enabling robust comparison and semantic similarity analysis. The prepared data is then divided into feature and label vectors for training various classifiers such as Support Vector Machines (SVM) and K-Nearest Neighbors (KNN), which learn to differentiate between similar and dissimilar requirements based on the extracted features. Finally, the trained models undergo evaluation using metrics like

F1 score, precision, recall, and accuracy to gauge their performance accurately.

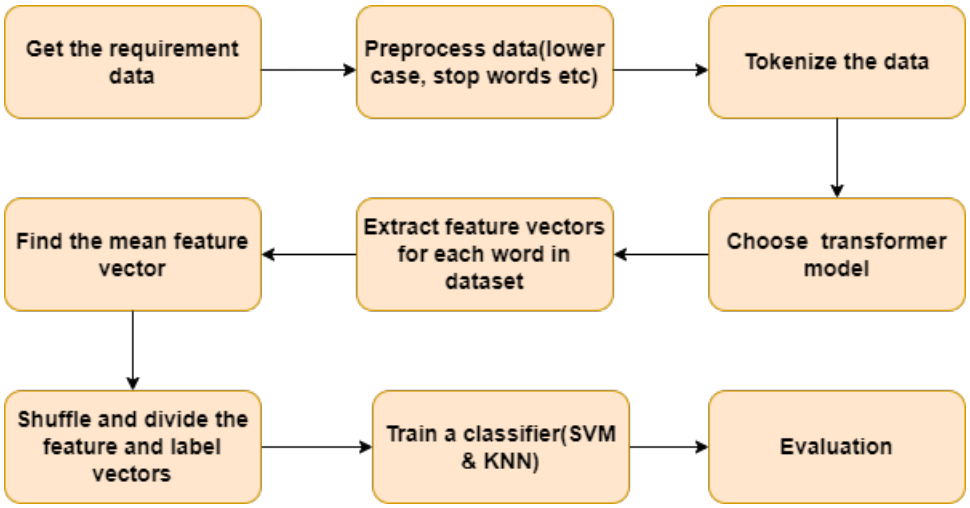


Figure 3.2: Methodology

This methodology offers a comprehensive approach to identify semantically similar requirements, crucial for tasks like requirement reuse and conflict resolution in software and automotive engineering domains. By integrating techniques from natural language processing , this approach enables the extraction and analysis of nuanced semantic relationships within textual data, ultimately enhancing decision-making processes in requirement engineering. Through meticulous data preprocessing, feature extraction, and model training, this methodology aims to provide practical insights into the semantic similarity between requirements, contributing to more efficient and reliable software development practices.

3.3 Techniques Used

The following techniques are used for the experimentation

3.3.1 Sentence Transformer

In this project, a Sentence Transformer, a variant of the popular BERT (Bidirectional Encoder Representations from Transformers) model, specifically fine-tuned for sentence-level embeddings, is utilized for semantic similarity detection of requirements. Unlike traditional BERT, which is primarily designed for token-level embeddings, Sentence Transformer is customized for generating high-quality fixed-size sentence embeddings. This makes it particularly well-suited for semantic similarity detection tasks, such as comparing requirements.

There are three main types of attention mechanisms employed in the Transformer architecture:

Encoder-Decoder Attention: This attention mechanism facilitates interaction between the input sequence and the output sequence in sequence-to-sequence tasks. By attending

to relevant parts of the input sequence while generating the output sequence, the model can effectively align input and output tokens, enabling accurate translation and sequence generation.

Self-Attention in the Input Sequence: In this mechanism, the model attends to all words in the input sequence to compute representations of each token. By considering the relationships between all tokens simultaneously, the model can capture global dependencies and extract relevant information from the input sequence, enhancing its ability to understand and process complex language structures.

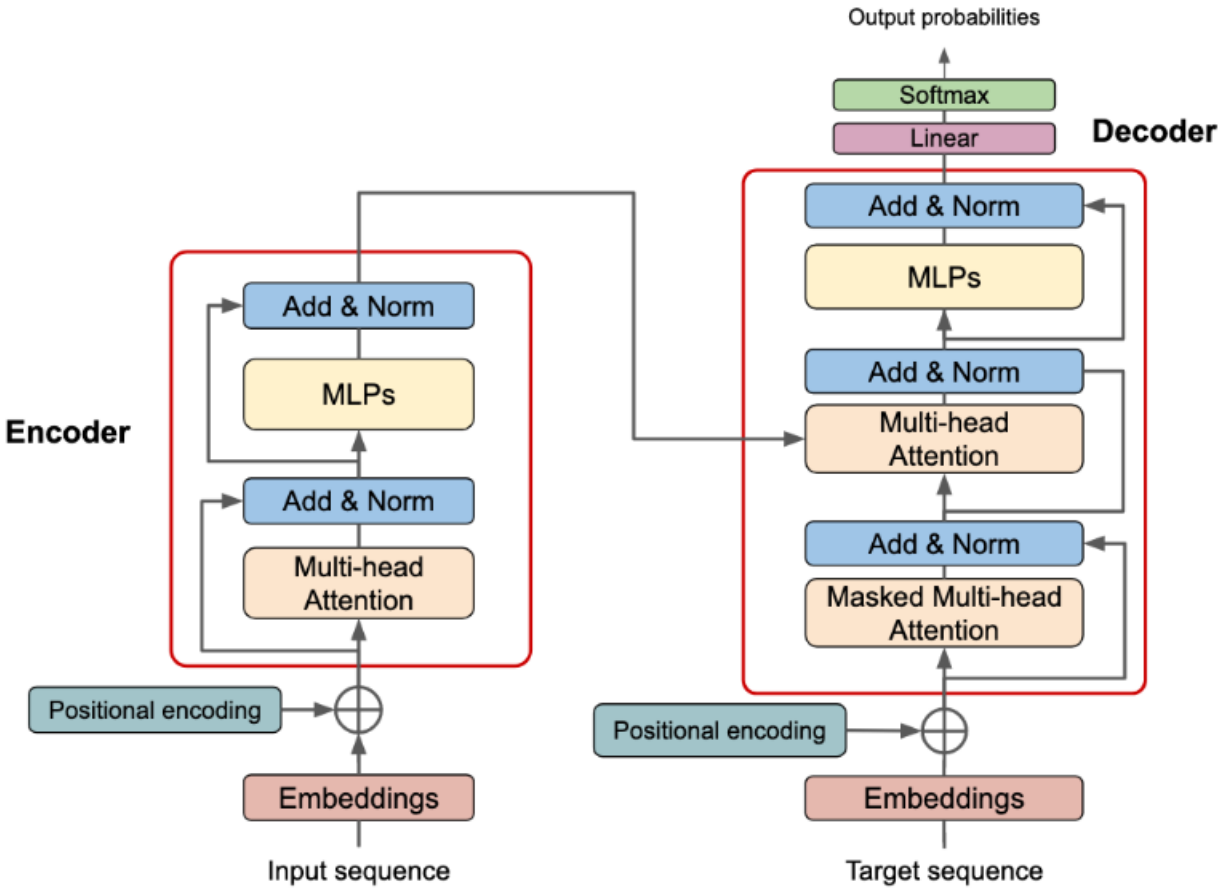


Figure 3.3: Transformer Architecture

Self-Attention in the Output Sequence: Similar to self-attention in the input sequence, this mechanism allows the model to attend to all words in the output sequence. However, a crucial consideration is that the scope of self-attention is limited to words occurring before a given word to prevent information leaks during training. This is achieved by masking words occurring after each step, ensuring that the model only attends to previous tokens during generation, thereby maintaining coherence and consistency in the output sequence.

The Transformer's reliance on self-attention mechanisms revolutionizes the field of NLP by overcoming limitations associated with sequential architectures. By enabling parallel

processing and capturing long-range dependencies effectively, the Transformer enhances the efficiency and performance of various NLP tasks, including machine translation, text summarization, and language understanding.

So the Transformer architecture represents a paradigm shift in neural network design, leveraging self-attention mechanisms to compute representations of input and output sequences efficiently. Through encoder-decoder attention and self-attention mechanisms in both input and output sequences, the Transformer can capture complex relationships within data sequences, thereby driving advancements in NLP and other domains requiring sequence modeling.

In the context of semantic similarity detection of requirements, the process typically starts with pretraining a Sentence Transformer model on a substantial corpus of textual data. This pre-trained model is then utilized to generate embeddings for individual sentences, including requirements. These embeddings encapsulate the semantic content of the sentences in a dense numerical format. These embeddings are then compared using similarity metrics such as cosine similarity or Euclidean distance to determine the semantic similarity between requirements.For requirement similarity detection, a dataset comprising pairs of requirements, each labeled with a similarity score, is created. This dataset serves as training data to fine-tune the pre-trained Sentence Transformer model specifically for the semantic similarity task. During fine-tuning, the model adjusts its parameters to generate embeddings that maximize the similarity between semantically similar requirements and minimize the similarity between dissimilar ones.

Once fine-tuning is complete, the model can be used to generate embeddings for new requirements. Given a pair of requirements, embeddings are computed. The semantic similarity between these requirements is quantified using a similarity metric, typically cosine similarity, which measures the cosine of the angle between the two embeddings. A higher cosine similarity score indicates a higher degree of semantic similarity between the requirements.To ensure the robustness and reliability of the semantic similarity model, it is crucial to validate its performance on a separate validation dataset. This evaluation involves comparing the model-generated similarity scores with the ground truth labels provided in the validation dataset. Metrics such as precision, recall, and F1 score are commonly used to assess the model's performance in accurately capturing semantic similarity relationships between requirements.
Furthermore, the interpretability of the semantic similarity scores is essential for effective decision-making in requirement analysis. Visualizations, such as heatmaps or dendrograms, can provide intuitive representations of the similarity relationships between requirements, aiding stakeholders in understanding the underlying semantic structure of the dataset. Post-processing techniques, such as thresholding or clustering, can further categorize requirements based on their semantic similarity levels, facilitating prioritization and decision-making processes in project planning.

In conclusion, the process of semantic similarity detection of requirements involves pretraining and fine-tuning a Sentence Transformer model, generating embeddings for requirements, computing similarity scores, validating model performance, and interpreting similarity relationships. By leveraging advanced machine learning techniques and visualization

tools, organizations can enhance requirement understanding, streamline project workflows, and improve decision-making processes in software development and project management contexts.

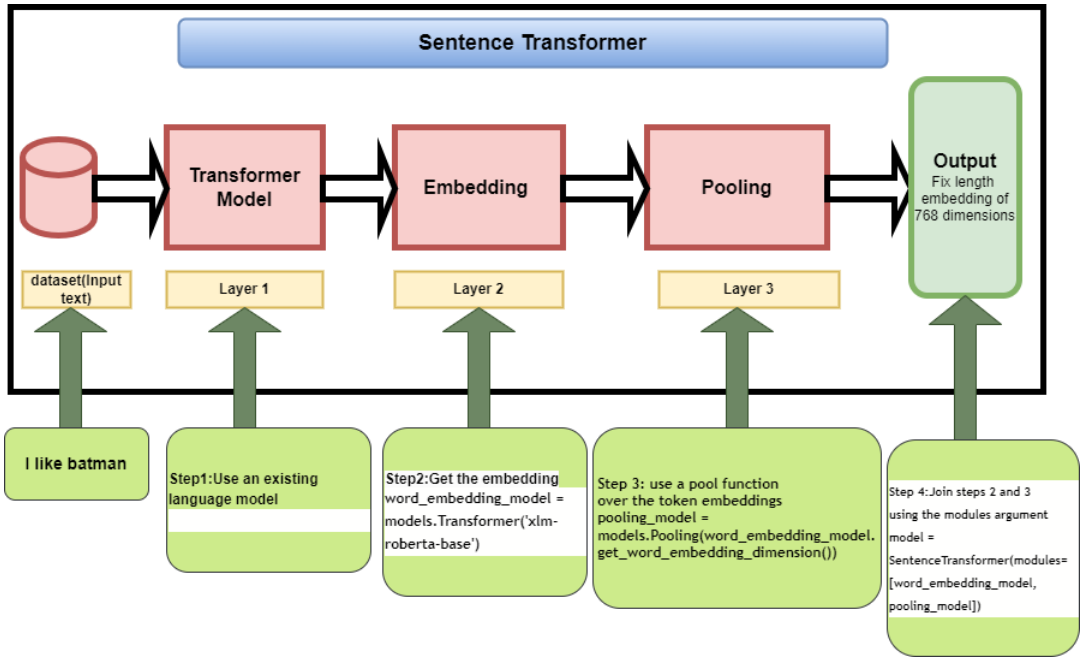


Figure 3.4: Sentence Transformer Architecture

The two pre-trained Sentence Transformer models used are:-

- all-MiniLM-L6-v2
- paraphrase-MiniLM-L6-v2

3.3.2 all-MiniLM-L6-v2

The "all-MiniLM-L6-v2" model emerges as a robust and adaptable solution engineered to tackle a myriad of natural language processing (NLP) tasks. Its strength lies in its extensive fine-tuning on a colossal dataset of over 1 billion training pairs, fostering a profound understanding of diverse linguistic nuances and contexts. Rooted in the "nreimers/MiniLM-L6-H384-uncased" architecture, characterized by six layers and a hidden size of 384, this model strikes a balance between complexity and efficiency. Its "uncased" nature enhances versatility, ensuring seamless handling of both lower and uppercased text inputs with equal adeptness.

Moreover, the model's flexibility extends to its support for various similarity score functions, including dot-product, cosine similarity, and Euclidean distance metrics. This versatility empowers users to tailor the model's performance to their specific task requirements, optimizing outcomes across diverse scenarios. Despite its formidable capabilities, the model

maintains a lightweight profile, boasting a compact size of 80 MB. With mean pooling employed for aggregating token embeddings, simplicity meets effectiveness, ensuring the generation of comprehensive sentence-level representations. In essence, the "all-MiniLM-L6-v2" model stands as a testament to adaptability and efficiency, poised to address the multifaceted challenges of modern NLP with unwavering reliability.

3.3.3 paraphrase-MiniLM-L6-v2

The "paraphrase-MiniLM-L3-v2" model represents a specialized solution finely tuned for detecting paraphrases within text data. Built upon the "nreimers/MiniLM-L3-H384-uncased" architecture, it adopts the MiniLM framework with three layers and a hidden size of 384, indicating a balance between computational efficiency and effective representation learning. Operating with a maximum sequence length of 128 tokens and an embedding dimensionality of 384, each token in the input text is encoded into a vector of consistent length, facilitating comprehensive semantic analysis.

Unlike its counterpart, the "all-MiniLM-L6-v2" model, the embeddings generated by the "paraphrase-MiniLM-L3-v2" model are not normalized, offering a different approach to capturing semantic similarities. Equipped with cosine similarity as the primary score function for calculating similarity between embeddings, the model ensures efficient detection of paraphrases across varied datasets. Trained on a diverse array of datasets, including AllNLI, sentence-compression, and others, the model draws upon a rich repository of paraphrase examples to hone its detection capabilities. With mean pooling employed to aggregate token embeddings, the model synthesizes sentence-level representations, enabling effective paraphrase detection across diverse linguistic contexts. In essence, the "paraphrase-MiniLM-L3-v2" model stands as a specialized tool meticulously crafted to excel in the nuanced task of paraphrase detection as well as similarity task, leveraging a combination of architectural finesse and comprehensive training data.

3.4 Classifiers Used

The classifiers employed for training the labeled requirements includes :-

- Support Vector Machine(SVM)
- K- Nearest Neighbour(KNN)

3.4.1 Support Vector Machine(SVM)

Support Vector Machine (SVM) stands as a stalwart in the realm of supervised learning, renowned for its efficacy in classification tasks. In our project, aimed at detecting the semantic similarity of requirements, SVM plays a pivotal role as our chosen classifier.At its core, SVM operates by identifying the optimal hyperplane that best separates different classes in a high-dimensional space. In our case, these classes represent the semantic similarity levels of requirements—0 denoting dissimilarity and 1 signifying similarity. By harnessing a labelled

dataset and leveraging the rich embeddings generated by the MiniLM-L6-v2 and paraphrase-MiniLM-L6-v2 encoding models from the Sentence Transformer library, SVM adeptly learns to discern intricate patterns within the data.
only the probability parameter is explicitly set to True to enable probability estimates. All other parameters are set to their default values. Throughout the training phase, SVM diligently learns from the labelled dataset, iteratively adjusting its decision boundary to maximize the margin between different classes while minimizing classification errors. This process culminates in a well-fitted model capable of accurately delineating between similar and dissimilar requirements.In the broader context of our project, SVM serves as a reliable cornerstone, empowering our system to effectively gauge the semantic congruity between diverse requirements—a critical task in numerous domains ranging from software engineering to natural language processing.

3.4.2 K- Nearest Neighbour(KNN)

K-Nearest Neighbors (KNN) is an powerful algorithm in the landscape of machine learning, particularly renowned for its simplicity and versatility in classification tasks. In our project focused on detecting the semantic similarity of requirements, we opted to employ KNN as our classifier, eschewing the intricacies of hyperplane optimization in favor of a more intuitive approach.At its essence, KNN operates on the principle of proximity, classifying data points based on the majority class among their nearest neighbors. In our context, where requirements are represented as rich vector embeddings derived from the MiniLM-L6-v2 and paraphrase-MiniLM-L6-v2 encoding models provided by the Sentence Transformer library, KNN adeptly exploits the geometric relationships embedded within the high-dimensional space.
One of the main parameter used in KNN is neighbour values.This parameter sets the number of nearest neighbors to consider when making predictions. The default is 5, meaning the model will look at the 5 closest training samples to determine the class of a test sample.Throughout the training phase, KNN diligently learns from the labeled dataset, adjusting its classification based on the distribution of nearest neighbors. This iterative process culminates in a well-fitted model capable of accurately delineating between similar and dissimilar requirements.In the broader context of our project, KNN serves as a robust foundation, enabling our system to effectively discern semantic congruity among diverse requirements—a critical task with applications spanning software engineering, natural language processing, and beyond.

3.5 Datasets

The following datasets were used for the experimentation.

- Requirements extracted from DNG(Requirement Database)
- Requirements from Volkswagon Group

3.5.1 Dataset 1 :- Requirements satisfied by Continental Automotive(DNG)

The dataset used in this study comprises of 10,500 distinct requirements already satisfied by Continental Automotives Pvt.Ltd. Each requirement have the corresponding requirement ID assigned to it.

SL.No	REQ.ID1	REQ.ID2	Requirements1	Requirements2	Is_duplicate
1	CMIV-155.6	866282	The supplier shall provide the OEM with an API to enable development of applications that is able to interface all hardware and all applicable services on the system. The specific API requirements are to be found in detail in the corresponding component section of this document. Note 1: Based on the API-specification, available hardware and resource will depend on the configuration of partitions and containers. Note 2: The preferred API to the inter-process communication (IPC) is through AUTOSAR ara::com.	The system shall send via BSM the following information: Front Fog Lamp Indication On - NAFTA region: according SAE-J2735 [83] - China region: according CCSA/CSAE/C-ITS: YD/T 3709-2020 (DTL-1009)	0
2	CMIV-72.2	866283	The supplier shall provide, distribute and continously update documentation of how to use the API.	The system shall send via BSM the following information: Rear Fog Lamp Indication On - NAFTA region: according SAE-J2735 [83] - China region: according CCSA/CSAE/C-ITS: YD/T 3709-2020 (DTL-1009)	0
3	CMIV-69.2	866284	The supplier shall provide a generic software API for, communicating with and controlling the hardware and software components running on the platform. Note: The name of the commands or its parameters are not mandatory to comply with.	The system shall send via BSM the following information: High Beams Active BSM element name:highBeamHeadlightsOn - NAFTA region: according SAE-J2735 [83] - China region: according CCSA/CSAE/C-ITS: YD/T 3709-2020 (DTL-1009)	0

Figure 3.5: Dataset 1

Key features and information about the dataset include:

Requirement ID

- Each requirement contains its corresponding ID assigned to it.The distinct requirement ID serves as a unique identifier for requirements, facilitating efficient communication, traceability of changes, and systematic documentation, essential for organized project management and collaboration.

Requirements

- A requirement in the sense is a documented statement that specifies a condition or capability that a system, product, or service must meet. It serves as a guideline for design, development, and testing.

Is.duplicate

- It represents whether two requirements are similar or not.0 indicates two requirements are not similar whereas 1 represents requirements are similar.

The following figure 3.6 illustrates how data is extracted from the DNG(Requirement Management Database) consisting of requirements satisfied by Continental Automotives with their corresponding requirement ID:-

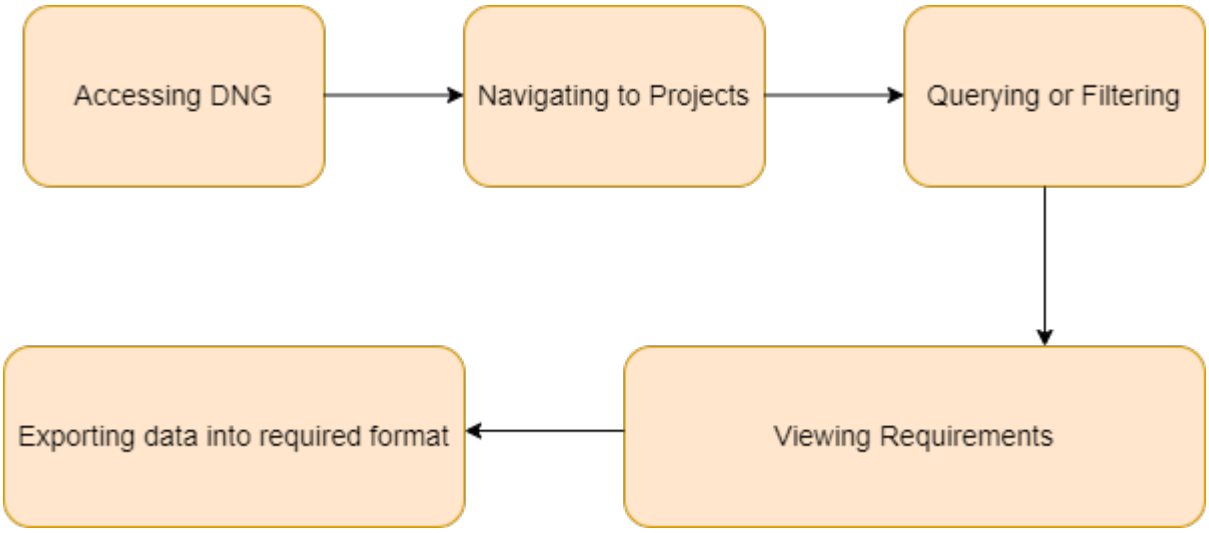


Figure 3.6: Requirement Extraction from DNG

3.5.2 Requirements from Volkswagon Group

The dataset used in this study comprises of 997 distinct requirements that need to be satisfied by Continental Automotives Pvt.Ltd. Each requirement have the corresponding requirement ID assigned to it.

Key features and information about the dataset include:

- Requirement ID: Each requirement contains its corresponding ID assigned to it.The distinct requirement ID serves as a unique identifier for requirements, facilitating efficient communication, traceability of changes, and systematic documentation, essential for organized project management and collaboration.
- Requirements: A requirement in the sense is a documented statement that specifies a condition or capability that a system, product, or service must meet. It serves as a guideline for design, development, and testing.

Object ID	Object Text
BTL-LAH-19	This BT-LAH describes the performances, requirements, test and validation conditions that must be met by the product to be developed and by the contractor.
BTL-LAH-2363	The contractor must analyze which hazards could potentially be posed by the component, including the hardware, software, mechanics and any other technology; this must also take place independently of the scope of ISO 26262. Potential causes must be systematically identified here.
BTL-LAH-1227	The microcontroller ports must be protected against overvoltage by means of suitable protective circuits.
BTL-LAH-1301	The contractor must ensure that all software modules are tested for proper function in the overall program before any delivery.
BTL-LAH-1384	The contractor receives the template for the interface description from the client's department responsible.
BTL-LAH-4082	A selectively switchable ECU that can be shut off even if terminal 15 is on must be designed for a service life of equal to or greater than 3,000,000 wake-up cycles.
FBR5-721	In addition, a parameter test (function test) shall be performed each time the upper test temperature is reached and each time the lower test temperature is reached.
	The DUT shall be fully functional before, during and after the test and all parameters shall meet the specifications. Verification is done by

Figure 3.7: Dataset 2

The following figure 3.8 illustrates how data is extracted from the PDF consisting of requirements specified by the Volkswagon Group with their corresponding requirement ID and workflow for processing and analyzing PDF documents to extract and manipulate requirements, convert them into Excel format, and analyze patterns using regular expressions (Regex).

This workflow streamlines the intricate process of extracting and manipulating requirements from PDF documents by employing a systematic approach. Beginning with the PDF Reader, it starts through the document, discerning and isolating relevant pages that contain essential information. These selected pages then undergo meticulous scrutiny through the PDF Manipulator, which deftly extracts individual requirements, including their respective IDs and types, and performs necessary document modifications, ensuring a streamlined and organized structure. Following this, the PDF Text Extractor steps in, converting the extracted textual content into a format amenable to further processing, a pivotal transition that sets the stage for subsequent analysis. The Excel Converter then seamlessly transforms this processed data into an Excel spreadsheet, providing a user-friendly interface for subse-

quent manipulation and analysis.

However, the most important factor of this workflow lies in the pattern analysis stage, where the raw text data undergoes scrutiny through regular expressions (Regex). This powerful tool meticulously scours the text, identifying and deciphering intricate patterns, formats, or structures, thus furnishing invaluable insights for refining and categorizing requirements with unparalleled precision. Through this structured methodology, the workflow ensures not only the accuracy of extracted information but also empowers efficient decision-making and project management through comprehensive requirement analysis.

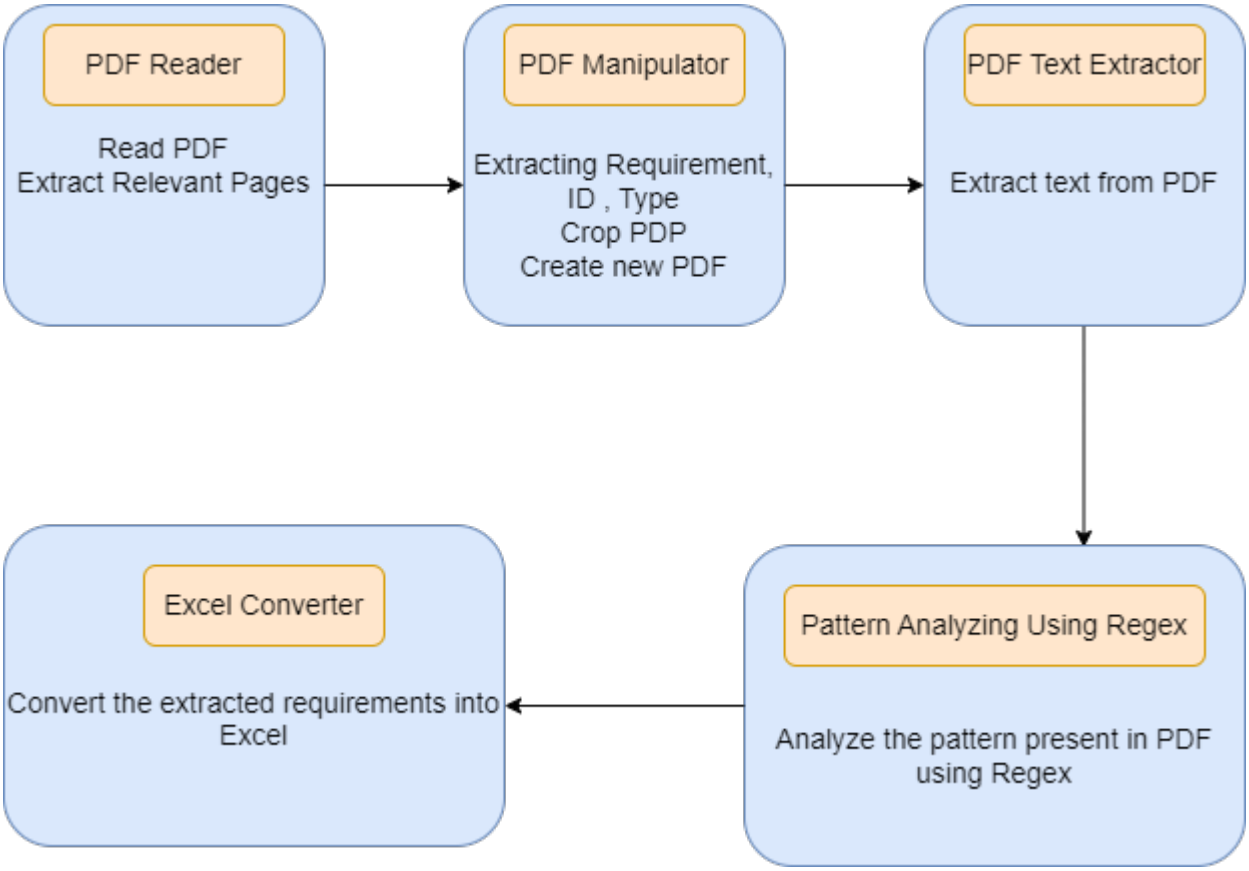


Figure 3.8: Requirement extraction from PDF of Volkswagon

3.6 Streamlit - Graphical User Interface

Integrating Streamlit into the project as the interface layer has significantly enhanced the accessibility and usability of our semantic similarity detection system. With its user-friendly design and interactive features, Streamlit offers a seamless experience for stakeholders to engage with the system effortlessly. Through the Streamlit interface, users can conveniently input requirements and receive immediate feedback on their semantic similarity, empowering

them to make informed decisions quickly. Leveraging Streamlit's integration capabilities, we seamlessly incorporate our trained KNN model for semantic similarity classification into the interface, providing real-time insights into the semantic congruity of submitted requirements. Additionally, Streamlit's visualization tools enable users to interpret and analyze the results effectively, fostering collaboration and facilitating communication among project stakeholders.

By deploying our interface through Streamlit's hosting services, we ensure widespread accessibility and seamless collaboration among project stakeholders. Streamlit's collaborative features facilitate effortless sharing and interaction, allowing teams to work together seamlessly regardless of their geographical location or technical proficiency. Furthermore, Streamlit's dynamic nature enables us to continually enhance and refine the interface based on user feedback, ensuring that it remains intuitive and responsive to the evolving needs of our users. In summary, Streamlit serves as a vital component of our semantic similarity detection system, bridging the gap between sophisticated machine learning algorithms and end-users with its intuitive interface and collaborative features, ultimately driving efficiency and effectiveness in requirement analysis and decision-making processes.

3.7 Performance Metrics

To measure the performance of the trained Sentence Transformer Model for semantic similarity detection of requirements, We made use of the F1-score measure, which is the harmonic mean of recall and precision. Precision is the percentage of correctly recognised labels. Recall is a component of successful label extraction.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.2}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP} \tag{3.3}$$

The weighted average of precision and recall is the F1-score. It considers both false positives and false negatives to determine the model's overall accuracy.

$$\text{F1score } = 2 * \frac{\text{Precision } * \text{ Recall}}{\text{Precision } + \text{ Recall}} \tag{3.4}$$

Confusion Matrix

A table used to assess a classification model's performance is called a confusion matrix. It lists the total number of false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN) that the model generated. These measures are frequently used to evaluate the model's F1-score, recall, accuracy, and precision, among other things. The confusion matrix is organized as follows:

- True Positives (TP): These are cases where the model correctly predicted the positive class (e.g., correctly detecting semantically similar requirements).
- True Negatives (TN): These are cases where the model correctly predicted the negative class (e.g., incorrectly detecting semantically similar requirements).
- False Positives (FP): These are cases where the model incorrectly predicted the positive class when it should have been negative (e.g., falsely detecting semantically similar requirements when it's not). Also known as Type I error.
- False Negatives (FN): These are cases where the model incorrectly predicted the negative class when it should have been positive (e.g., failing to detect semantically similar requirement when it's actual). Also known as Type II error.

A confusion matrix helps assess the model's performance in terms of correctly classifying instances and identifying any biases it may have toward certain classes. From the confusion matrix, various performance metrics like accuracy, precision, recall, F1-score, and the Matthews correlation coefficient can be calculated to provide a comprehensive evaluation of the model's effectiveness.

ROC Curve (Receiver Operating Characteristic Curve)

A graphical tool for assessing a binary classification model's performance at various discrimination thresholds is the ROC curve. At different threshold values, it compares the True Positive Rate (TPR) against the False Positive Rate (FPR). Here is what the ROC curve illustrates:

- True Positive Rate (TPR), also called Sensitivity or Recall: TPR measures the proportion of actual positive cases that the model correctly identifies as positive. It is calculated as TPR = TP / (TP + FN).
- False Positive Rate (FPR): FPR measures the proportion of actual negative cases that the model incorrectly classifies as positive. It is calculated as FPR = FP / (FP + TN).

The trade-off between specificity (1 - FPR) and sensitivity (TPR) is depicted by the ROC curve as the model's threshold for identifying positive or negative examples is changed. An ideal model would have a ROC curve with high sensitivity and specificity that reaches the top-left corner (TPR = 1, FPR = 0). A greater AUC-ROC denotes better model performance. The area under the ROC curve (AUC-ROC) is a single value that sums together the whole performance of the model.

Cosine Similarity

Cosine similarity is a measure used to determine the similarity between two vectors in a multi-dimensional space. In the context of natural language processing (NLP), cosine similarity is often employed to assess the similarity between two text documents represented as vectors in a high-dimensional space, typically using techniques like word embeddings or document embeddings.
Cosine similarity is a mathematical measure commonly used in natural language processing

(NLP) and information retrieval to determine the similarity between two vectors in a multi-dimensional space. It calculates the cosine of the angle between the two vectors, providing a value between -1 and 1, where a value of 1 indicates perfect similarity and -1 indicates perfect dissimilarity. Mathematically, cosine similarity (cosine_similarity($\mathbf{A}, \mathbf{B}$)) between two vectors $\mathbf{A}$ and $\mathbf{B}$ is computed as the dot product of the vectors divided by the product of their magnitudes:

The cosine similarity between two vectors $\mathbf{A}$ and $\mathbf{B}$ is defined as:

$$\text{cosine\_similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{3.5}$$

Here, $\mathbf{A}$ and $\mathbf{B}$ represent the vectors, $\mathbf{A} \cdot \mathbf{B}$ denotes their dot product, and $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ represent their magnitudes. The cosine similarity metric is particularly useful in measuring the semantic similarity between text documents or individual requirements in system and software engineering. By representing requirements as numerical vectors using techniques like word embeddings, cosine similarity enables efficient comparison and identification of semantically similar requirements, facilitating better requirement management and decision-making processes in software development projects.

Euclidian Distance

Euclidean distance is a measure of the straight-line distance between two points in a multi-dimensional space. In the context of natural language processing (NLP) and information retrieval, Euclidean distance is often used to quantify the dissimilarity between vectors representing text documents or features extracted from them.
Here's how Euclidean distance is defined mathematically:

Given two vectors A and B in an n-dimensional space, the Euclidean distance is given by:

$$\text{euclidean\_distance}(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2} \tag{3.6}$$

Here, A_i and B_i represent the elements of vectors $\mathbf{A}$ and $\mathbf{B}$, respectively. Both cosine similarity and Euclidean distance are valuable tools for comparing and analyzing vectors in NLP tasks. While cosine similarity captures the angle between vectors, which is useful for measuring semantic similarity, Euclidean distance provides a measure of geometric distance, which can be beneficial in clustering and classification tasks. In the context of system and software engineering, these metrics enable efficient comparison and identification of semantically similar requirements, facilitating better requirement management and decision-making processes in software development projects.

In general,cosine similarity and euclidean distance are two common metrics used in natural language processing (NLP) and information retrieval to assess the similarity between vectors in a multi-dimensional space. Cosine similarity calculates the cosine of the angle between two vectors, providing a value between -1 and 1, where 1 indicates perfect similarity and -1 indicates perfect dissimilarity. Mathematically, cosine similarity (cosine_similarity($\mathbf{A}, \mathbf{B}$)) between two vectors $\mathbf{A}$ and $\mathbf{B}$ is computed as the dot product of the vectors divided by the

product of their magnitudes.On the other hand, Euclidean distance measures the straight-line distance between two points in a multi-dimensional space. Mathematically, the Euclidean distance (euclidean_distance($\mathbf{A}, \mathbf{B}$)) between two vectors $\mathbf{A}$ and $\mathbf{B}$ is computed as the square root of the sum of the squared differences between corresponding elements of the vectors.

3.7.1 Key benefits of similarity metrics

The key benefits of similarity metrics includes:

- Distribution Analysis
- Distinct Capabilities
- Contextual Suitability

Distribution Analysis

- 50-70% Range: Both methods show a substantial portion of similarity scores within the 50-70% range—74.3% for Euclidean distance and 70.7% for cosine similarity. This indicates that both methods generally agree on the moderate similarity between pairs.
- $<$ 50% Range: Both methods also have a quarter of their scores below 50% (25.7%), suggesting a consensus on identifying pairs with low similarity.

Distinct Capabilities

- Euclidean Distance: This method clusters the majority of its scores within the 50-70% range and has no scores in the 70-90% or 90-100% ranges. This pattern suggests that Euclidean distance is effective in identifying moderate similarity but lacks the sensitivity to detect higher similarity levels. Euclidean distance is influenced by the absolute differences in magnitudes, which can be beneficial in scenarios where the actual size of differences matters. However, its inability to recognize higher similarity scores might limit its usefulness in tasks requiring fine-grained differentiation among highly similar items.
- Cosine Similarity: While it also has a large portion of scores in the 50-70% range, cosine similarity identifies 3.6% of the scores in the 70-90% range, which Euclidean distance does not. This indicates that cosine similarity is more sensitive to higher degrees of similarity. By measuring the cosine of the angle between vectors, it focuses on the direction rather than the magnitude, making it particularly effective in contexts where the orientation of data points is crucial. This sensitivity allows for a more nuanced understanding of relationships, especially in domains where similar items share a common direction but may differ in magnitude, such as text analysis or collaborative filtering in recommendation systems.

Contextual Suitability

- Euclidean Distance: Best suited for tasks where the overall magnitude of differences is important, such as in spatial analyses or certain clustering tasks where absolute differences in feature values are significant.
- Cosine Similarity: More appropriate for contexts requiring the identification of higher similarity degrees. It excels in applications like text analysis, recommendation systems, and other scenarios where the relative similarity (direction of vectors) is more relevant than the absolute magnitude.

The choice between Euclidean distance and cosine similarity should be guided by the specific requirements and characteristics of the task at hand. If the goal is to detect and differentiate higher similarity relationships, cosine similarity is preferable due to its sensitivity to higher similarity scores and its focus on the direction of vectors. In contrast, if the task involves scenarios where moderate similarities are predominant and absolute differences in feature values are crucial, Euclidean distance might be more effective. Both methods offer distinct advantages, and understanding these can help in selecting the most appropriate approach for mapping requirement similarity in a given context.

3.7.2 Benefits of Streamlit as a Graphical User Interface

Streamlit is an excellent choice for developing a graphical user interface (GUI) for detecting semantic similarity of requirements due to several compelling reasons:-

- Interactivity and Real-time updates : Streamlit enables the creation of highly interactive interfaces that update in real-time. This interactivity is crucial for users analyzing semantic similarities as it allows them to upload files, choose models and similarity measures, and instantly see the results of their analysis. Real-time feedback enhances the user experience and makes the analysis process more efficient and engaging.
- Simplicity and ease to use : Streamlit's simplicity and ease of use are significant advantages. With minimal coding effort, developers can create sophisticated, user-friendly GUIs. Streamlit abstracts much of the complexity involved in traditional web development, allowing developers to focus on building functionality rather than dealing with HTML, CSS, or JavaScript. This makes it easier to quickly prototype and deploy interfaces for requirement analysis.
- Integration with wide range of libraries : Streamlit seamlessly integrates with popular data science libraries such as Pandas, NumPy, Scikit-learn, and TensorFlow. This integration allows for the efficient processing and analysis of requirements data. Developers can leverage pre-trained models and natural language processing (NLP) techniques to detect semantic similarities, directly incorporating these tools into the Streamlit interface.
- Visualisation Capabilities : Streamlit offers robust support for data visualization through libraries like Matplotlib, Seaborn, and Plotly. Visualization is key in requirement analysis as it helps users to intuitively understand the relationships and similarities between different requirements. Streamlit makes it easy to generate and display charts, graphs, and other visual aids that can enhance the analysis and interpretation of data.

- Rapid deployment and sharing : One of Streamlit's standout features is its ease of deployment. Applications can be shared with stakeholders or team members via simple web URLs, without the need for complex deployment pipelines. This is particularly beneficial in collaborative environments where multiple users need to access and interact with the requirement analysis tool.
- Customizability and flexibility : Streamlit provides a high degree of customizability, allowing developers to tailor the GUI to meet specific project needs. Users can add various input elements like sliders, dropdowns, and text boxes to customize the analysis process. This flexibility ensures that the tool can adapt to different workflows and user requirements.
- Community and support : Streamlit has a vibrant and growing community, with extensive documentation and support resources available. This community support is invaluable for developers, offering tutorials, example projects, and forums where they can seek help and share knowledge.

In conclusion, Streamlit emerges as a highly effective tool for developing graphical user interfaces for tasks such as detecting semantic similarity of requirements. Its ability to create interactive, real-time updating interfaces makes it particularly suited for applications requiring dynamic user engagement. The simplicity and ease of use provided by Streamlit enable rapid development and deployment, allowing even those with minimal web development experience to build powerful and user-friendly applications.

Chapter 4

Experimental Analysis and Quantitative Results

4.1 Experiments on data pre-processing task

The preprocessing steps are meticulously executed to prepare the dataset for detecting semantically similar requirements. Initially, thorough text cleaning is performed to eliminate extraneous characters such as punctuation marks, special symbols, and non-text elements, ensuring data consistency. Tokenization follows, breaking down the text into meaningful units, which facilitates structured input for the model. Standardizing the text through lowercasing eliminates discrepancies caused by case sensitivity, ensuring uniformity. Stopword removal reduces noise by eliminating common words that do not contribute significant meaning, allowing the model to focus on more relevant parts of the text. Lemmatization or stemming is then applied to reduce words to their base or root form, capturing the core meaning of words and ensuring different forms of the same word are recognized as equivalent.

Addressing synonyms and variations is another crucial step to enhance the model's adaptability to diverse phrasings, ensuring different expressions of the same concept are understood correctly. Numerical data is handled appropriately to maintain its significance within the text, either by normalizing numbers, converting them into words, or preserving their format based on the context. Quality control checks are conducted throughout the preprocessing steps to ensure data integrity, verifying that the cleaned and processed text remains accurate and meaningful. These tasks involve iterative adjustments to optimize the preprocessing process, continuously refining each step based on experimental outcomes. The combination of these steps results in a high-quality dataset, crucial for enhancing the performance of models like Sentence-BERT in capturing semantic similarities among requirements, ultimately ensuring accurate and meaningful results..

4.2 Environmental Setup

Experiments using datasets were conducted in a .py script within Visual Studio Code on a system with 8 GB RAM, an AMD Ryzen 5 5600H processor at 3.30 GHz, integrated Radeon Graphics, and an additional 4 GB NVIDIA RTX Graphics card. The experimental

setup leveraged Keras with TensorFlow as the backend to build and train the models. For generating sentence embeddings, the 'Paraphrase-MiniLM-L6-v2' model from the Sentence Transformer library was utilized. This model is known for its efficiency and effectiveness in capturing semantic nuances, providing robust embeddings that serve as the foundation for subsequent analysis.

Performance analysis was conducted using an accuracy performance evaluation matrix. This matrix provided a comprehensive assessment of each model's effectiveness, allowing for a detailed comparison of their accuracy in identifying semantically similar requirements. The evaluation considered various metrics to ensure a thorough understanding of each model's strengths and weaknesses. Through this rigorous analysis, the experiments aimed to identify the best-performing model, ultimately guiding the selection of the most suitable approach for semantic similarity detection in the given datasets. This methodical approach ensured that the final model chosen would offer the highest accuracy and reliability for practical applications.

4.3 Results

4.3.1 Distribution of duplicates and non-duplicates

- The figure 4.1 represents a visual depiction of the distribution of requirements, distinguishing between duplicates and non-duplicates. In this context, duplicates signify requirements that are similar, while non-duplicates represent those that are dissimilar. This distinction is crucial for understanding the semantic relationships among various requirements within the dataset.
- Remarkably, the dataset exhibits a balanced composition, with approximately 64% of the requirements classified as non-duplicates and the remaining 36% as duplicates. This balance ensures that the dataset adequately encompasses both similar and dissimilar pairs of requirements, facilitating robust training and evaluation of models for requirement analysis tasks.

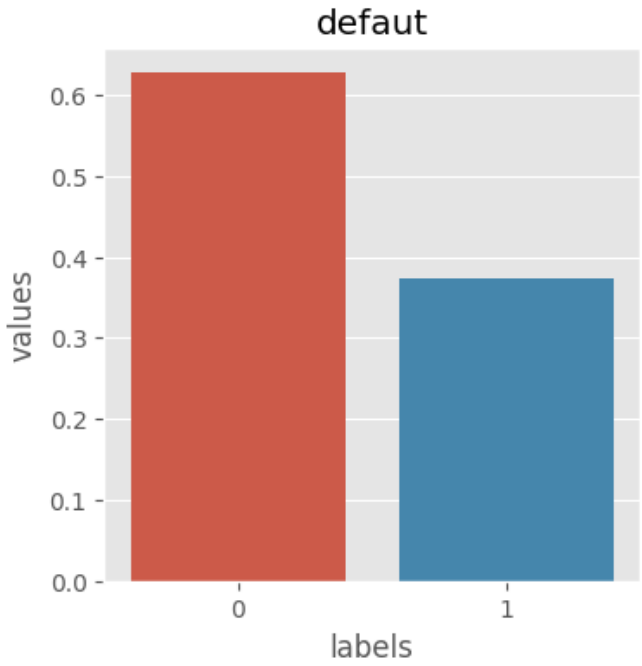


Figure 4.1: Duplicates and Non-duplicates distribution

4.3.2 Text length distribution

- The distribution of text lengths for requirements in the dataset can help researchers or analysts understand the variability and patterns in the length of the requirement. This analysis might be useful for subsequent natural language processing tasks, model training, or simply for gaining insights into the characteristics of the dataset.

- The significance of text length lies in its ability to offer valuable insights into various aspects of written communication. Analyzing the length of text provides a glimpse into the communication style, content complexity, and readability. Longer texts often indicate detailed or complex information, while shorter ones tend to be more concise and accessible.

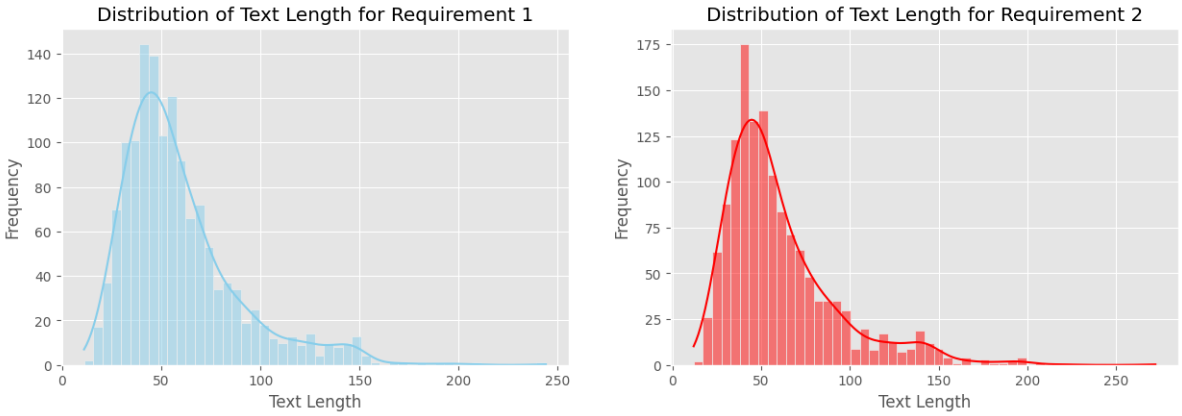


Figure 4.2: Text length distribution of requirement 1 and requirement 2

4.3.3 Performance Evaluation of Sentence-Transformer Models

The comparison between the 'all-MiniLM-L6-v2' and 'paraphrase-MiniLM-L6-v2' models reveals that 'all-MiniLM-L6-v2' outperformed the latter in several key metrics. Specifically, 'all-MiniLM-L6-v2' achieved an accuracy of 92%, a precision of 95%, a recall of 82%, and an F1 score of 88%.
These metrics are indicative of the model's overall performance in accurately classifying requirements as either similar or dissimilar. A high accuracy score of 92% demonstrates the model's ability to make correct predictions on a majority of the test data. Additionally, a precision of 95% indicates that when the model predicts a requirement pair as similar, it is correct 95% of the time. A recall of 82% reflects the model's capacity to identify a significant proportion of the actual similar requirement pairs. Finally, an F1 score of 88% provides a balanced measure of the model's precision and recall, taking into account both false positives and false negatives.

Evaluation Metrics	MODEL 1 (SVM)	Model 1 (KNN)	Model 2 (SVM)	Model 2 (KNN)
Accuracy	85%	84%	86%	85%
Precision	88%	87%	89%	88%
Recall	90%	89%	91%	90%
F1 score	83%	82%	84%	83%

Table 4.1: Performance Evaluation of the Models Used

where Model 1 stands for paraphrase-MiniLM-L6-v2 and Model 2 stands for all-MiniLM-L6-v2

4.3.4 Accuracy versus Epoch Evaluation

The accuracy versus epoch graph provides a visual representation of the model's performance over the course of training. Initially, the accuracy stands at 88.03% in the first epoch, indicating a solid starting point. As training progresses through subsequent epochs, the accuracy steadily increases, reflecting the model's learning and refinement process. By the fifth epoch, the accuracy reaches its peak at 94.68%, demonstrating the model's ability to effectively capture and understand complex patterns in textual data. The plotted trajectory illustrates the model's incremental improvement over time, affirming its capability to adapt and enhance its performance with each epoch. This graph serves as a valuable tool for monitoring the training progress and assessing the convergence of the model towards optimal accuracy.

This incremental enhancement serves as a testament to the model's robust learning capability and its capacity to assimilate and internalize intricate patterns inherent within textual data. The plotted graph visually reinforces this narrative, showcasing the model's ability to dynamically adapt and refine its understanding with each successive epoch, thereby solidifying its effectiveness in capturing nuanced semantic relationships within system and software requirements

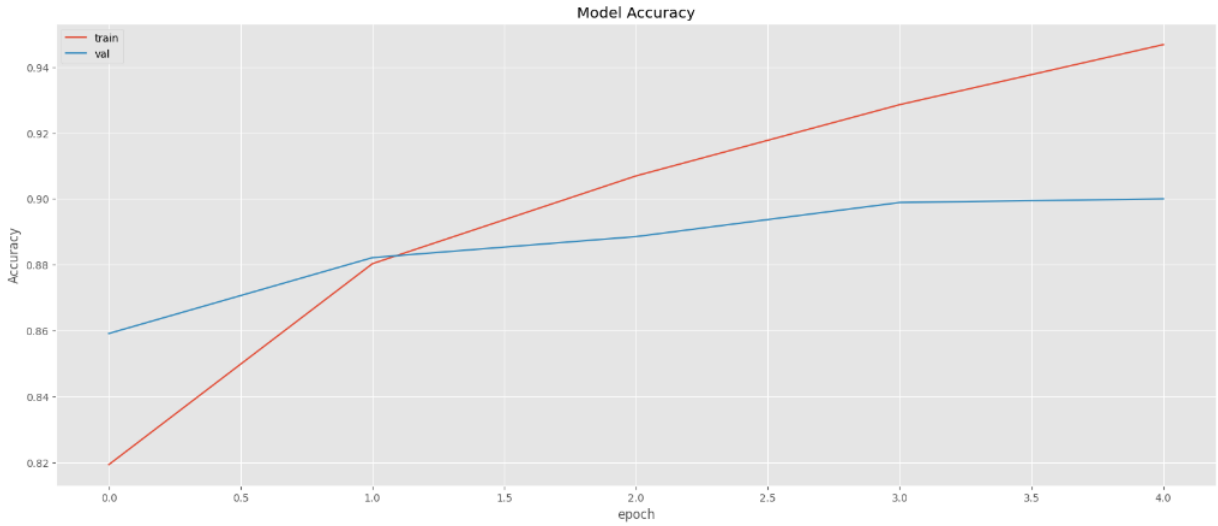


Figure 4.3: Accuracy graph of Siamese-BERT

4.3.5 ROC Curve and Confusion Matrix of SVM Classifier

An AUC (Area Under the Curve) score of 0.96 for the test set is indicative of the model's excellent generalization ability to unseen data. This high AUC value suggests that the model effectively distinguishes between duplicate and non-duplicate questions with a strong separability. Importantly, it underscores that the model's performance extends beyond the training dataset, highlighting its robustness and reliability in real-world applications.Furthermore,

the AUC score of 0.96 indicates that the model has not succumbed to overfitting during the training process. Overfitting occurs when a model learns to memorize the training data too well, resulting in poor performance on unseen data. However, with such a high AUC score, it is evident that the model has successfully captured the underlying patterns in the data without overfitting, thus ensuring its efficacy and reliability in practical scenarios.

The ROC (Receiver Operating Characteristic) curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR). The equations are given by:

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN} \tag{4.1}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN} \tag{4.2}$$

where:

TP = True Positives (correctly classified duplicates)
FN = False Negatives (incorrectly classified non-duplicates)
FP = False Positives (incorrectly classified duplicates)
TN = True Negatives (correctly classified non-duplicates)

The AUC (Area Under the Curve) can be computed from the ROC curve.

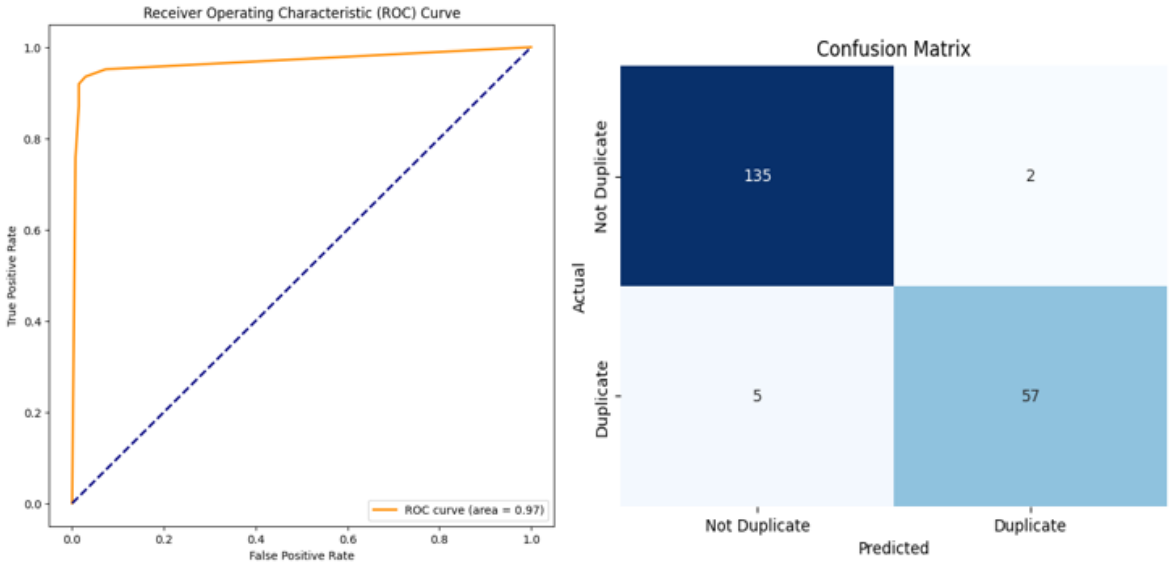


Figure 4.4: ROC Curve and Confusion Matrix of SVM Classifier

In essence, the AUC score of 0.96 in case of SVM Classifier provides strong evidence of the model's capability to generalize well, maintain robust performance, and effectively

discriminate between duplicate and non-duplicate questions, thereby instilling confidence in its deployment for requirement analysis tasks.

4.3.6 ROC Curve and Confusion Matrix of KNN Classifier

An AUC (Area Under the Curve) score of 0.92 for the test set is indicative of the model's strong generalization ability to unseen data. This high AUC value suggests that the model effectively discriminates between duplicate and non-duplicate questions with a strong separability. It demonstrates that the model's performance is robust and not overfitting to the training data.

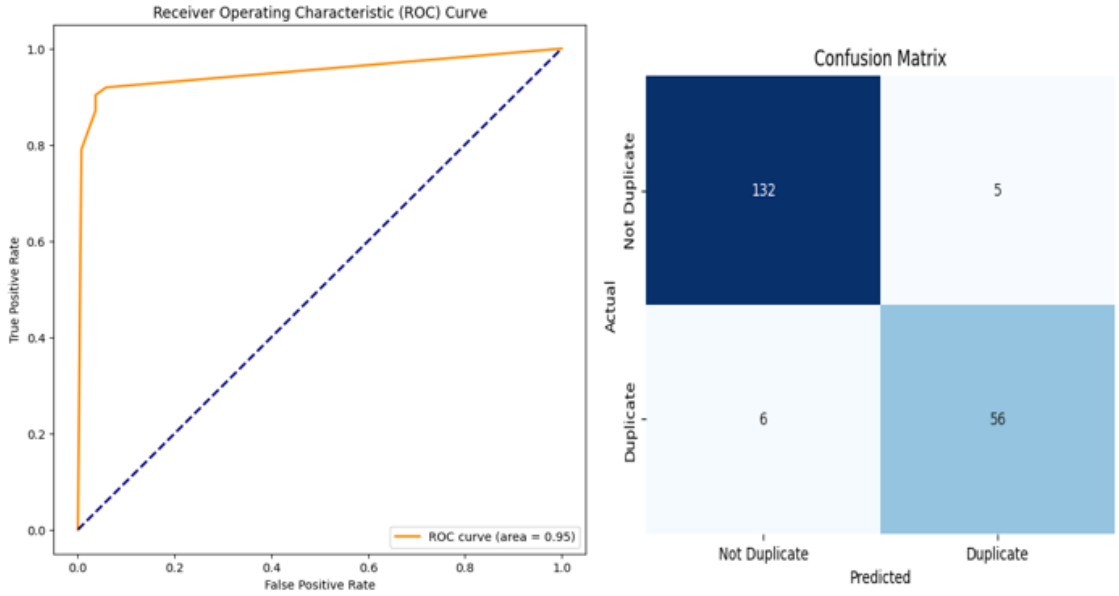


Figure 4.5: ROC Curve and Confusion Matrix of KNN Classifier

A high AUC score indicates that the model correctly ranks a higher proportion of duplicate questions higher than non-duplicate questions, which is desirable in requirement analysis tasks. Moreover, the AUC score of 0.92 suggests that the model maintains its performance across different threshold settings, further validating its effectiveness in distinguishing between duplicates and non-duplicates.Overall, the AUC score of 0.92 provides strong evidence of the model's ability to generalize well and make accurate predictions on unseen data. It instills confidence in the model's reliability and suitability for practical deployment in requirement analysis tasks, contributing to improved decision-making processes in software development projects.

4.3.7 Comparison between Cosine Similarity and Euclidian distance

The pie charts illustrate the distribution of similarity scores using two different methods such as Euclidean distance and cosine similarity. Each chart categorizes the scores into four distinct ranges: less than 50%, between 50% and 70%, between 70% and 90%, and between 90% and 100%.

For Euclidean distance, the distribution shows that a significant portion of the similarity scores falls within the 50-70% range, comprising 74.3% of the total scores. This indicates that the majority of pairs have moderate similarity. In contrast, 25.7% of the scores are below 50%, indicating a lower similarity for these pairs. Notably, there are no scores in the higher similarity ranges of 70-90% and 90-100%, both of which account for 0.0% of the distribution. This pattern suggests that Euclidean distance tends to group most scores in the moderate similarity range and does not effectively differentiate among highly similar pairs.

On the other hand, the distribution for cosine similarity shows a slightly more varied pattern. Similar to Euclidean distance, 25.7% of the scores are below 50%, indicating that a comparable proportion of pairs are identified as having low similarity. The largest category remains the 50-70% range, which includes 70.7% of the scores. However, unlike Euclidean distance, cosine similarity identifies 3.6% of the scores within the 70-90% range, suggesting that it can capture higher similarity relationships that Euclidean distance might overlook. The 90-100% range remains negligible, with 0.0% of the scores, similar to the Euclidean distance method.

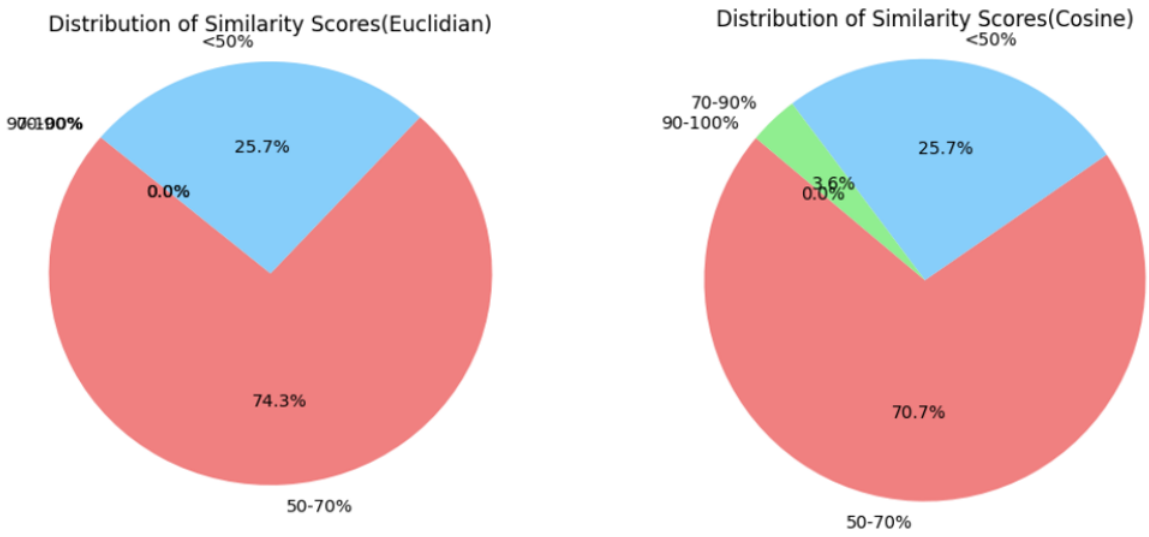


Figure 4.6: Comparison between similarity metrics

Comparing the two methods, both show a substantial portion of scores in the 50-70% range and a quarter of scores below 50%, indicating a general agreement on low to moderate similarity. However, cosine similarity offers a slight advantage by differentiating higher sim-

ilarity scores within the 70-90% range, where Euclidean distance fails to identify any scores. This difference indicates that cosine similarity might be more sensitive in distinguishing higher degrees of similarity between pairs, providing a more nuanced understanding in such cases. Overall, these insights highlight the varying capabilities of Euclidean distance and cosine similarity in mapping requirement similarity, with each method offering distinct advantages based on the distribution of similarity scores.

4.3.8 Streamlit

A user-friendly graphical interface has been developed using the Streamlit library to facilitate the exploration of semantically similar requirements. The interface commences with a secure login page where users are prompted to enter their credentials to access the system. Upon successful authentication, users are granted access to the main functionality. Here, they can input their requirements and initiate the search for semantically similar ones. Leveraging the intuitive design of Streamlit, users can seamlessly navigate through the interface, allowing for efficient and effective requirement analysis. This GUI-based approach enhances user experience and promotes accessibility, ultimately facilitating better decision-making in software development projects

Furthermore, Streamlit simplifies the process of sharing and deploying applications. Users can quickly deploy their interfaces to the web or share them with others via URLs, without the need for complex setup or infrastructure. This accessibility and ease of deployment make it straightforward for users to collaborate on projects, share insights, and showcase their work to a wider audience.

Streamlit also provides robust support for machine learning models, allowing users to seamlessly integrate models and pipelines into their applications. This enables users to build end-to-end data science applications that include model training, evaluation, and inference, all within the Streamlit environment. With Streamlit's functionalities, users can create intuitive and interactive graphical interfaces for a wide range of data science and analysis tasks, ultimately enhancing user experience and facilitating better decision-making across various domains, including software development projects.

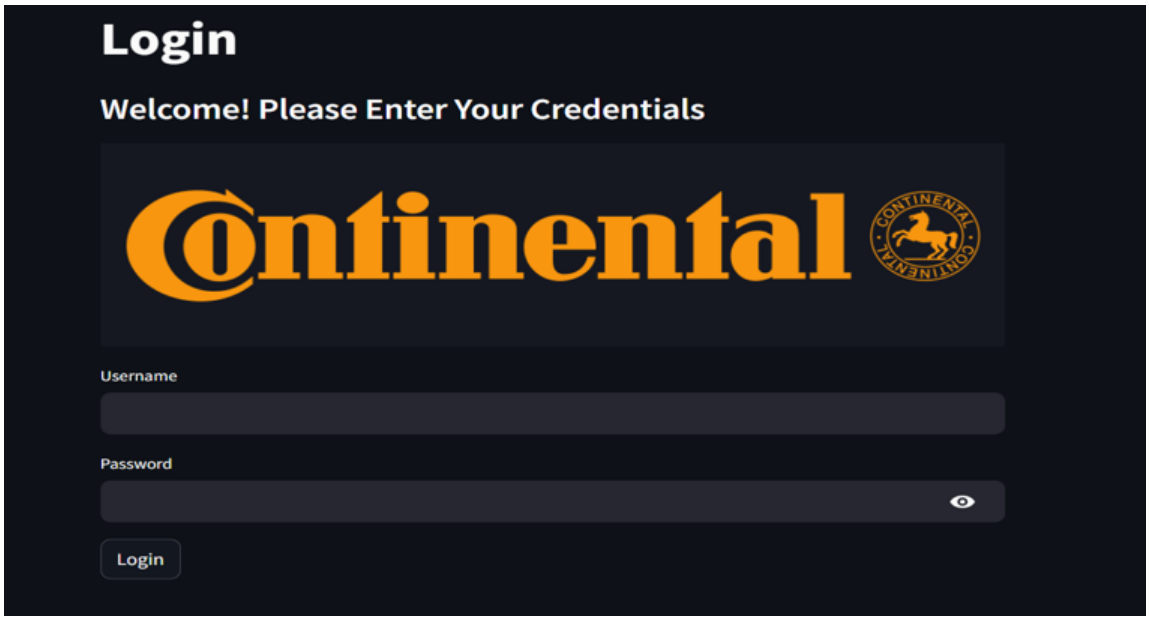


Figure 4.7: Login Page

Once users successfully enter their login credentials, they are directed to another page within the graphical user interface. This new page is dedicated to detecting semantic similarity among requirements. Users can input their requirements or select them from a list, and the system will analyze the text to identify semantically similar requirements. Leveraging advanced natural language processing techniques, the system compares the input requirement with a database of existing requirements to find the most relevant matches. Users are presented with the results, including the degree of similarity and any relevant information associated with the identified requirements. This streamlined process allows users to efficiently explore semantic relationships and make informed decisions in software development projects.

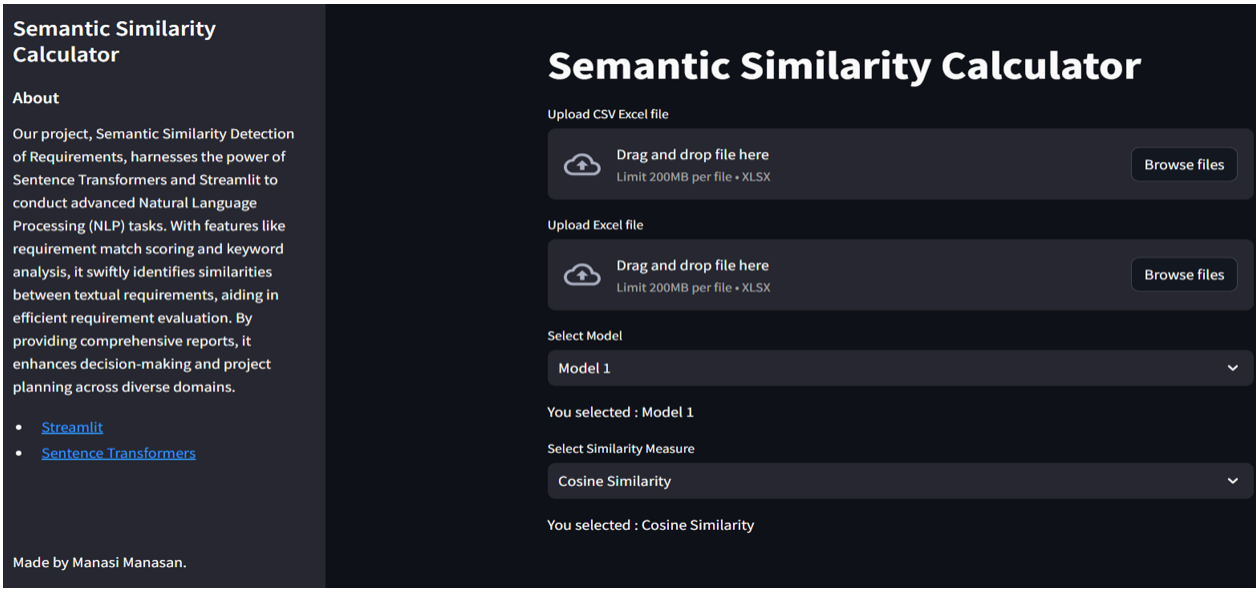
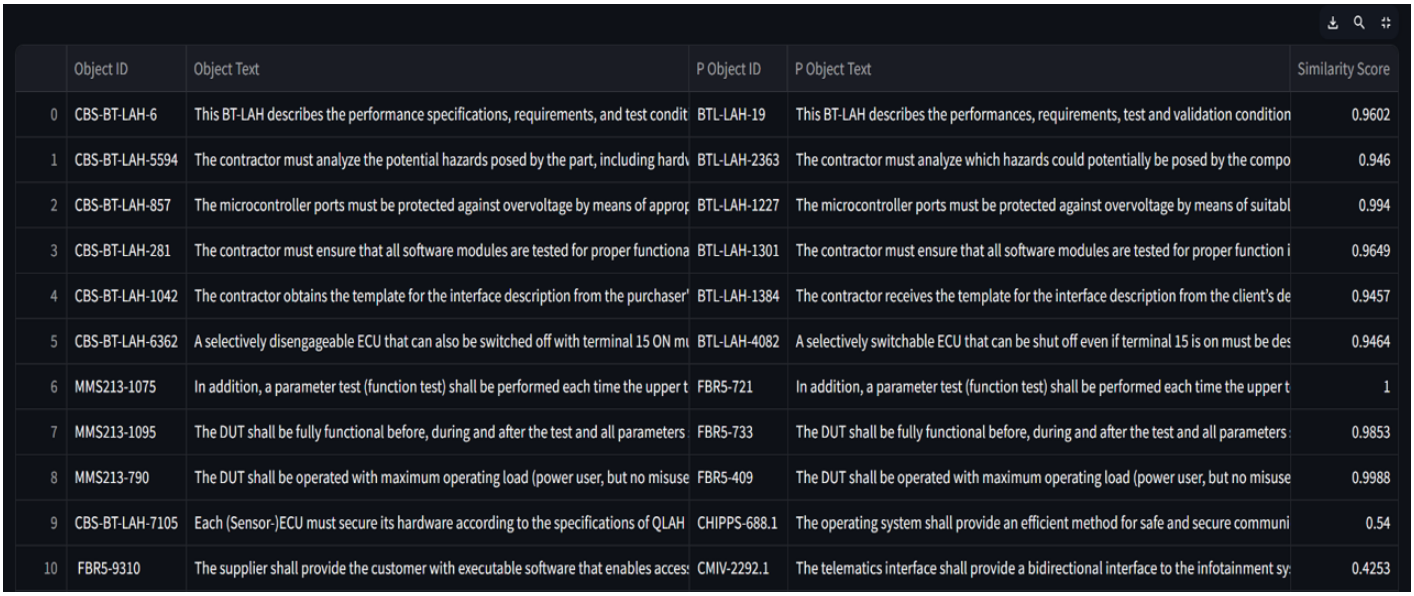


Figure 4.8: Similarity Detection Page

Upon uploading the files containing the requirements to be compared for semantic similarity, users are presented with options to select the desired model and similarity measure. These choices empower users to tailor the analysis to their specific needs and preferences. Once the model and similarity measure have been selected, users can proceed by clicking the "Submit" option.

Upon successful submission, the system initiates the semantic similarity detection process. Leveraging the chosen model and similarity measure, the system compares the uploaded files and identifies semantically similar requirements. This analysis considers various linguistic and contextual factors to ensure accurate and meaningful results.After completion, users receive the results of the semantic similarity detection process. The results are presented in a user-friendly format, providing insights into the identified similar requirements, along with relevant metrics or information to aid in understanding the degree of similarity. This streamlined process empowers users to efficiently explore and interpret semantic relationships among requirements, facilitating informed decision-making in software development projects.



	Object ID	Object Text	P Object ID	P Object Text	Similarity Score
0	CBS-BT-LAH-6	This BT-LAH describes the performance specifications, requirements, and test condit	BTL-LAH-19	This BT-LAH describes the performances, requirements, test and validation condition	0.9602
1	CBS-BT-LAH-5594	The contractor must analyze the potential hazards posed by the part, including hardv	BTL-LAH-2363	The contractor must analyze which hazards could potentially be posed by the compo	0.946
2	CBS-BT-LAH-857	The microcontroller ports must be protected against overvoltage by means of appropr	BTL-LAH-1227	The microcontroller ports must be protected against overvoltage by means of suitabl	0.994
3	CBS-BT-LAH-281	The contractor must ensure that all software modules are tested for proper functiona	BTL-LAH-1301	The contractor must ensure that all software modules are tested for proper function i	0.9649
4	CBS-BT-LAH-1042	The contractor obtains the template for the interface description from the purchaser'	BTL-LAH-1384	The contractor receives the template for the interface description from the client's de	0.9457
5	CBS-BT-LAH-6362	A selectively disengageable ECU that can also be switched off with terminal 15 ON mu	BTL-LAH-4082	A selectively switchable ECU that can be shut off even if terminal 15 is on must be des	0.9464
6	MMS213-1075	In addition, a parameter test (function test) shall be performed each time the upper t	FBR5-721	In addition, a parameter test (function test) shall be performed each time the upper t	1
7	MMS213-1095	The DUT shall be fully functional before, during and after the test and all parameters	FBR5-733	The DUT shall be fully functional before, during and after the test and all parameters	0.9853
8	MMS213-790	The DUT shall be operated with maximum operating load (power user, but no misuse	FBR5-409	The DUT shall be operated with maximum operating load (power user, but no misuse	0.9988
9	CBS-BT-LAH-7105	Each (Sensor-)ECU must secure its hardware according to the specifications of QLAH	CHIPPS-688.1	The operating system shall provide an efficient method for safe and secure communi	0.54
10	FBR5-9310	The supplier shall provide the customer with executable software that enables acces	CMIV-2292.1	The telematics interface shall provide a bidirectional interface to the infotainment sy	0.4253

Figure 4.9: Result of Semantic Similarity Requirements

Once the result were obtained,the user can download in their local personal computer for further process.

Chapter 5

Conclusion and Future Scope

In conclusion, the implementation of Sentence Transformer-based models in requirement similarity mapping systems has indeed marked a significant leap forward in enhancing both accuracy and efficiency. By harnessing the power of transfer learning, particularly through Sentence Transformers, this study showcases a novel approach to requirement analysis that can greatly benefit various industries and domains reliant on precise requirement understanding.In the realm of natural language processing, the distinction between 'all-MiniLM-L6-v2' and 'paraphrase-MiniLM-L6-v2' models is noteworthy. In terms of user accessibility and interface design, the utilization of the open-source Python library 'Streamlit' to build a graphical user interface (GUI) for requirement similarity mapping adds another layer of functionality and user-friendliness to the overall system. This choice not only simplifies the process of interacting with the model but also enhances its usability across various stakeholders, including project managers, developers, and quality assurance teams.

Moreover, beyond the immediate benefits witnessed in requirement analysis, the implementation of Sentence Transformer-based models holds promise for broader applications within the realm of natural language understanding. As these models continue to evolve and adapt to diverse datasets and linguistic contexts, their potential to streamline various tasks across industries becomes increasingly apparent. From automated summarization and sentiment analysis to chatbot development and document classification, the versatility of these models opens avenues for improved efficiency and accuracy in countless use cases. By leveraging the power of transfer learning and fine-tuning techniques, organizations can not only expedite their workflows but also gain deeper insights from textual data, ultimately driving informed decision-making and innovation at scale. Thus, the impact of Sentence Transformer-based models extends far beyond requirement analysis, offering a transformative paradigm for natural language processing across domains.

Through rigorous evaluation, it was observed that 'all-MiniLM-L6-v2' consistently outperformed 'paraphrase-MiniLM-L6-v2', boasting impressive metrics such as a remarkable accuracy rate of 92%, coupled with a precision of 95%, recall of 82%, and an F1 score of 88%. These statistics underscore the robustness and reliability of the chosen architecture, affirming its suitability for semantic similarity detection across a diverse array of requirements.

So the successful amalgamation of advanced machine learning techniques, meticulous

evaluation methodologies, and intuitive user interface design underscores the transformative potential of Sentence Transformer-based models in revolutionizing requirement analysis and semantic similarity detection. As industries continue to evolve and adapt to the demands of an increasingly complex landscape, the adoption of such innovative approaches will undoubtedly play a pivotal role in driving efficiency, fostering innovation, and ensuring the successful delivery of projects and products alike.

5.1 Future Scope

Looking ahead, there are several avenues for future research and development in this field

- Integration with Requirement Management Systems: Explore integration possibilities with existing requirement management systems to provide real-time semantic similarity checks during requirement input or modification.Integrating Sentence Transformer-based models with requirement management systems can revolutionize the way requirements are handled in software development projects. Real-time semantic similarity checks during requirement input or modification can ensure consistency, reduce errors, and improve overall quality. By seamlessly embedding these capabilities into existing workflows, teams can streamline the requirement gathering and validation process, leading to more efficient project delivery.
- Multi-language Support: Extending the model's capabilities to support multiple languages is crucial for its widespread adoption in multinational or diverse linguistic environments. By training the model on diverse language corpora and fine-tuning it to handle linguistic nuances, researchers can enhance its effectiveness across a broader range of languages. This expansion in language support not only improves accessibility but also promotes inclusivity and enables organizations to cater to diverse user bases and global markets effectively.
- Model Fine-Tuning: While pre-trained models offer a strong foundation, fine-tuning is essential to adapt them to specific datasets and tasks. Unfreezing certain layers of the model and fine-tuning them on domain-specific or task-specific data can significantly enhance performance. This iterative process of fine-tuning allows the model to learn from new examples and refine its understanding, leading to improved accuracy and relevance in semantic similarity detection tasks.
- Ensemble Models: Ensemble learning, which involves combining predictions from multiple models, offers a promising approach to further enhance the robustness and performance of semantic similarity detection systems. By leveraging a diverse set of models, each with its own strengths and weaknesses, ensemble methods can mitigate individual model biases and uncertainties, resulting in more reliable and accurate predictions. This approach also enables the incorporation of complementary features and representations, thereby capturing a broader range of semantic relationships and nuances present in textual data.
- Security and Privacy Enhancements: As with any system dealing with sensitive data, security and privacy considerations are paramount. When implementing semantic similarity mapping systems within requirement management frameworks, it's crucial to

prioritize security measures to safeguard against unauthorized access, data breaches, and malicious attacks. Implementing encryption protocols, access controls, and robust authentication mechanisms can help mitigate security risks and ensure the confidentiality and integrity of requirement data. Additionally, adherence to privacy regulations and best practices, such as data anonymization and minimization, further enhances user trust and compliance with legal requirements. By proactively addressing security and privacy concerns, organizations can deploy semantic similarity mapping systems with confidence, knowing that sensitive information is adequately protected.

References

[1] Mansoor, Muhammad, et al. "Deep learning based semantic similarity detection using text data." Information Technology and Control 49.4 (2020): 495-510.

[2] Das, Souvick, et al. "Sentence embedding models for similarity detection of software requirements." SN Computer Science 2 (2021): 1-11.

[3] Vrbanec, Tedo, and Ana Meštrović. "Comparison study of unsupervised paraphrase detection: Deep learning—The key for semantic similarity detection." Expert Systems 40.9 (2023): e13386.

[4] Agarwala, Saurabh, Aniketh Anagawadi, and Ram Mohana Reddy Guddeti. "Detecting Semantic Similarity of documents using Natural Language Processing." Information Technology and Control 189 (2021): 128-135.

[5] Alshammeri, Menwa, Eric Atwell, and Mhd ammar Alsalka. "Detecting semantic-based similarity between verses of the Quran with Doc2vec." Knowledge-Based Systems 189 (2021): 351-358.

[6] Agarwal, B., Ramampiaro, H., Langseth, H., Ruocco, M. A Deep Network Model for Paraphrase Detection in Short Text Messages. Information Processing Management,2018, 54(6), 922-937. https://doi.org/10.1016/j.ipm.2018.06.005

[7] Arora, S., Liang, Y., Ma, T. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. ICLR, 2017.

[8] Arshad, H., Khan, M. A., Sharif, M. I., Yasmin, M., Tavares,J. M. R., Zhang, Y. D., Satapathy, S. C. A Multilevel Paradigm for Deep Convolutional Neural Network Features Selection with an Application to Human Gait Recognition. Expert Systems, 2020, e12541. https://doi.org/10.1111/exsy.12541

[9] Aurangzeb, K., F. Akmal, M. A. Khan, M. Sharif, Javed,M. Y. Advanced Machine Learning Algorithm Based System for Crops Leaf Diseases Recognition. In 2020 6th IEEE Conference on Data Science and Machine Learning Applications (CDMA), 2020. https://doi.org/10.1109/CDMA47397.2020.00031

[10] Bao, W., Bao, W., Du, J., Yang, Y., Zhao, X. Attentive Siamese LSTM Network for Semantic Textual Similarity Measure. In 2018 IEEE International Conference on Asian Language Processing (IALP).2018.https://doi.org/10.1109/IALP.2018.8629212

[11] Batool, F. E., Attique, M., Sharif, M., Javed, K., Nazir, M., Abbasi, A. A., Iqbal, Z., Riaz, N. Offline Signature Verification System: A Novel Technique of Fusion of GLCM

and Geometric Features Using SVM. Multimedia Tools and Applications, 2020, 1-20. https://doi.org/10.1007/ s11042-020-08851-4

[12] Bogdanova, D., dos Santos, C., Barbosa, L., Zadrozny, B. Detecting Semantically Equivalent Questions in Online User Forums. In Proceedings of the Nineteenth Conference on Computational Natural Language Learning, 2015. https://doi.org/10.18653/v1/K15-1013