TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A PROJECT REPORT
ON
TEXT SIMILARITY IDENTIFICATION WITH THE REFERENCE DATABASE

Submitted By:
Bishad Koju
Gaurav Jyakhwa
Kriti Nyoupane
Luna Manandhar

Submitted To:
LABORATORY FOR ICT RESEARCH AND DEVELOPMENT
LALITPUR, NEPAL

(March 14, 2023)

Acknowledgement

First and foremost, we would like to convey our deep gratitude to all of our teachers for their consistent guidance and invaluable support. It would have been a challenging road for us without their great supervision and advice.

Also, we would like to thank the Laboratory for ICT Research and Development, Pulchowk Campus, for providing us with the opportunity to participate in a collaborative project that has allowed us to apply the knowledge we have gained over the years, which has greatly enhanced our knowledge and broadened our horizon. We'd also like to thank all of our friends who have supported us with this project, both directly and indirectly.

Authors:
Bishad Koju
Gaurav Jyakhwa
Kriti Nyoupane
Luna Manandhar

Abstract

Similarity detection, a fundamental task in Natural Language Understanding, is used to detect similarity between words, sentences, paragraphs, and documents that is applicable in information retrieval, document clustering, word-sense disambiguation, automatic essay scoring, short answer grading, machine translation and text summarization. This report proposes a method to capture the semantics of sentences and detect the similarity between questions using Sentence Transformer, a state-of-the-art model. Two questions may be asking the same thing but may vary in word choice and sentence structure. Most of the techniques to detect the similarity in texts uses Bag of Words, TfIdf which fails to capture the semantics of the sentence.

This project detects the semantic similarity in questions from the reference database based on the transformer model distilbert-nli-mean-tokens, smaller and faster than BERT which was pretrained on the PAN corpus in a self-supervised fashion, using the BERT base model as a teacher. Quora questions pair dataset, consisting of more than four hundred thousand questions, out of which ten thousand questions were used as the reference database. The search for semantically similar questions in the reference database is done using a faster search approach, FAISS.

Keywords: BERT, Sentence Transformer, Similarity search, Semantic similarity, FAISS

Contents

List of Figures

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag of Words
CPU	Central Processing Unit
FAISS	Facebook AI Similarity Search
IOE	Institute of Engineering
KNN	K Nearest Neighbour
LSA	Latent Semantic Analysis
MCW	Most Common Word
MLM	Masked Language Model
NLP	Natural Language Processing
NSP	Next Sentence Prediction
RNN	Recurrent Neural Network
SBERT	Sentence BERT
SVM	Support Vector Machines

1 INTRODUCTION

1.1 Background

It is undeniable that the questions in most of the examinations have a high tendency to repeat. Any question is said to be repeated compared with the previous questions if the question expresses the same intent. i.e. any valid answer to one question is also a valid answer to the other question. E.g. "How far is the sun from the earth?" expresses the same meaning as the question "What is the distance between the sun and the earth?". These two questions have different structures but their semantics are the same. Many of the questions being asked at any given time have already been asked, usually with different wording or phrasing.

The tendency of questions being repeated has developed a habit of mugging up past questions among students. This has clearly reduced the efficiency of the evaluation system. Hence, a system to assist the teacher by automatically detecting the repetition of questions is the utmost requirement.The goal of our project is to detect such repetition in questions.

1.2 Objectives

The objectives of the project are:

- To provide a word add-in that provides an interface for the users to check the similarity between the texts.
- To provide an web portal using which admin can upload the files and update the database.
- To develop a system using a transformer-based model, distilbert-base-uncased, capable of automatically detecting semantically similar questions.
- To deliver an end product that can be used in real life application.

1.3 Scope of Project

Text similarity detection has been used in several tasks such as information retrieval, document clustering, questions generation, question answering, etc. Using this technique to detect similarity between questions has various usages.

- The project can be used to assist teaching staff to create effective questions for better evaluation.
- The project can be used to locate and address the pre answered questions on any online platform.
- The project can be extended to find the uniformity over the questions asked in different universities provided the same course.
- The deep end modification in this project can be used to detect similarities between questions in various languages.

2 LITERATURE REVIEW

Plagiarism detection is gaining increasing importance due to requirements for integrity in education. There are several plagiarism types according to the similarity of the plagiarised passage with the source document. Content-based approach and stylometry-based approach are the most used approaches for analysing plagiarism on text-based documents [1].

The content-based approach focuses on semantic features of the document while the stylometry-based approach focuses on the grammatical style of the document [2]. Content-based approach is widely used to detect duplicate documents [3] [4]. The content of a document is represented in several methods such as the sequence of a bag of words, vector space model, tree, and graph.

In [5] a new integrated approach, which is based on four popular techniques: (i) Bag of Words (BOW), a robust and well-performing feature generation model that breaks a document into all of its unique words and calculates the frequency of each word, (ii) Latent Semantic Analysis (LSA) that offers a quantitative representation of a semantic domain, (iii)Stylometry which implements a statistical analysis of the text and aims to characterise author writing style based on extracted features such as MCW, sentence length, and content words, (iv) Support Vector Machines (SVM), a supervised learning technique that can classify items based on features is proposed.

To compute the similarity between the sentences, they have to be converted into vectors so that the similarity measures can be calculated. The recent model BERT[4] out of the box Maps the sentences into vectors which are 3 unsuitable with common similarity measures. Reimers et al. showed in [6] that BERT out of the box performed worse than the average GloVe embeddings. So, they have proposed Sentence-BERT (SBERT), a pre-trained BERT network that uses Siamese and triplet network architectures to generate semantically relevant sentence embeddings that can be compared using cosine-similarity. Their approach achieved a significant improvement over other state-of-the-art sentence embeddings on var-

ious common benchmarks.

In [7] Approximate Similarity Search with FAISS Framework, a novel design to address the problem of significant computational and energy overhead in high dimensional nearest neighbour queries on billion-scale datasets has been implemented based on a hardware-accelerated approximate KNN algorithm built upon FAISS framework (Facebook Artificial Intelligence Similarity Search) using FPGA-OpenCL platforms on the cloud.

3 THEORETICAL BACKGROUND

3.1 Tokenization

In Natural Language Processing(NLP), one of the fundamental tasks is tokenization. Tokenization is the process of converting a sequence of characters into smaller units called tokens. Here, tokens can be either words, characters, or subwords. Some of the punctuations may be removed during tokenization. Tokenization is an important step in the preprocessing of texts in Natural Language Processing. Different libraries are available in Python for tokenization. Some of them are nltk, spacy, gensim, keras, etc. Tokenization can be broadly classified into 3 types – word, character, and subword (n-gram characters) tokenization.

- Word Tokenization: A word-based tokenization algorithm will break the sentence into words. The most common one is splitting based on space.
- Character Tokenization: A character-based tokenization algorithm will break the sentence into characters.
- Subword Tokenization: A subword-based tokenization algorithm will break the sentence into subwords.

For example, consider the sentence "**Learn Tokenization**"

- Word tokenization: ["Learn","Tokenization"]
- Character tokenization: ["L", "e", "a", "r", "n", "T", "o", "k", "e", "n", "i", "z", "a", "t", "i", "o", "n"]
- Subword Tokenization: ["Learn","Token","ization"]

3.2 Transformer Architecture

The Transformer architecture follows an encoder-decoder structure but does not rely on recurrence and convolutions in order to generate an output. It consists

of stacked layer of multihead attention and feed forward networks as shown in Figure 2. Since transformer does not process the text sequentially as in RNN so, positional encoding is used to give information to the model about the relative position in the sequences.

In a nutshell, the encoder's job is to convert an input sequence into a series of continuous representations, which are subsequently fed into a decoder. On the right half of the architecture, the decoder gets the encoder's output along with the decoder's output from the previous time step to create an output sequence.

The scaled dot product attention is calculated as,

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

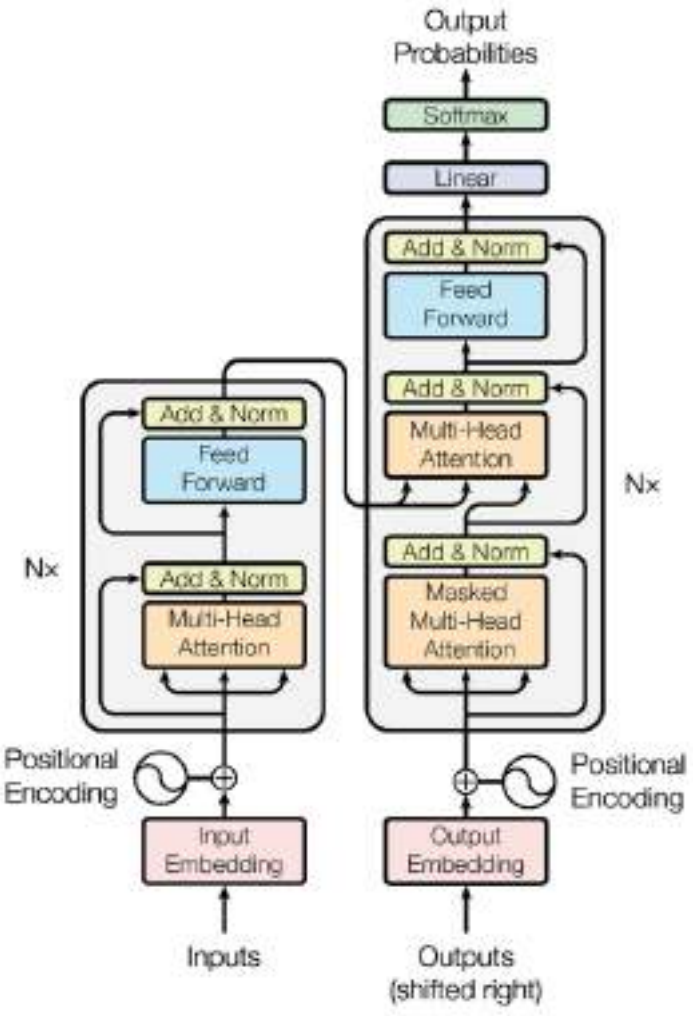

Figure 1: Transformer Architecture

Image Source: Rothman D. (2021), Transformers for Natural Language Processing

where Q is the query matrix(vector representation of one word in the sequence), K are all the keys(vector representation of all the words in the sequence) and V are the values(also, vector representation of all the words in the sequence). The softmax function will output a probability which will decide the value in matrix V to give more attention. Multihead attention is simply a multiple number of scaled dot product attention used to capture different levels of relevance relations of a token.

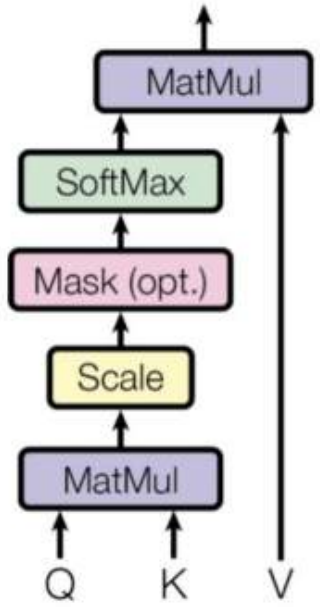

Figure 2: Scaled Dot Product Attention

Image Source: Rothman D. (2021), Transformers for Natural Language Processing

3.3 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a natural language processing model proposed by the Google research team in 2018. It is based on the Transformer architecture and uses the encoder part of the Transformer trained on a large corpus of unlabelled text from Wikipedia and Book corpus. Using this bidirectional capability, BERT is pre-trained on two

different, but related, NLP tasks: Masked Language Modeling and Next Sentence Prediction.

3.3.1 Mask Language Modeling

The objective of Masked Language Model (MLM) training is to hide a word in a sentence and then have the program predict what word has been hidden (masked) based on the hidden word's context.The input tokens were masked in a tricky way to force the model to train longer but produce better results with three methods:

- Surprise the model by not masking a single token on 10% of the dataset
- Surprise the model by replacing the token with a random token on 10% of the dataset
- Replace a token by a [MASK] token on 80% of the dataset

3.3.2 Next Sentence Prediction

The objective of Next Sentence Prediction training is to have the program predict whether two given sentences have a logical, sequential connection or whether their relationship is simply random. Two new tokens were added:

- CLS is a binary classification token added to the beginning of the first sequence to predict if the second sequence follows the first sequence. A positive sample is usually a pair of consecutive sentences taken from a dataset. A negative sample is created using sequences from different documents.
- SEP is a separation token that signals the end of a sequence.

3.4 Sentence Transformer

Sentence Transformers is a Python framework for state-of-the-art sentence, text, and image embedding proposed in the paper Sentence-BERT: Sentence Embed-

dings using Siamese BERT-Networks. This framework computes sentence/text embeddings for more than 100 languages. These embeddings can then be compared e.g. with cosine-similarity to find sentences with similar meaning. This can be useful for semantic textual similarity, semantic search, or paraphrase mining.

3.5 K Nearest Neighbor

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. Most of the time, similar data points are close to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.

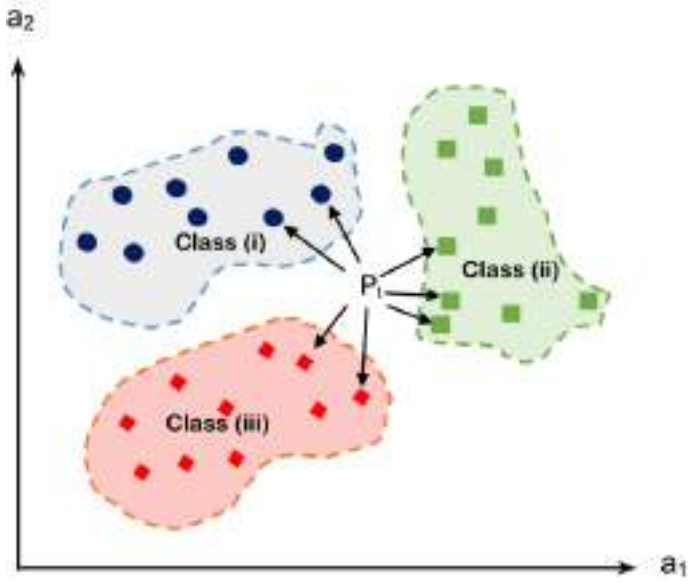

Figure 3: K Nearest Neighbor

3.6 FAISS

FAISS (Facebook AI Similarity Search) is a library that allows developers to quickly search for similar embeddings of multimedia documents. It addresses the

shortcomings of standard query search engines that are specialized for hash-based searches and provides more scalable similarity search functionalities.

3.7 Cosine Similarity

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 4: Cosine Similarity

3.8 Open AI Services

OpenAI provides an API (Application Programming Interface) called the OpenAI API, which allows developers to integrate state-of-the-art artificial intelligence models into their applications. The OpenAI API provides access to a range of language models that can generate human-like text, answer questions, translate languages, and more.

The OpenAI API is designed to be easy to use, with clear documentation and comprehensive guides. Developers can use the API to build a range of applications, from chatbots and virtual assistants to automated content generation and language translation services.

4 METHODOLOGY

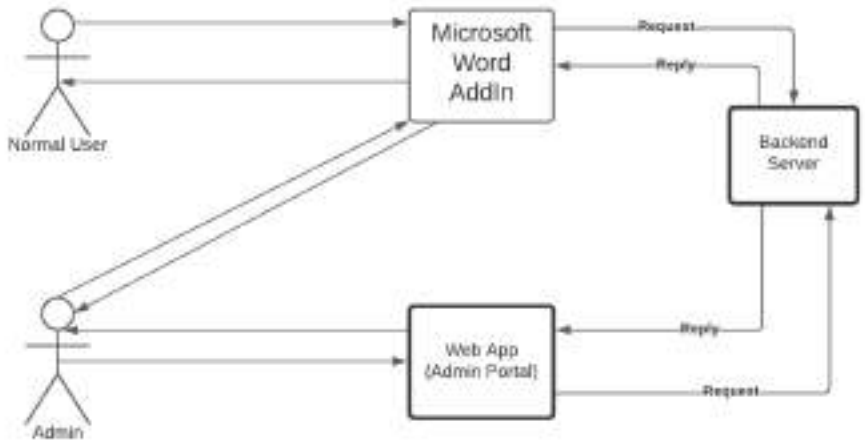

Figure 5: User interaction diagram

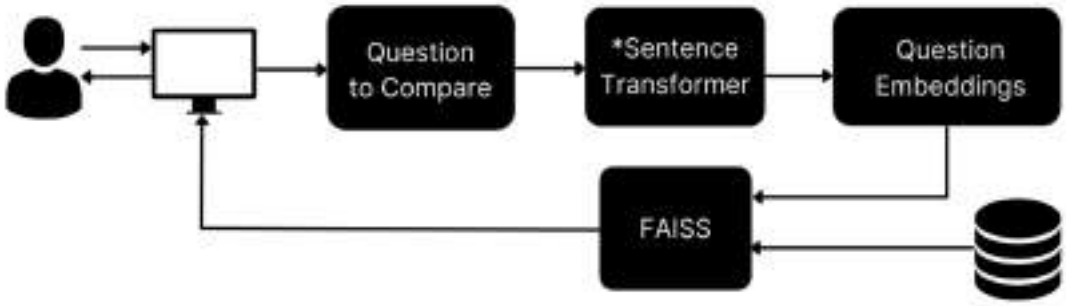

Figure 6: System Block Diagram1

4.1 Dataset Collection

The open-quora dataset from Kaggle consists of over 400,000 lines of potential question duplicate pairs. Each line contains IDs for each question in the pair, the

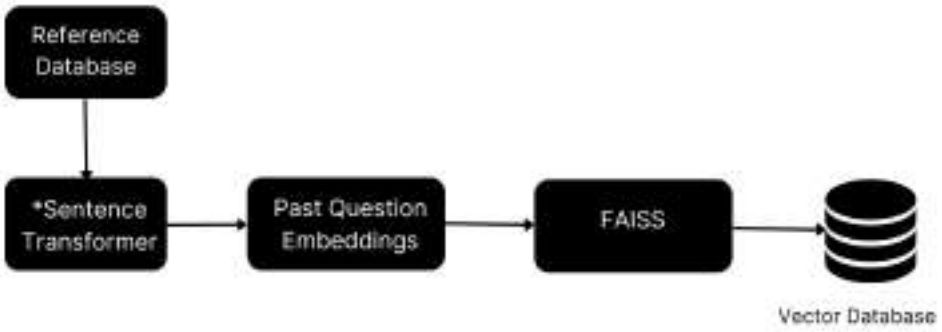

Figure 7: System Block Diagram2

full text for each question, and a binary value that indicates whether the line truly contains a duplicate pair.

id	qid1	qid2	question1	question2	is_duplicate
447	895	896	What are natural numbers?	What is a least natural number?	0
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?	1
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

Figure 8: Quora Question Pair dataset

Only the required attributes 'question1', 'question2', and 'is_duplicate' were used, and the rest were dropped for the training. While the first 10,000 questions of the 'question1' attribute were used to find the embeddings and saved in the database as a CSV file. The CSV files were then serialised, converting an object in the memory to a byte stream that can be stored as a binary file on disk, using pickle. When loaded back to a Python program, this binary file can be deserialized back to a Python object.

The generated embeddings of the questions were then indexed using FAISS and stored it as a vector database. Further, the questions were also stored in the normal database with some metadata information(uploaded date, examination

type and exam year).

4.2 Model

The model, sentence transformer, a state-of-art model was specially built to find the embeddings of sentences capturing the semantics using the self-attention mechanism. The model under the hood consisted of a BERT layer and a pooling layer. BERT produced contextualized word embeddings for all input tokens in the text. As a fixed-sized output representation ($\overrightarrow{u}$) was desired, next comes a pooling layer. Different pooling options were available, the most basic one being mean-pooling: simply averaging all contextualized word embeddings, BERT provided. This gave a fixed 768-dimensional output vector independent of how long the input text was. Also, the sentence transformer provided an option to use a pre-trained model of choice varying according to tasks. 'distilbert-nli-means-token' is one used in this project.

4.3 Similarity Search

When the user enters the question via the interface, the embeddings of the question are obtained using the model. The similarity between the embeddings of the question is searched in the database of questions embeddings which are indexed using FAISS. FAISS uses the K Nearest Neighbor approach to find the nearest neighbor of the question in the given vector dimension making the search super fast. The results obtained i.e. top five semantically similar questions are then displayed to the user via the web interface.

4.4 Word AddIn

One can use this AddIn to find out similar questions in the database.

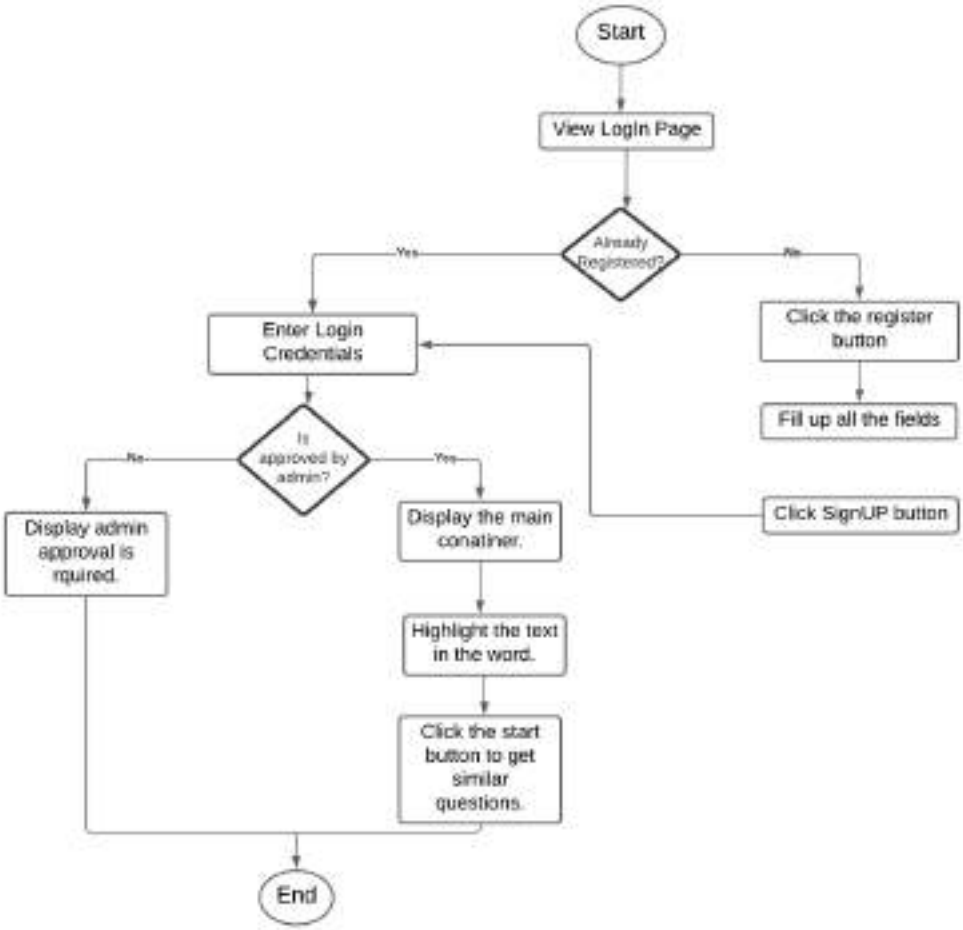

Figure 9: Word AddIn Pipeline

Steps to use Word AddIn

1. Any new user who wants to use the app must register by clicking the signup button.

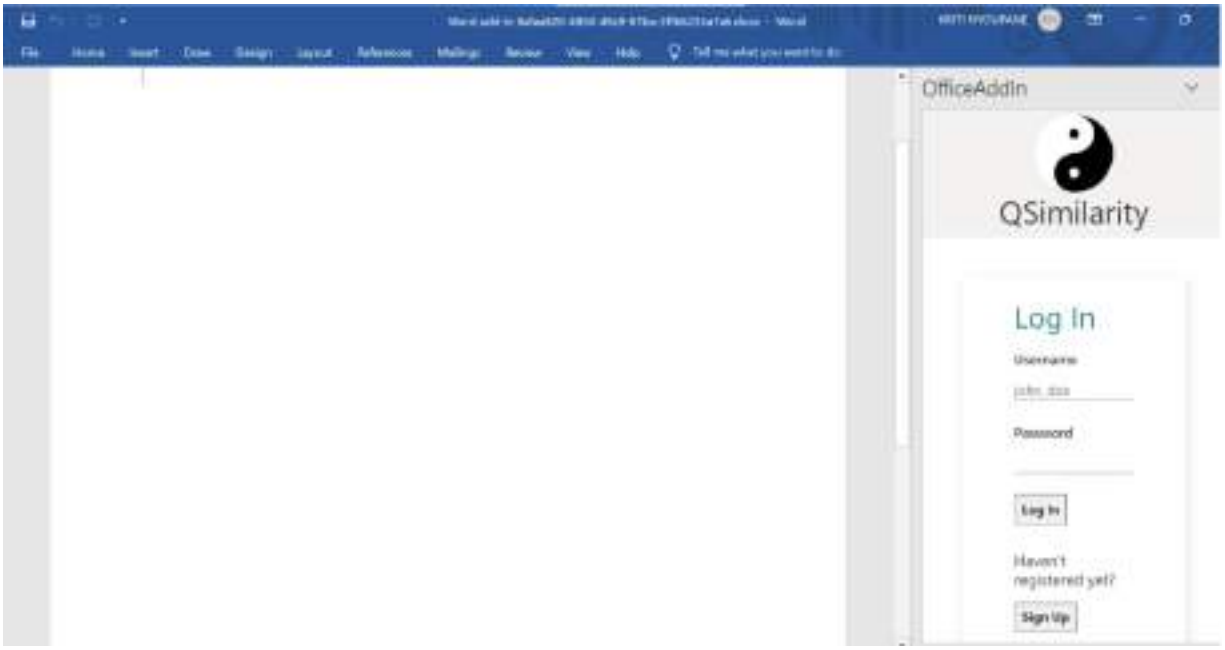

Figure 10: step1

2. After clicking the signup button, the user must fill in all the fields and register.

Figure 11: step3

3. Though the user has registered, one cannot use the service until the user is approved by admin.

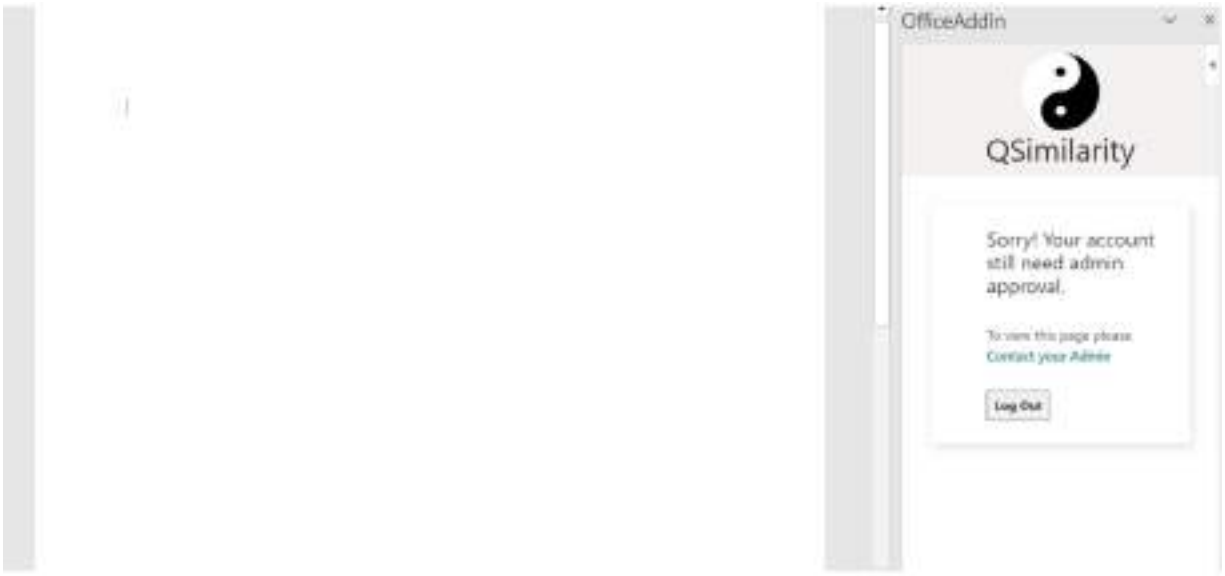

Figure 12: step3

4. Once the user is approved by the admin, the following page is displayed.

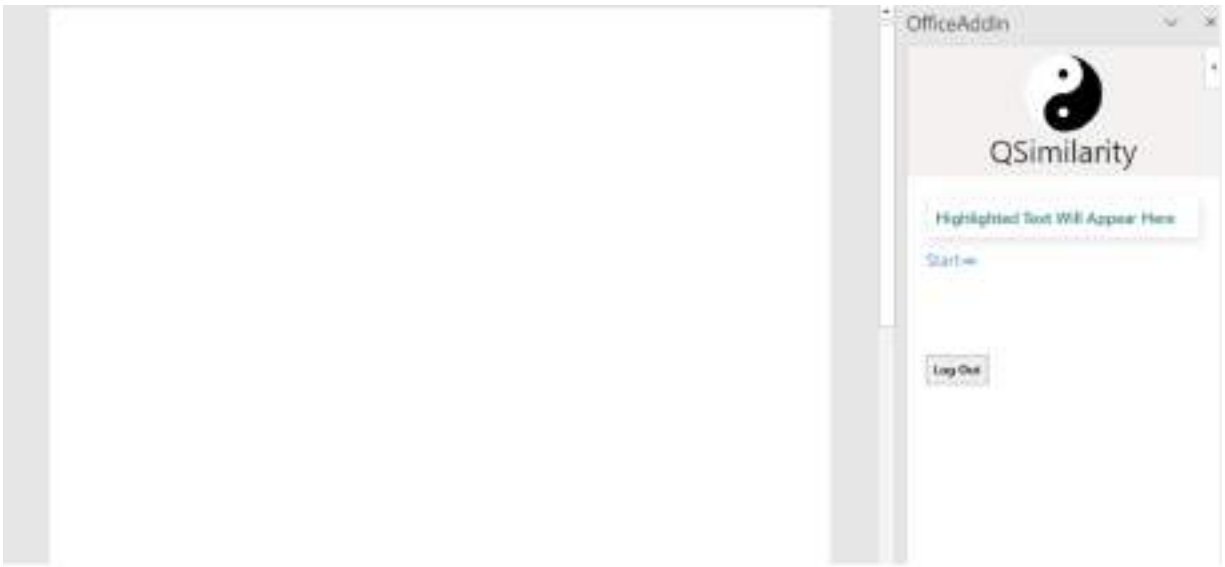

Figure 13: step4

5. The user must highlight the question and click the start button to get similar questions from the database.

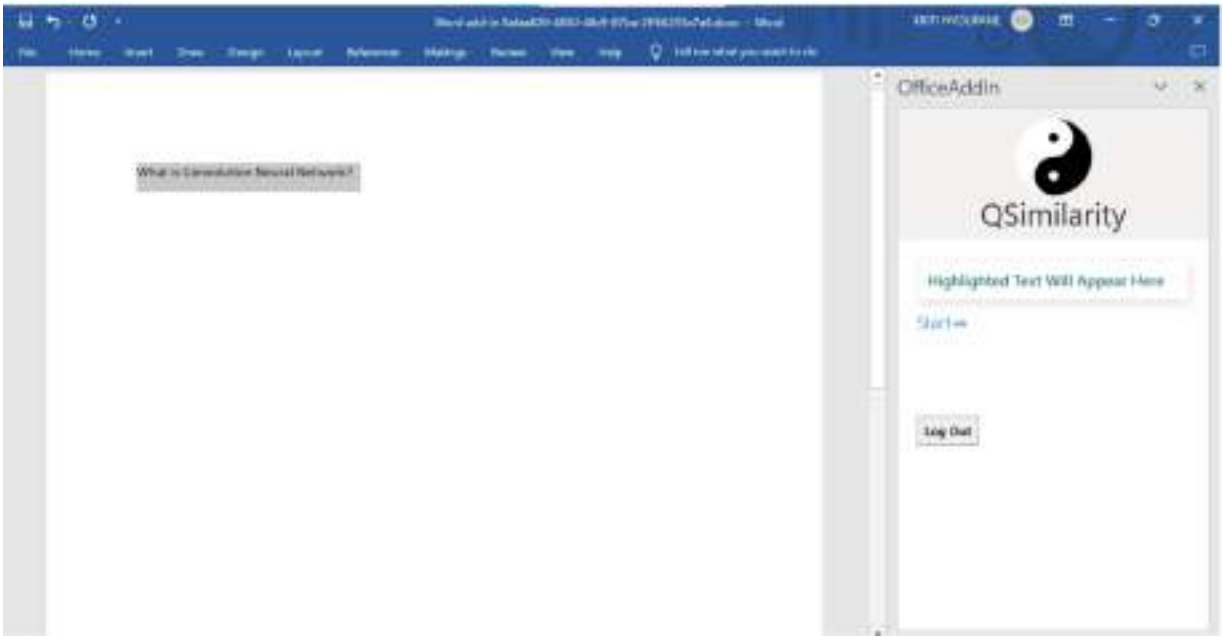

Figure 14: step5

6. Once the user click the start button, user can get the list of similar question with metadata information

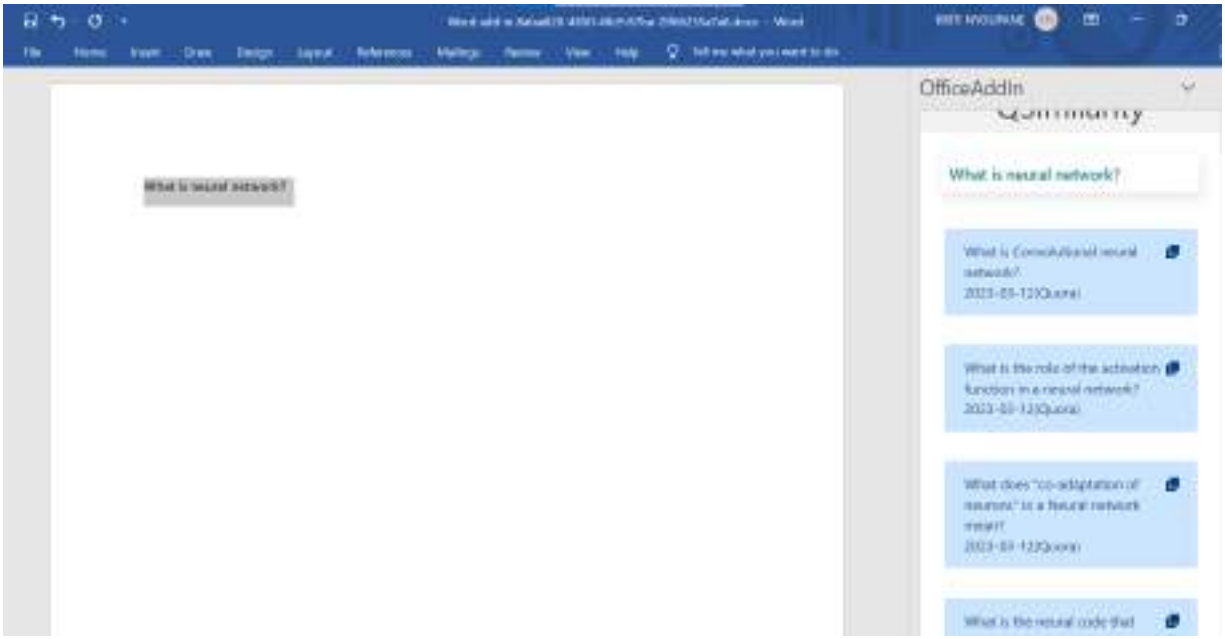

Figure 15: step6

4.5 Paraphrasing

After users highlight a piece of text in the word addIn, they will receive a similar question from the database. Moreover, they can utilize the paraphrase button to rephrase the selected text, which will trigger an OpenAI API call to generate relevant paraphrase questions. However, to access this API, users must first create an account and obtain an API key. Once they have the key, they can access the API via various programming languages like Python, JavaScript, and Ruby. The OpenAI API provides both simple and advanced interfaces, catering to a diverse range of use cases.

1. User can also click the paraphrase button to paraphrase the highlighted question.

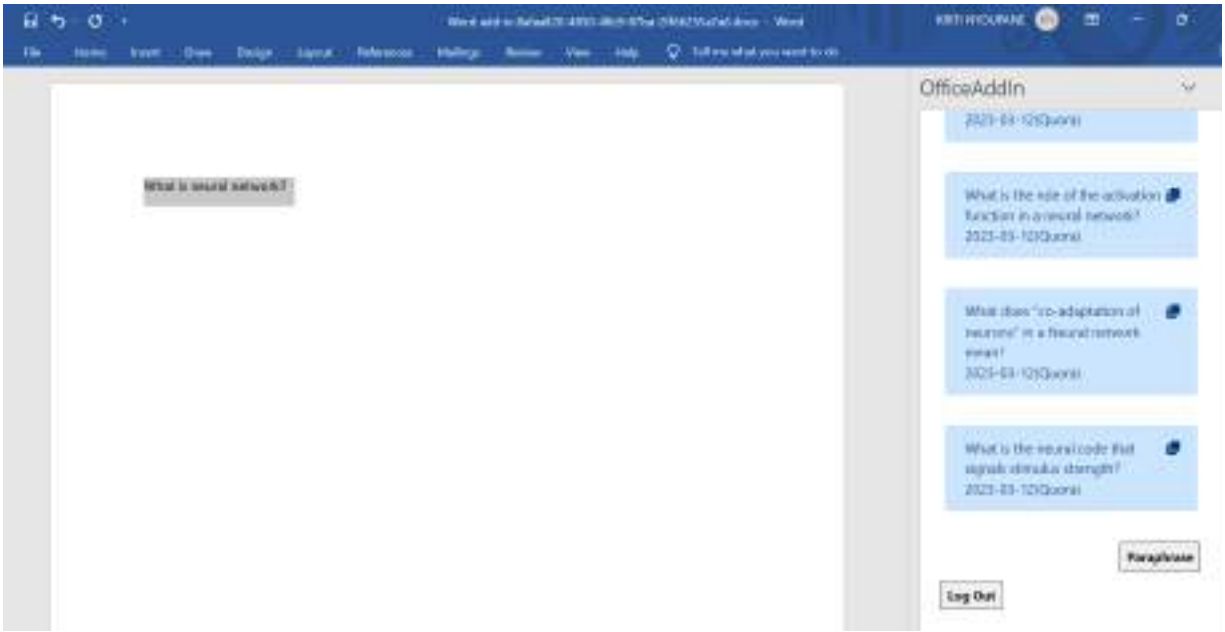

Figure 16: step7

2. Then the result of paraphrase is generated

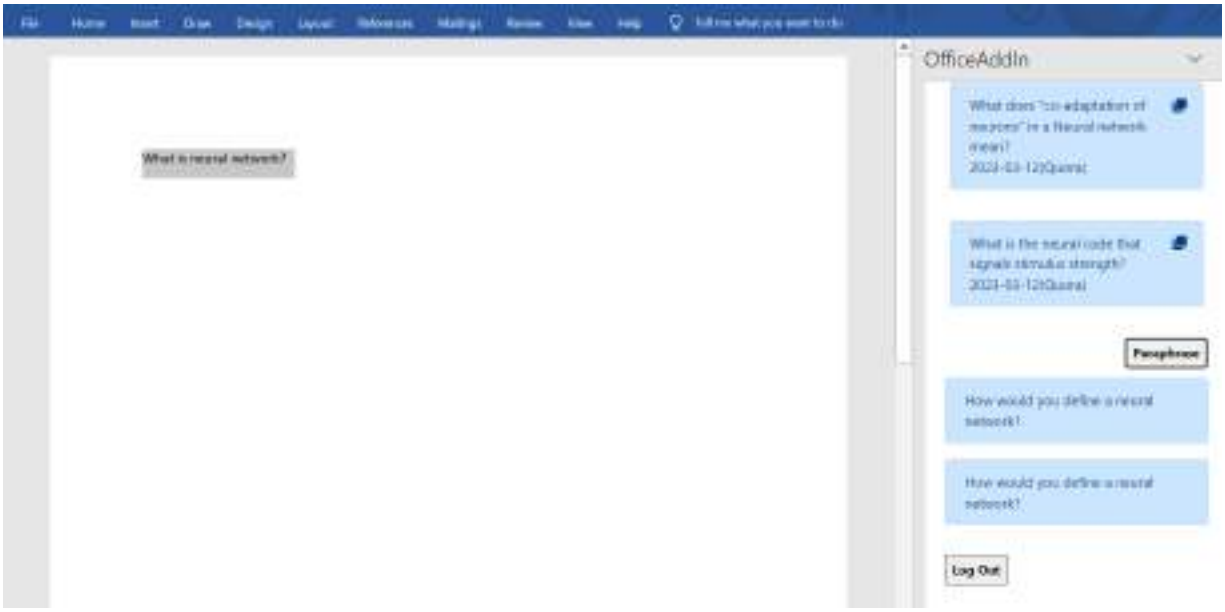

Figure 17: step8

4.6 Web Interface

The admin is granted access to a Web Portal, providing an overview of the server's status, along with summary statistics such as the question count, questions stored in the database, and check-in/check-out times of other users. The portal also facilitates uploading a CSV file containing questions, which is a crucial functionality. Once the CSV file is uploaded, the metadata information and each question within it are parsed and stored individually in the database. Next, the embeddings for each question are generated, and indexing is performed using faiss. The new index is then appended to the existing vector database, resulting in the creation of a new vector database. Once the new vector database is complete, the old one is deleted, and the new database is used for indexing purposes.

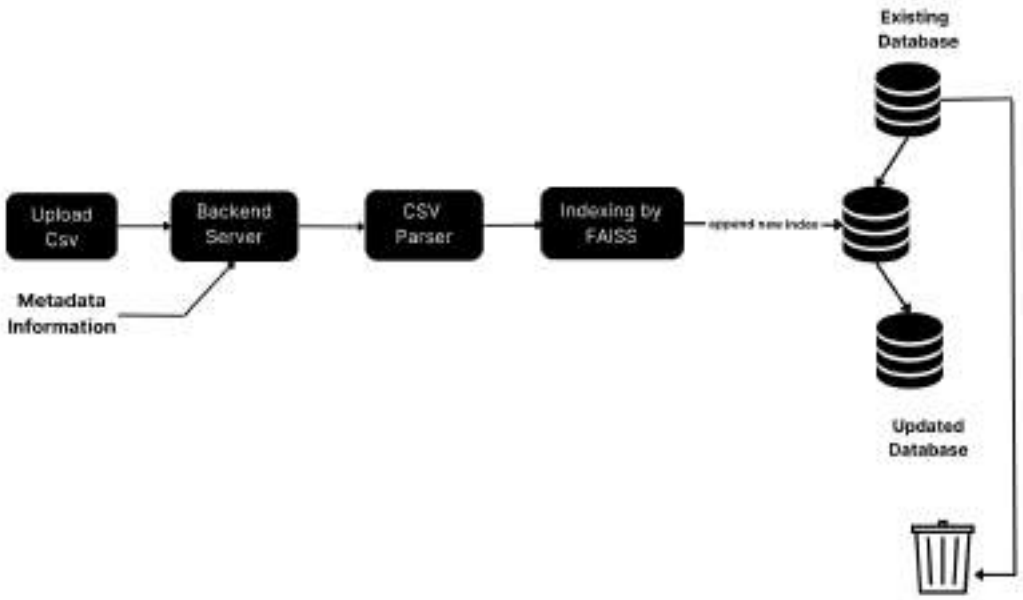

Figure 18: Uploading a CSV file

5 SOFTWARE REQUIREMENTS

5.1 Django

Django is an open source high-level Python web framework that encourages rapid development and clean, pragmatic design built by experienced developers and takes care of much of the hassle of web development. It focuses on writing an app without needing to reinvent the wheel.

5.2 Keras

Keras is an open-source software library that provides a Python interface for an artificial neural network Keras acts as an interface for the TensorFlow library. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

5.3 Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

5.4 Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intu-

itive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

5.5 PyTorch

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs. The packages provided by PyTorch is used to finetune the transformer model in the quora dataset

5.6 React

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It is used for composing complex UIs from small and isolated pieces of code called “components”.

5.7 Faiss-cpu

Faiss is a library for efficient similarity search and clustering of dense vectors. It contains algorithms that search in sets of vectors of any size, up to ones that possibly do not fit in RAM. It also contains supporting code for evaluation and parameter tuning.

6 RESULT AND ANALYSIS

Once the user inputs the question, a list of similar questions was returned within 0.14 seconds. The fast computation was possible because of multithreading i.e. a different thread was used to load the model. Also, the indexing of the embeddings in the database and the use of FAISS which is based on an approximate KNN algorithm made the search faster. The similarity score found using cosine similarity is displayed side by side with the returned semantically similar question indicating how similar the question is with the entered question (1 being the most similar and 0 being the least similar). Also, the metadata like exam year and exam name were displayed along with the results in the word addin'.

The performance of the model was evaluated manually since the natural language is ambiguous, thus there is no proven ground truth regarding the similarity of the given two sentences. The results obtained from the sentence transformer alone were much more satisfactory than from the fine-tuned model. To add on top of it, while fine-tuning, the model using ReLU as an activation function performed well though it consisted of sparse embeddings. In contrast, TanH provided dense embeddings but performed poorly.

7 CONCLUSION

In this project, we have endeavored to develop a text similarity detection system for the Quora dataset questions. To achieve this, we have created a word add-in that enables users to detect the similarity between the question they enter and the questions in the database. To accomplish this, we have utilized the Sentence Transformer, which utilizes a pre-trained model called "DistilBERT-Base-Uncased" to compute the embeddings of the questions while also taking into account their semantics. We then used FAISS to obtain the reference database's embeddings, which were searched to detect similarities with the user's question embeddings. Additionally, we have also created a web portal specifically designed for administrators to upload new questions into the database. This feature will help expand the database and improve the system's accuracy by providing more questions for comparison.

8 LIMITATIONS

This project had to face some limitations due to various factors. Different factors such as the nature of data, size of data, computational power, and resources have their own limitations. Some of the limitations faced by the project are:

- Since the BERT is a heavy model in itself and the sentence transformer built on it is also a heavy model, a great amount of time was needed to train the model.
- The sentence transformer is already trained in a large corpus and highly effective for finding semantic similarity, fine-tuning couldn't improve the result much.

9 FUTURE ENHANCEMENT

As technology and time evolve, undertakings in the field of computer science and information technology require alterations and adaptations. In the future, the following improvements can be made to this project.

- By making suitable dataset, transformer architecture based methods such as BERT can be finetuned for text similarity detection
- The model that is able to find the similarity among the problem statements of Mathematics consists of figures.
- Instead of adding layers on top of the Sentence-transformer, the pretrained model like DistilBertForSequenceClassification could be trained using a different optimizer, tokenizer, and custom functions for training and evaluating.

10 BIBLIOGRAPHY

1. Mohamed El Bachir Menai, "Detection of plagiarism in Arabic document," Information Technology and Computer Science, 80-89 2012.

2. Michael Tschuggnall and Gunther Specht, "Plagiarism detection in text document through grammar analysis of authors".

3. Sudhir D. Salunkhe and S.Z Gawali, "A plagiarism detection mechanism using reinforcement learning," International Journal of Advanced Research in Computer Science and Management Studies, ISSN 2321-7782, 2013.

4. Devlin, K. et al. (2019). BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 4171–4186). Association for Computational Linguistics.

5. M. AlSallal, R. Iqbal, S. Amin, A. James, V. Palade (2016). An Integrated Machine Learning Approach for Extrinsic Plagiarism Detection. In 2016 9th International Conference on Developments in eSystems Engineering (DeSE) (pp. 203-208)

6. Reimers, N., Gurevych, I. (2019). SentenceBERT: Sentence Embeddings using Siamese BERTNetworks. In Proceedings of the 2019 Conference on 11 Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

7. Danopoulos, D., Kachris, C., Soudris, D. (2019). Approximate Similarity Search with FAISS Framework Using FPGAs on the Cloud.

8. Rotchman, D. (2021). Transformers for Natural Language Processing

11 APPENDIX

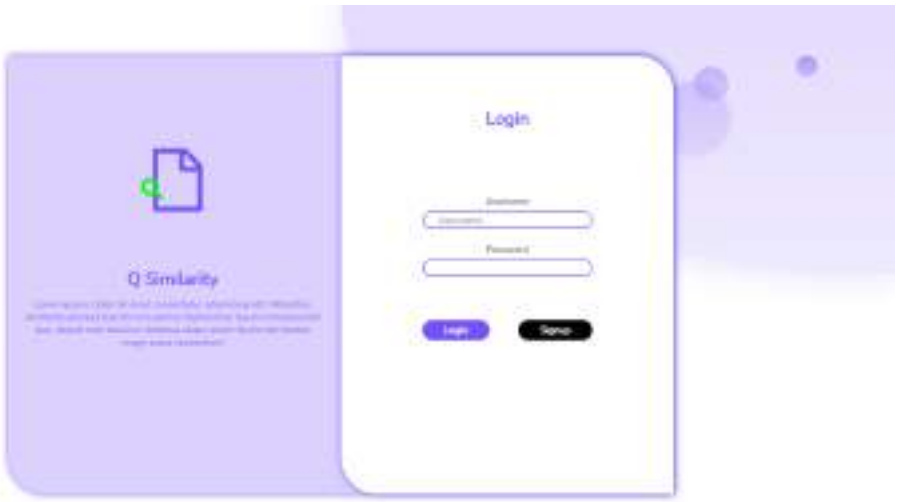

Figure 19: Login Page

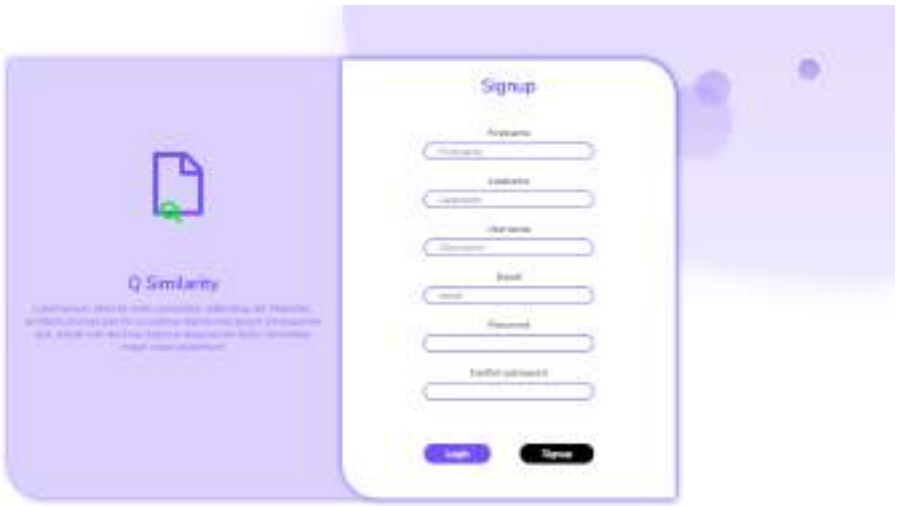

Figure 20: Signup Page

Figure 21: Approval Page

Figure 22: History Page

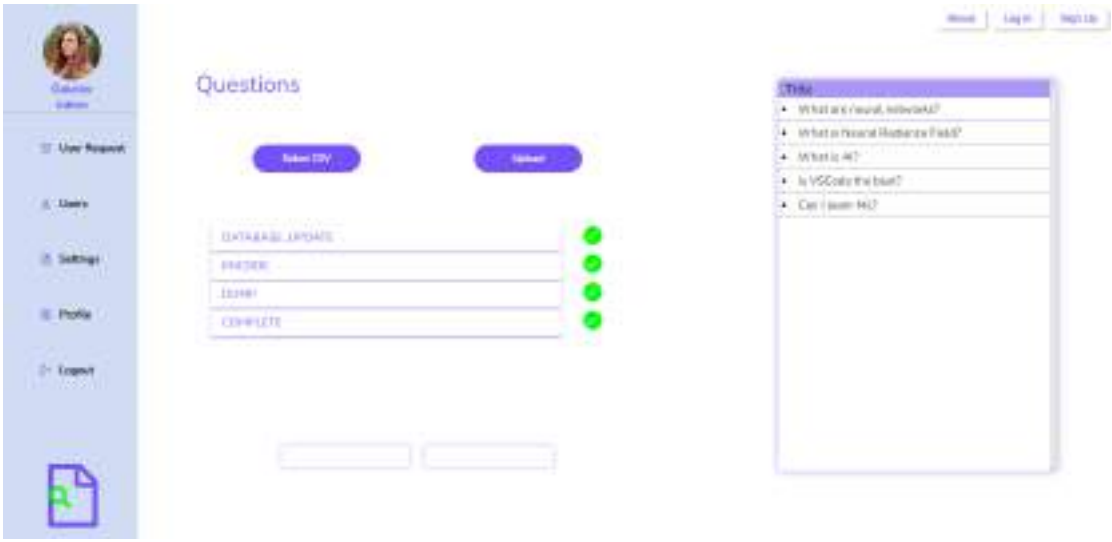

Figure 23: CSV Upload Page

Project Links:

AdminPortalFrontEnd

MSWord AddIn

BackEnd