Paraphrase Identification Using Quora Question Pair

Neha Thakur (017442906), Rutuja Kokate (017453865), Saumya Varshney (017417283), and Vinay Bhati (016853239)

San Jose State University

DATA 270: Data Analyst Process

Dr. Eduardo Chan

May 17, 2024

Abstract

Data is being generated in high volume and velocity in the current world, data storage and data quality is becoming a critical issue. Paraphrase Identification is an important problem since it can help reduce the amount of data being generated and improve overall data quality. To solve this problem, a system needs to be built that takes two texts as input and outputs a label that asserts whether the questions are semantically similar or not. Quora Question Pairs, which consists of more than 400000 labelled samples of semantically similar and dissimilar questions, is used as a dataset. It is prepared by resolving missing data fields, data duplication and data inconsistency issues, followed by tokenization and vector embedding generation using GloVe and FastText. Class imbalance is corrected using a graph-based data augmentation strategy that introduced new similar data samples. Siamese network models are a category of Neural Network architectures that fit this problem perfectly. In past research, models like BiMPM, ESIM and CAS-LSTM have used the same architecture to push the threshold of accuracy on this problem. Siamese network models in this project are designed and trained using RNN, LSTM, GRU and Transformer units as building blocks. These models are then comprehensively evaluated using accuracy, precision, recall, F1 and AUC scores. The Siamese network model built using Transformer proved effective with an accuracy of 0.83 and F1 score of 0.77, proving that the methodology followed in this project is effective at paraphrase identification.

Keywords: paraphrase identification, Siamese networks, deep learning

Introduction

Project Background and Execute Summary

It is in today's world of online knowledge sharing that paraphrase identification has become a significant problem especially when it comes to platforms like Quora, where users often ask questions that are semantically similar. This results not only in redundancy of data but also confuses the user as there will be the same information across so many threads. The core aspect of this task is recognizing the semantic equivalence between pairs of texts regarding plagiarism checks and smooth handling of data. Deep learning models have recently been developed, especially those based on Siamese network structure, to address these problems by capturing the subtle nuances of language semantics. These models use complex architectures like ESIM, CAS-LSTM, TRANS-BLSTM etc., and they combine embeddings and recurrent layers for word interaction within sentences hence giving a statistical basis for understanding language.

Needs and Importance

The urgency behind the need to identify paraphrases is driven by a critical requirement to optimize data storage and improve information quality on digital platforms. In respect of Quora specifically, the vastness of data can result in inefficiencies hence this project intends to use advanced machine learning architectures in content streamlining and user navigation facilitation. As such, this is particularly important in an era where there is so much information that users are overloaded with it resulting in a paradox of choice thereby preventing knowledge from being effectively disseminated. By getting rid of wordiness and presenting material more logically than before, here the intend is not only to make user experience better but also foster precision and clarity as norms for digital communication. Moreover, broader application areas of plagiarism detection have shown its importance as far as maintaining the integrity of academic and

professional texts is concerned. It is inspired by making digital environments more organized, accessible, and reliable whereby high-quality information can be found or contributed by anyone without much effort. The aim is setting new standards for digital interactions through this project thus making platforms like Quora a place where people can acquire knowledge easily in an efficient way.

Target Problem

Massive data sets such as the one provided by Quora have diverse information, but the problem with this is that there are no effective means to identify and differentiate paraphrased content. Like with similar questions, a lot of identical inquiries accumulate. This makes it difficult for users to find relevant and unique information through a confusing mass of answers or queries, thereby adding redundancy to the database. The issue goes beyond just being inconvenient for users but also affects how the platform works generally and its usefulness as an effective resource bank of knowledge.

Motivation and Goals

The project is motivated by the desire to enhance the efficiency of Quora-like digital platforms with a machine learning algorithm for finding similar texts. By cutting back on repetition and making information less cluttered, the intention is to make user experiences better so that search queries end with distinctive and relevant responses. Moreover, this model can be utilized in fighting plagiarism and maintaining content integrity through the web. This project strives for a more ordered, dependable digital atmosphere that speeds up knowledge exchange. The purpose is ultimately to set new records in content management thereby supporting the digital community through the originality of quality content.

Project Approaches and Methods

The project employs the CRISP-DM approach, commonly used in industry, which consists of six stages: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation and Deployment. Hence, when it comes to choosing a suitable methodology for this machine learning project. CRISP-DM Agile methodology is opted for because of its models as well as trial-and-error. Also, agile methodologies are designed to accommodate changing requirements and new information making them ideal for use in a machine learning project. To become an Agile version of CRISP-DM these six phases should be broken into smaller iterations or sprints that can be easily managed by the team. By focusing on one or two CRISP-DM phases per sprint, the team may move forward with the project and maintain flexibility and work together towards achieving the best results.

Deploying Siamese Networks based on RNN, bidirectional LSTM, Transformers and GRU forms the foundation of the approach. These models, well known for their proficiency in processing sequences, will be optimized to identify fine grained differences and similarities between texts. For example, the work of Li et al. (2018) extensively informed the strategy regarding the effectiveness of attention mechanisms in improving word representation and sentence pair modeling. This is crucial to the improvement of RNN-based model's ability to detect paraphrases with high precision.

The aim is to establish new benchmarks regarding paraphrase detection via a comprehensive and multi-dimensional project approach, leading to more orderly, reachable, and reliable information transfer platforms in digital environment.

Project Contributions and Applications

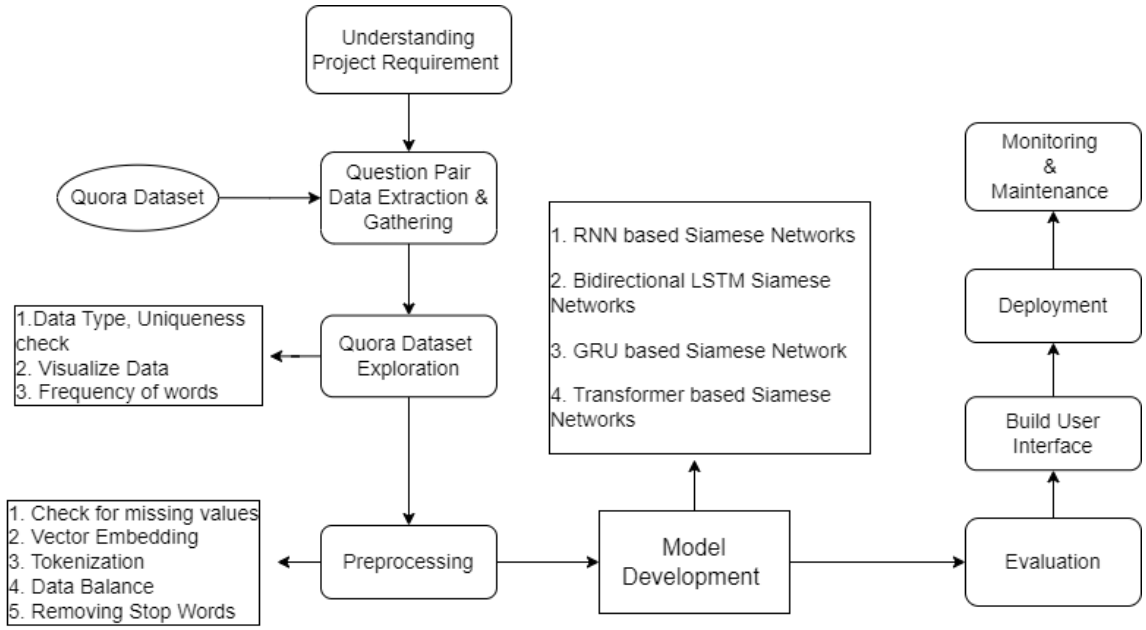
This project is on the front line in terms of improving digital content authenticity and availability; it addresses paraphrase detection challenges found within platforms like Quora. It is a pioneer in using state-of-the-art machine learning techniques including RNN and GRU-based Siamese Networks that offer a holistic approach to detecting and managing rephrased content. Through its meticulous data preparation, advanced modeling, and strategic deployments, the project emphasizes improving digital storage optimization and enhancing the quality of content. The innovative employment of transformer and bidirectional LSTM architectures by this project represents significant progress in deep learning that pushes boundaries for paraphrase identification accuracy. Additionally, this goes beyond technical achievements bringing about strong outcomes towards plagiarism detection thereby contributing to more organized and trustworthy digital environments. In this regard, it sets a new standard for content management that will guide research on future areas and applications focusing on data integrity and user experience in digital platforms.

The effects of the project are varied and far-reaching. In educational and academic settings, this can be a unique way to curb plagiarism among scholars and maintain scholarly standards. The model also works as an enhancer of content quality on digital platforms like Quora by sieving through duplicated queries and replies, hence making information exchange more fluid. On the other side, in legal and professional contexts it may facilitate checking the validity of texts included in documents involved in legal procedures. Furthermore, its advances in paraphrase identification technology could revolutionize content management systems, search engines, and customer service solutions by enabling them to reply more accurately to user queries which would be very relevant. Ultimately, therefore, this project helps create a far less

chaotic online information world that is more reliable for all stakeholders whether individuals or large organizations. Figure 1 Shows the Workflow Diagram of the project.

Figure 1

Workflow Diagram of Research Project



Project Requirements

Functional Requirements

In this phase, design a model which will identify the question pair semantic similarity using the Siamese network architecture. The models with RNN, Bidirectional LSTM, Transformer and GRU are suitable for finding the embeddings and semantic analysis of the questions pairs. The dataset is provided by Quora (Iyer et al., 2017) and is available publicly. The dataset contains thousands of question pairs labeled as paraphrased or not. It will be stored in Google Drive which will allow authorized team members to work in a collaborative environment.

For effective project management JIRA is utilized where Work Breakdown Structure (WBS) is developed creating hierarchical structure breaking down complex activities to manageable tasks making use of Epics, Tasks, and sub-tasks. Preprocessing and cleaning of the dataset will be done on local systems using Jupyter Notebook, which is a Python Integrated Development Environment (IDE). Data cleaning and preprocessing plays a crucial role in understanding data distribution and doing exploratory data analysis (EDA) to expose patterns or anomalies in the dataset that might affect model performance. Model training and evaluation will take place on a cloud-based platform like Google Cloud's Vertex AI that has scalable computing resources along with seamless integration with cloud storage. It is an environment that allows for dynamic allocation of computational resources to ensure efficient model training, evaluation, and iteration.

The metrics used to evaluate this model will be accuracy, precision, recall and F1 score based on rigorous evaluation criteria specific to natural language processing tasks. Furthermore, the project will investigate advanced metrics and methods suitable for semantic similarity assessment to accurately identify the paraphrases by the model. After successful development and validation, this paraphrasing identification system shall be prepared for implementation in real-life situations.

AI powered feature requirements

To develop a Siamese network model for paraphrase identification in Quora questions, several python libraries and AI techniques is employed at different stages throughout the project. Firstly, initial data pre-processing will take place in python using libraries such as NumPy and sci-kit learn. NLTK (natural language toolkit) and spaCy will be an advantage for tokenization and other NLP preprocessing tasks such as POS tagging, dependency parsing and word vectors.

TensorFlow library features will be used for building machine learning models including neural network architectures. GloVe or fasttext embeddings from language models will be beneficial for converting text into numerical vectors capturing semantic meanings. These embeddings enable the model to understand context and semantic similarity between words in the question. Contrastive loss or triplet loss functions will be utilized in training Siamese networks. These loss functions are designed to ensure that semantically similar questions are brought closer in the embedding space, while dissimilar ones are pushed apart. A blend of these techniques will mark a pivotal enhancement in the project efficiency and performance.

Data requirements

The project proposal proposed three data sets publicly available for solving the pairwise detection problem. In this project Quora question pair dataset is used as this is high in volume and it complies with privacy and ethical guidelines specified by Quora to use it for research or development purposes. The dataset is provided by Quora itself adhering to the authenticity of data. The dataset consists of pairs of questions with a binary label indicating whether the pair is semantically similar (paraphrased) or not. This is essential for training the Siamese network to distinguish between paraphrased and non-paraphrased question pairs. Each question pair will be carefully vetted to ensure high-quality annotations. The preprocessing is done by removing any duplicates, irrelevant content, or incomplete data entries.

The dataset will be split into training, validation, and test sets. Here there are two options, one is to consider the split ratio of 80% for training, 10% for validation, and 10% for testing to maintain a robust and accurate model. Second, is to use the train, test, validation split provided by the Glue Benchmark tasks (Wang et al., 2019). An imbalance in the dataset can lead the model to develop a bias towards the performance of the model, using different evaluation metrics

like F1 score, precision, recall, AUC-ROC, Rank Accuracy, and validations like K-fold validations to analyze the data and deal with data imbalance challenges.

Project Deliverables

The project lifecycle begins with understanding the business problem of interest which aims to solve followed by drafting a detailed Project Proposal describing the goal, objectives, and literature survey of existing solutions. After defining phases of the project through WBS, Data is collected from the source mentioned in Table 1 and is analyzed for the patterns and structure making the data ready for further processing.

Once equipped with the required data, estimation of efforts through Gantt Chart provides a roadmap ensuring optimal resource utilization through tasks and their outlines. Determining the critical path for the projects becomes crucial for its success, which is achieved through Critical Path Analysis (CPA) in Milestone-oriented PERT chart that outlines the critical tasks and time taken for them to complete.

The data ready to be modeled is then applied to the proposed models in parallel and evaluated through a series of validation processes. The model is then evaluated against the test data partition and is deployed. Inference script, Python notebook, intensive documentation spread across chapters and JIRA dashboard for project is prepared explaining the end-to-end project lifecycle, methodologies, and outcomes. The prototype inference script will be loaded with the best performing model and their optimum weights to demonstrate the identification of test question pairs. The production or development application will consist of a Streamlit (Python Library) user interface seamlessly integrated with the model's inference script. This interface will facilitate the evaluation of model predictions on test question pairs. All the documentation will adhere to APA style formatting guidelines facilitating standardization, credibility, organization,

and proficient reading to the audience. By employing a parallel approach that combines project management methodologies with adherence to technical task deadlines, ensures optimal and punctual outcomes as shown in Table 1.

Table 1

Table Outlining Project Deliverables, Description, and their Due Dates

Deliverable	Due Date	Description
Business Understanding	02/12/2024	The project aims to address paraphrase identification challenges in domains like data duplication, storage optimization and plagiarism detection while leveraging advanced ML techniques to achieve accurate and robust solutions.
Project Proposal	02/16/2024	Document outlining the purpose, scope, background, research, and SWOT analysis of the proposed research problem.
PM tool, methodology selection and WBS setup	03/02/2024	Work Breakdown Structure to clearly define deadlines, task flow and lifecycle, CRISP-DM methodology with chosen Project Management tool (JIRA) for efficient project planning and tracking.

Data Collection Plan	03/08/2024	The data is sourced from Quora's official data platform quoradata.quora.com and collected from one of their dataset releases (Question Pairs). Analysis of underlying structure of data and patterns.
Gantt Chart and Effort Estimation Plan	03/09/2024	Effort estimation for project components and team member roles assignment through Gantt chart providing a concise Roadmap.
Introduction	03/16/2024	An APA format document outlining the background information, context, significance, existing technology, and solutions in research area through literature survey to prepare the reader for the research topic.
Data Preparation Plan	03/18/2024	Performing Exploratory Data Analysis (EDA) through a series of data cleaning, tokenization and partitioning to prepare data for modeling.
PERT Chart	03/22/2024	Milestone-oriented PERT chart outlining Critical path post identifying project

		milestones, WBS, and CPA (Critical Path Analysis) for a high-level project overview.
Data Modelling Plan	03/28/2024	Implementation of proposed models, RNN, Bidirectional LSTM, Transformer and GRU based Siamese Networks using the data partitions from data preparation stage.
Model Evaluation	04/15/2024	Evaluating models against the test data partition to validate them against performance metrics. Retraining or tuning models to achieve better performance through validation process. Comparing outcomes with existing solutions known through literature.
Model Deployment	04/25/2024	Integration with deployment environment while addressing compatibility issues. Develop monitoring, logging, and inference script coupled with Streamlit User Interface.

Abstract Presentation	04/26/2024	Present Abstract of the project providing a summary of the research.
Project Presentation	05/10/2024	Present the project demonstrating the models used, results achieved and showcasing the best model with the help of a user-friendly user interface.
Individual Model development Research report in APA format	05/15/2024	Each team member documents in-depth details of one model each from the list of proposed models while following APA style formatting.
Group Project Report in APA format	05/17/2024	A comprehensive APA formatted document exhaustively covering every single concept through culmination of various chapters like Introduction, Data Management plan and Engineering, Model development serving as a solid guide for future work and advancements.

Technology and Solution Survey

In this section the existing technology for solving the problem of paraphrase identification using Quora Question Pairs dataset is reviewed. This is a classic Natural Language Processing task, and this dataset is included as part of the GLUE benchmark, which evaluates how adept a Natural Language model is across a collection of tasks (Wang et al., 2019). Quora is a question-and-answer site where questions are asked, answered, edited, and organized by its community of users (Abishek et al., 2019). The team behind Quora released a dataset in 2017 that contained more than 400,000 lines of potential question duplicate pairs. Each line contains IDs for each question in the pair, the full text for each question, and a binary value that indicates whether the line truly contains a duplicate pair (Iyer et al., 2017). A training, validation and test split for the same dataset is also available via the GLUE benchmark and has been the standard for comparison between different models on this task using this dataset.

Another dataset available to solve this task is the Semantic Textual Similarity (STS) benchmark dataset (Agirre et al., 2007). The dataset consists of 8628 sentence pairs, with 4299 pairs coming from news headlines, 3250 coming from image captions, and 1079 coming from user forums category. This dataset, just like QQP, is also part of the GLUE benchmark, and a train, validation and test split are also available from the same benchmark. The Microsoft Research Paraphrase Corpus (MRPC) dataset is also a similar dataset. It consists of 5800 sentence pairs extracted from online news sources (Dolan & Brockett, 2005). The pairs are identified automatically, but annotated manually by humans with labels to identify if the sentence pair is semantically equal or not.

Python provides a vast array of tools and libraries that enables machine learning development. The project is planned to use tools like Pandas (McKinney, 2010) for data

processing and loading the dataset to memory. Pandas supports a rich interface of functions for interacting with the data and provides tooling to get statistical insights on the data.

Scikit-Learn (sklearn) is a very popular library for training machine learning models (Pedregosa et al., 2011). It has a very mature and stable API that is accepted by many developers worldwide. Scikit-Learn has APIs available for clustering, dataset loaders, matrix decomposition, building ensemble models, feature selection, linear modeling, metrics, model selection, naïve bayes algorithms, nearest neighbor algorithms, basic neural network models, building pipelines, data preprocessing, SVM based modeling and decision tree-based modeling.

TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. It is flexible and can be used for training and inference algorithms, especially for deep neural network models. Models built using TensorFlow have been deployed in domains like speech recognition, computer vision, robotics, information retrieval, natural language processing, geographic information extraction, and computational drug discovery (Dean et al.).

Like TensorFlow, PyTorch is a framework designed from first principles to support Pythonic programming style, which supports code as a model, makes debugging easy, and is consistent with other popular scientific computing libraries. PyTorch has been very popular very popular in the research community with, for instance, 296 ICLR 2019 submissions mentioning PyTorch (Steiner et al., 2019).

Keras is a deep learning API written in Python and capable of running on top of either TensorFlow or PyTorch (About Keras 3, 2023). Keras is often used to develop modular components, that can be combined in a layered approach to build deep learning models. It has an

API that is like Scikit-Learn, and provides rich utilities for hyperparameter tuning, computer vision and natural language processing.

Wang et al. (2017) proposed a bilateral multi-perspective matching (BiMPM) model. Given 2 questions, P and Q, that are to be compared, this model first encodes the sentence P and Q using a bidirectional LSTM (Long Short-Term Memory) (Hochreiter & Schmidhuber, 1997) encoder. Then each time step of one sentence is matched against all time steps of the other sentence, in both directions i.e., from P to Q and Q to P. The matching results are aggregated using a bidirectional LSTM, before a decision is made using a fully connected layer. Using this method, they are able to get accuracy of 88.17% on the test split of QQP.

Li et al. (2018) proposed an attention boosted sequential inference model named aESIM, which adds word level attention mechanism to the traditional Bi-LSTM. The model consists of four main parts: encoding layer, local inference modeling layer, decoding layer, and classification layer. This model also introduced a new layer called Bi-aLSTM, which allows a normal LSTM layer to attend to different words in the sentence. aESIM is thus able to effectively learn representation of words and model inference between pairs of sentences. This model is able to achieve a test accuracy of 88.01% on Quora Question Pairs dataset.

Tan et al. (2018) proposed multiway attention networks, wherein four attention functions (concat, bilinear, dot, minus) are used to match the sentence pairs. A GRU (Gated Recurrent Unit) layer is used along with learned embeddings to generate the initial embeddings for each sentence. These embeddings are then passed through the proposed multiway-matching-aggregation framework to the final softmax output layer for class prediction. This model achieved a test accuracy of 89.12% on Quora Question Pairs dataset.

Choi et al. (2019) proposes a novel way of stacking LSTM layers, where each layer passes not only the hidden state, but also the memory cell states to the next layer. This ensures that features from lower layers are effectively conveyed to upper layers. A bidirectional version of this architecture achieved a test accuracy of 88.6% on Quora Question Pairs dataset.

Zhang et al. (2020) proposed Multi Fusion Asking Emphasis (MFAE) model. In this model, BERT is first used to generate word embeddings for each sentence. Then, bidirectional LSTM layers are used to generate inter and intra-asking emphasis by summing self and cross-attention between the sentences, respectively. Eight-way combinations are then used to generate multi-fusion asking emphasis and multi-fusion word embeddings. An ensemble of MFAE models is able to achieve a test accuracy of 90.54% on Quora Question Pairs dataset.

Godbole et al. (2018) investigates the use of Gated Recurrent Units (GRUs) in conjunction with other machine learning techniques like Random Forest, Adaboost, and Support Vector Machines (SVM). Utilizing GloVe word embeddings to capture semantic similarities between queries, the authors offer a model architecture that uses a bidirectional GRU within a Siamese neural network framework. Their tests show that the best performance can be obtained by combining GRUs with a Random Forest classifier, leading to competitive outcomes in a Kaggle tournament. This work demonstrates how well GRU-based models handle the intricacies of natural language and raises classification accuracy through the use of ensemble approaches, hence endorsing their application for paraphrase detection tasks.

He et al. (2021) proposed a Transformer based model called, which stands for Residual Attention Layer Transformer. They used skip-edges to connect the multi-head attention layers of the regular Transformer architecture, thus providing a direct path to propagate attention scores to the latter layers. This helped stabilize the training and provided significant gains on various

benchmarks. The RealFormer model got a test accuracy of 91.34% on Quora Question Pairs dataset.

Tay et al. (2022) proposed a Transformer based model called Charformer, which allows the standard Transformer model to learn subword level tokenization automatically during training by using Gradient Based Subword Tokenization (GBST) module. This is quite different from the standard practice where tokenization is either rule based (using regular expressions) or constructed in a data-driven approach as a preprocessing task before the model training using the same or separate dataset. Using this approach, the authors achieved a test accuracy of 91.4% on the Quora Question Pairs dataset.

Table 2

Comparison of Technical Survey

Authors	Technical Approaches	Models Used	Evaluation Metrics	Results	Conclusion
Wang et al. (2017)	Bidirectional LSTM, multi-perspective matching	BiMPM	Accuracy on test dataset	88.17%	BiMPM can detect duplicate questions very well
Li et al. (2018)	Bi-aLSTM, local inferencing and inference composition	aESIM	Accuracy on test dataset	88.01%	aESIM can also detect duplicate questions very well

Authors	Technical Approaches	Models Used	Evaluation Metrics	Results	Conclusion
Tan et al. (2018)	GRU, multiway-matching-aggregation framework	Multi-way Attention Networks (MwAN)	Accuracy on test dataset	89.12%	MwAN increased the benchmark for detecting duplicate questions
Choi et al. (2019)	Cell-aware stacking of LSTM	CAS-LSTM	Accuracy on test dataset	88.6%	CAS-LSTM is a capable model for wide variety of NLP tasks
Zhang et al. (2020)	LSTM Embeddings, BERT, multi-fusion asking emphasis	MFAE	Accuracy on test dataset	90.54%	LSTM embeddings boost performance, raising the bar for duplicate question detection
Godbole et al. (2018)	GRU, SVM	GRU	Accuracy on test dataset	89.21%	GRU is a capable model for wide variety of NLP tasks

Authors	Technical Approaches	Models Used	Evaluation Metrics	Results	Conclusion
He et al. (2021)	Skip-edges for improved flow of Attention scores	RealFormer	Accuracy on test dataset	91.34%	Realformer raised the benchmark for duplicate question detection, despite being trained for other tasks too
Tay et al. (2022)	Gradient Based Subword Tokenization (GBST)	Charformer	Accuracy on test dataset	91.4%	Charformer improves the benchmark for duplicate question detection

Literature Survey of Existing Research

Paraphrase Detection using Quora Question Pairs is a classic Natural Language Processing task, and hence models built for solving this task often use the same techniques – tokenization, embedding and attention-based layers like LSTMs and Transformers.

Tokenization is the process of breaking down sentences into smaller units called tokens, and it helps structure and organize textual data for further analysis using statistical means. Embedding is the process of representing tokens as numerical vectors in high-dimensional space. This allows the high-dimensional vector representations to capture the semantic and syntactic relationship between the tokens, thereby encoding the meaning of each sentence in the sequence of vectors associated with the sentence.

Pennington et al. (2014) introduced GloVe, an unsupervised learning algorithm for obtaining vector representation for words. Given a corpus of text, like Common Crawl, the GloVe model is trained using the frequency of word-word co-occurrence in the corpus. The model's main intuition is that ratios of word-word co-occurrence probabilities can encode each word's meaning. GloVe learns these embeddings by optimizing the objective that the dot product of the word vectors should be equal to the logarithm of the words' probability of co-occurrence. GloVe, when trained on a large corpus of text like Common Crawl, can learn word embeddings in such a manner that other words related to the same concept lie within its vicinity based on cosine similarity. For example, the nearest neighbors for the word embedding for frog are as follows – frogs, toad, litoria, leptodactylidae, rana, lizard, eleutherodactylus. GloVe has also shown that such algorithms can capture the nuances of difference between city and zip code, providing separate word vectors for the city and its zip code, even though objectively they both denote the same underlying concept.

Hochreiter and Schmidhuber (1997) introduced the Long Short-Term Memory (LSTM) architecture, which is a type of recurrent neural network (RNN). LSTMs incorporate memory cells with gating mechanisms that control the flow of information, granting LSTMs the capability to selectively store and discard information at each time step. This enables the model to effectively model sequences with long-range dependencies and resolves the vanishing gradient problem associated with RNNs. The capabilities of LSTM revolutionized the field of Natural Language Processing. It offered solutions to tasks that required understanding of sequential data, such as language translation and sentiment analysis. LSTMs excel in capturing intricate patterns and contextual information in textual data, enabling more accurate predictions, and generating coherent and contextually relevant outputs. A lot of models developed for solving Paraphrase

Identification using Quora Question Pair task have utilized LSTM and its variant in some form or other.

Vaswani et al. (2017) introduced the Transformer, which has since changed the landscape of Natural Language Processing. The core novelty of the Transformer lies in its use of self-attention mechanism, which allows the model to weigh the importance of input tokens independent of its position in the sequence. This architecture helps the model capture complex dependencies and relationships within the data more effectively compared to previous sequence modeling approaches. Unlike traditional models like Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) units, and Gated Recurrent Units (GRU) that processed data sequentially, the Transformer processes all input tokens in parallel, significantly improving efficiency and performance on NLP tasks. The attention mechanism of Transformer is very efficient at capturing long-range dependencies within the text, overcoming the vanishing gradient problem of previous models. The Transformer's self-attention mechanism eliminates the need for sequential data processing, allowing for much faster training times. Its ability to model and generate sequences has revolutionized tasks like machine translation, text summarization, and question-answering. Its efficiency, scalability, and superior performance have established it as a foundational architecture in NLP, driving forward the development of more advanced and capable models in the field. The introduction of Transformers helped cross the 90% test accuracy threshold on the Quora Question Pairs dataset.

Tung and Xu (2023) investigate several methods for entailment identification in the Quora Question Pairs dataset. In the study, attention-based models and more complex techniques like RNNs with GRUs and LSTM cells are contrasted with straightforward models like the Bag of Words. The models that incorporate word-by-word attention get the best accuracy, indicating

that attention processes considerably improve model performance. Because GRU-based models are effective at capturing complicated semantic links, the findings highlight the significance of combining temporal information and attention mechanisms and offer strong support for their use in paraphrase detection tasks.

Choi et al. (2019) proposed a unique architecture called Cell-aware Stacked LSTM (CAS-LSTM) to improve Natural Language Processing (NLP) phrase modeling capabilities. CAS-LSTM contains memory cell states from previous levels in addition to hidden states, which are only passed between layers in standard stacked LSTM designs. A soft gating system controls the movement of information both horizontally and vertically. Its design enables enhanced control of data and efficient feature transfer between tiers. CAS-LSTMs beat regular LSTMs in benchmark datasets for sentiment classification, paraphrasing recognition, natural language inference, and machine translation. To understand the underlying workings of the model, the researchers carried out a thorough qualitative examination. By proving how their strategy of using LSTM's gated processes to improve phrase representation is successful, their method has greatly enhanced the field. The work opens the way for developing new methods to improve stacked LSTM architectures that might transform a variety of natural language processing applications.

The study by Kumari et al. (2021) is about the problem of question repetition on platforms like Quora and represents a novel solution regarding this common challenge in large-scale knowledge sharing databases. In addition, they enhance Siamese-LSTM architecture with two new sections: firstly, the fusion of dense layer and secondly, mixing a machine learning classifier. The use of hand-engineered features greatly improves both models whereby Model -1 achieves an accuracy of 89.11% which demonstrates its good performance in duplicate question

recognition. While not only presenting the possibility of improving semantic analysis through augmentation with hand-crafted features; this piece of research establishes a new standard for identifying similarity between questions among other things. Their method not just reduces redundancy to make knowledge sharing platforms more efficient but also opens new avenues for future investigations into natural language processing particularly on optimizing question-answer systems.

Liu et al. (2021) introduce the Trans-Encoder model that revolutionizes unsupervised sentence-pair modeling by utilizing self-distillation and mutual distillation to learn without the need for labeled data. This is big because it addresses the difficult issue of absence of enough annotated datasets in tasks such as determining whether two sentences are alike? In contrast to other models, which require large amounts of data to be useful, the Trans-Encoder does not require such datasets thereby making it an invaluable tool when you can't easily or cheaply access labeled data. These writers show their model can compete with and surpass top scores in several benchmarks defined on sentence pairs by extracting knowledge from unlabeled sentences pairs. Unsupervised learning has always been something that has changed during natural language processing, but this study goes beyond that by creating ways that enable model sentence pairs without using annotated data. Unsupervised models like Trans-Encoder could potentially lead to major disruptions in paraphrase identification tasks, thus pointing towards an exciting direction for future work in NLP.

Yang et al. (2019) defined a technique to textual content matching of their studies, titled "Simple and Effective Text Matching with Richer Alignment Features", targeted at the improvement of a model which identifies alignment functions to enhance the accuracy of semantically similar text pairs. Their look presents a way that diverges from traditional

complicated deep gaining knowledge of architectures, aiming alternatively for simplicity and performance without sacrificing overall performance. The authors seize semantic relationships among texts via extracting and integrating numerous alignment signals. With the rigorous experiments on datasets, the authors diagnosed that their model now not handiest outperforms current textual content matching systems in accuracy but additionally operates with considerably decreased computational complexity. This study contributes to the sphere by imparting a robust and efficient solution for text matching. Thus, it doubtlessly aids programs including paraphrase identity, query answering, and records retrieval structures.

Graesser et al. (2019) delves into the demanding situations and improvements in natural language understanding (NLU) via their work with the Quora Question Pairs dataset. Their research underscores the dataset's capability as a tool for enhancing machine learning models' potential to figure semantic similarities between textual content pairs, an essential thing of NLU. By using advanced neural network architectures, including Long Short-Term Memory (LSTM) networks and Transformers, they sought to overcome the nuances of language that make paraphrase identification a complicated mission. Their evaluation found out that while conventional methods have struggled with the dataset's subtleties, the combination of context-aware embeddings and interest mechanisms significantly enhances models' overall performance. The paper contributes to the field by presenting a comprehensive assessment of different model architectures on the Quora dataset, highlighting the significance of contextual knowledge in NLU. Furthermore, exploration into the effects of dataset size and comparison on model accuracy offers precious insights for research, suggesting that larger, extra diverse datasets can be key to advancing NLU systems.

He et al. (2021) delivered a modern approach to enhancing the Transformer architecture, known as RealFormer, which incorporates residual attention mechanisms. Traditional Transformer models have revolutionized numerous fields of gadget studying by providing outstanding performance in obligations requiring knowledge of sequential facts, including natural language processing (NLP) and paraphrase identity. However, these fashions regularly come across demanding situations associated with attention distribution and training efficiency, specifically as version sizes and input sequences develop. RealFormer addresses those problems through integrating residual connections inside the attention mechanism itself, making it greater at powerful mastering and representation of dependencies throughout exclusive parts of the input sequence. This novel architecture now not only simplifies the gradient flow within the model, making it simpler to train, however, additionally enhances the model's capability to check nuances in information, thereby enhancing its universal overall performance on complicated obligations.

Hosking et al. (2021) introduce SEPARATOR, a new method that creates paraphrases to questions in English that maintain the same intent but change how they are expressed. This is an intelligent method that brings together a specific training goal with an intelligent design so that there exists a space of ideas where what is said can be separated from how it is said. In this case, they train their models to disassemble a question into two parts: rewriting it and putting it back together such that both the meaning remains intact as well as the appearance of the examples. By using Vector-Quantized Variational Autoencoder for representing the question, these authors employ a specific set of latent variables; thereby enabling them select different ways of expressing at test time without relying on external examples. Extensive testing and human evaluation indicate that SEPARATOR preserves meaning in its paraphrases while also providing

novel ways of expression better than other existing approaches. This milestone not only pushes ahead natural language processing by showing how to make intricate variations in paraphrases with minimal loss of original intent but also establishes a new standard for providing substantial linguistic modifications.

Tan et al. (2018), present the Multi-way Attention Networks (MwAN) which are designed to understand the complex relations between sentences in paraphrase identification, natural language inference and answer sentence selection. The researchers leverage on four different attention mechanisms to conduct a comprehensive analysis of pair of sentences at word-levels. MwAN integrates these diverse perspectives through a multi-faceted attention mechanism that efficiently captures the substance of such relationships. This paper examines the performance of this network across several datasets indicating its superiority over previous models in understanding and classifying sentence pair relations.

Gunyu L et al. (2018) in their paper titled "Attention Boosted Sequential Inference Model (aESIM)," introduce a revised version of the Enhanced Sequential Inference Model (ESIM) for natural language inference called attention boosted sequential inference model (aESIM). Word attention and adaptive direction-oriented attention mechanisms are added to the Bi-LSTM layer to improve word representation and capture local subsentential inference between premise and hypothesis sentences in aESIM. On evaluation with benchmark datasets such as Stanford Natural Language Inference (SNLI) corpus as well as MultiNLI corpus, ESIM models surpass other baseline models including aESIM in terms of performance hence indicating their better accuracy rate concerning NLI tasks. The visualization of attention indicates that the model can make semantic relationships visible between sentences. The importance of this research lies in its

exploration of advanced ways for enhancing understanding of sentence relationships which is crucial for accurate paraphrase identification in this project.

Ansari and Sharma's (2020) research looks into machine learning and deep learning approaches for detecting semantically duplicate queries on Quora. The study investigates a range of models, including deep learning models that incorporate GloVe embeddings, LSTMs, CNNs, and attention processes, as well as feature engineering and conventional machine learning classifiers like Random Forest and XGBoost. The best architecture of the authors' deep learning models reached an accuracy of 85.82%, which is in close agreement with Quora's state-of-the-art benchmarks. These models produced noteworthy results. This work shows how well deep learning techniques and GRU-based models capture semantic similarities, which is very helpful for the ongoing study on modeling GRU for paraphrase identification.

Wang et al. (2017) published work on Bilateral Multi-Perspective Matching (BiMPM) Model developed is a significant step forward as it overcomes flaws of previous works by introducing bidirectional matching as well as multi-perspective comparison. By combining word and character embeddings, BiLSTM encoders, and a new multi-perspective cosine matching function, this model checks contextual embeddings of two sentences from different angles. Ablation studies indicate that bilateral matching and different matching strategies are effective while experimental results on multiple datasets show that it is better than existing models for paraphrase identification. This comprehensive approach improves matching accuracy and gives insights into sentence relations vital for tasks such as natural language inference or answer sentence selection.

Shen et al. (2018) published an article aiming to provide an extensive comparative study of different types of SWEM (Simple embedding-based models) architectures in various natural

language processing tasks. The performance of SWEM variations in things like matching text sequence, sentiment analysis and short sentence processing are evaluated, and their efficiency has been compared with other models such as CNN or LSTM based models. This manuscript investigates the importance of word-order information which is very important for paraphrase identification project. It proposes a SWEM hierarchical pooling operation called SWEM-hier that captures local word order features which makes it competitive in sentiment analysis tasks. Consequently, the paper finds that considering word-order information might be significant when accurately identifying paraphrases helps the model capture fine-grained similarities and differences between sequences of words.

Table 3

Comparison of Literature Survey

Paper	Author	Year	Venue	Methodology
Long Short-Term Memory	Hochreiter and Schmidhuber	1997	Neural Computation, 9(8)	LSTM model
GloVe: Global Vectors for Word Representation	Pennington et al.	2014	2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)	GloVe embeddings

Paper	Author	Year	Venue	Methodology
Attention is All You Need	Vaswani et al.	2017	31st Conference on Neural Information Processing Systems (NIPS 2017)	Transformer
Bilateral Multi-Perspective Matching for Natural Language Sentences	Wang et al.	2017	Twenty-Sixth International Joint Conference on Artificial Intelligence	Bi-encoders, Cross encoders, Augmentation of training
Attention Boosted Sequential Inference Model	Li et al.	2018	arXiv (preprint)	Enhanced Sequential Inference Model (ESIM), Bi-LSTM.
Baseline Needs More Love: On Simple Word-Embedding-Based Models	Shen et al.	2018	56th Annual Meeting of the Association for Computational Linguistics	Bidirectional matching, multi-perspective comparison, Word and character

Paper	Author	Year	Venue	Methodology
and Associated Pooling Mechanisms				embeddings, BiLSTM encoders, multi-perspective cosine matching function
Multiway Attention Networks for Modeling Sentence Pairs	Tan et al.	2018	Twenty-Seventh International Joint Conference on Artificial Intelligence	Multiway Attention Networks (MwAN) for modeling sentence pairs,
Determining Entailment of Questions in Quora Dataset	Tung et al.	2023	Standford achieve	Bag of words, RNN, GRU, LSTM cells and attention-based models.
Cell-aware Stacked LSTMs for Modeling Sentences	Choi et al.	2019	Machine Learning Research 101. ACML 2019	Stacking Long Short-Term Memory (LSTM) layers called Cell-aware Stacked

Paper	Author	Year	Venue	Methodology
				LSTM (CAS-LSTM)
Natural Language Understanding with the Quora Question Pairs Dataset	Graesser et al.	2019	ICLR 2020	Continuous Bag of Words (CBOW) model
Simple and Effective Text Matching with Richer Alignment Features	Yang et al.	2019	Association for Computational Linguistics	Neural approach for general purpose text matching applications
Identifying Semantically Duplicate Questions Using Data Science Approach: A	Ansari et al.	2020	Arxiv Achieve	Optimization of GRU model architecture

Paper	Author	Year	Venue	Methodology
Quora Case Study				
Detecting Duplicate Question Pairs Using GloVe Embeddings and Similarity Measures	Kumari et al.	2021	Springer Link	Augmenting a Siamese architecture with Long Short-Term Memory (LSTM)
Factorising Meaning and Form for Intent-Preserving Paraphrasing	Hosking et al.	2021	59th Annual Meeting of the Association for Computational Linguistics	SEPARATOR, a paraphrase generation approach using a Transformer-based encoder-decoder model with a Vector-Quantized Variational Autoencoder

Paper	Author	Year	Venue	Methodology
Trans-Encoder: Unsupervised Sentence-Pair Modeling Through Self- and Mutual-Distillations	Liu et al.	2021	ICLR 2022	Trans-Encoder model
RealFormer: Transformer Likes Residual Attention	He et al.	2021	Association for Computational Linguistics	RealFormer for connecting the multi-head attention layers of the regular Transformer architecture

Data and Project Management Plan

Data Management Plan

Data Collection Approaches

This project has a powerful and relevant text dataset for research on paraphrase identification that is publicly available and widely recognized in Natural Language Processing (NLP). The key dataset used for this study is Quora Question Pairs, which has many question pair data examples marked for paraphrase identification.

The Quora Question Pairs dataset is essential for training models that can distinguish semantic equivalence between pairs of questions. It is significant due to its structured nature and the depth of linguistic features embedded in its questions. It has over 400,000 labeled examples, selected carefully to reflect a good mix of paraphrased and non-paraphrased cases.

Quora.com is an information-sharing website with a collection of questions and answers, where the question pairs dataset is obtained. The Quora Question Pairs dataset replicates real-life situations, reflects real life situations in addition to using language taken from them, thereby catching the intricacies and intricacies involved in paraphrase identification.

For this project, the data collection approach seeks to take advantage of this dataset's potential by carrying out a series of preprocessing steps that prepare text for deep learning models. These steps incorporate stemming, lemmatization, tokenization, and embedding to ensure that the model is accurate and smooth and does not include irrelevant information that could impact its accuracy.

Like all industry best practices, the data collection approach also involves an elaborate examination of the dataset distribution. Balance between paraphrase and non-paraphrase classes has to be maintained so that there will not be any bias towards the more occurring class by the model. Precision and recall in paraphrase identification tasks are contingent upon maintaining this equilibrium.

This research project relies on effort to choose datasets carefully, following stringent guidelines for their preparation with utmost seriousness. In this study, use a comprehensive and well-annotated Quora Question Pairs data set; thus, the study endeavors to be at least novel in paraphrase identification research with possible significant implications in NLP applications.

Table 4

Dataset and its Attribute description

Attribute	Details/Description
Dataset Name	Quora Question Pairs
Dataset Source	Data @ Quora
Categories	Diverse topics ranging from technology, health, science, to education and more
Classification Type	Binary (Paraphrase / Not Paraphrase)
Number of Question Pairs	Over 400,000 pairs
Number of Classes	2 (Paraphrase, Not Paraphrase)
File Format	.tsv
Data Attributes	Question ID 1, Question ID 2, Question Text 1, Question Text 2, Is Paraphrase?

Attribute	Details/Description
Preprocessing Steps	Removal of duplicates, incomplete pairs, cleansing of textual anomalies
Dataset Link	https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

Data Storage and Management Methods

This research project can only be successful if it is stored and accessed securely. The raw dataset, along with the final cleaned data will be stored on a Google Shared Drive with strict access control. This ensures secure shared access to data, with built-in version control and folder management.

Data Volume and Organization

The QQP dataset, segmented into 'test', 'train', and 'validation' sets, is strategically stored within a Google Shared Drive folder named "QQP". These splits have been obtained from the GLUE benchmark. The organized data is as follows:

Training Set (train.tsv). At the heart of machine learning endeavor lies the training set, weighing in at 49.9 MB. It serves as the primary dataset from which models learn and derive patterns, crucial for the prediction accuracy of question pair duplicates.

Testing Set (test.tsv). The testing set, at 48.4 MB, plays a critical role in objectively assessing the performance and generalization of models beyond the data they are trained on.

Validation Set (validation.tsv). The validation set, carefully curated at 5.55 MB, is integral to fine-tuning models. It acts as a buffer between training and testing, ensuring models performance is robust and reliable.

The shared drive backs up the data set, preserving it from accidental loss. Such information is only accessible to authorized individuals who control access to it to protect it and ensure the integrity of the research conducted. As shown in the below Figure 2 and Figure 3.

Figure 2

Folder Structure of the Data set

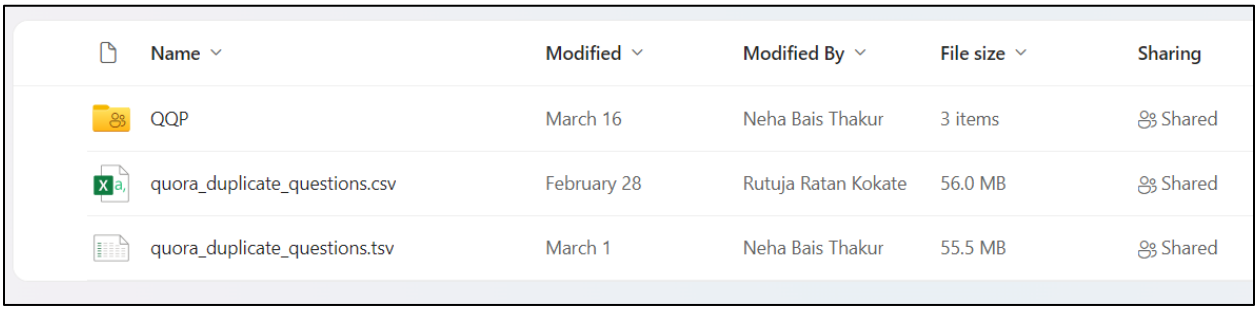
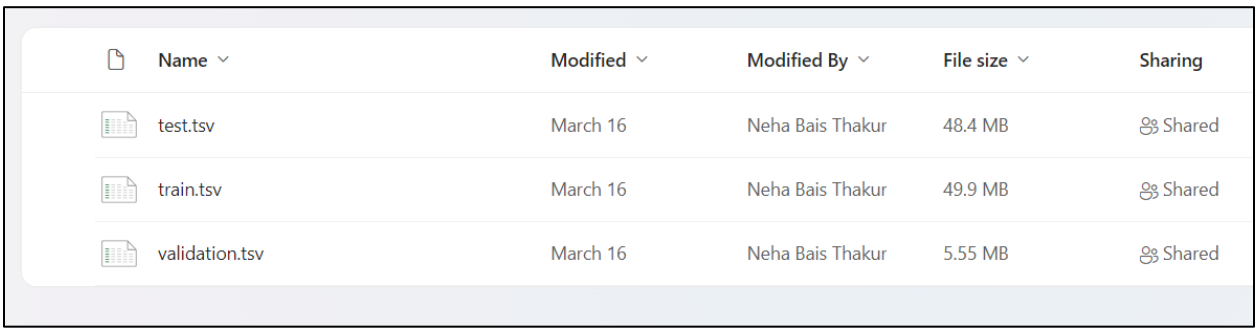


Figure 3

Folder structure after Dataset Split



Data Usage Mechanism

To ensure the data usage mechanism for the Quora Question pairs (QQP) dataset names and handles data in a systematic way that maintains clarity and consistency across the methods using processing and validation, the system addresses each of these steps thoroughly. There is a

structured naming convention for each data file in the QQP folder which shows its role within the architecture of the dataset.

Test.tsv. The testing subset, pivotal for evaluating model's performance on unseen data.

Train.tsv. The foundational dataset on which models will train and learn to discern patterns.

Validation.tsv. A crucial dataset for fine-tuning model parameters before the final evaluation.

QQP Dataset Filename Structure The naming convention provides immediate insight into the data's purpose and content.

[id]. A unique identifier for each question pair, ensuring traceability.

[qid1] & [qid2]. Question identifiers that uniquely mark each question within a pair

[question1] & [question2]. The actual textual content of the paired questions.

[is_duplicate]. Binary indicator denoting whether the question pair is a duplicate (1) or not (0).

Data Preprocessing and Handling. Before using the QQP dataset, it is necessary to take some preliminary steps to ensure that the data is of excellent quality. These entail turning questions into tokens, normalizing text, removing stop words, or using stemmers and lemmatizes. The required tools for this include Python libraries such as NLTK or spaCy.

Validation and Cross-Validation. Models will be validated using cross-validation techniques. While the dataset does not inherently come with a predefined cross-validation structure, implement a k-fold cross-validation mechanism, using a 70-10-20 split for training, validation, and testing sets respectively, unless the project requirements dictate otherwise. This approach will mitigate overfitting and ensure that the model's performance is generalizable.

Distribution of Responsibility among Team Members. The DMP stage of this project is divided into 4 broad phases carried out by different group members represented in Table 2 below.

Table 5

Distribution of Responsibility among Team Members

DMP Phase	Responsible POC	Work Items	Student ID
Data Collection	Neha Thakur	Ensuring dataset applicability to project goals; managing dataset download protocols; understanding and documenting metadata; ensuring legal compliance.	017442906
Data Processing	Rutuja Kokate	Processing and preparing data for analysis; performing quality control checks; splitting data into training, validation, and test sets.	017453865
Data Storage	Saumya Varshney	Evaluating and implementing secure storage solutions; managing data on Google Shared Drives; planning data recovery and backup protocols.	017417283

DMP Phase	Responsible POC	Work Items	Student ID
Data Utilization	Vinay Bhati	Establishing data pipelines for modeling; managing structured dataset use for reproducible results; version control.	016853239

Project Development Methodology

A project management methodology is a set of guiding principles a team uses to plan, execute, and manage a successful project. Some of the standard project management technologies are Agile and waterfall techniques. The 3C's Collaboration, Communication, and Coordination are key elements of agile methodology that help to ensure the success of any Agile project. On the other hand, the waterfall technique depends on three principles: low degree of customer involvement, robust project documentation, and sequential structure of projects. The input of one phase is dependent on the previous phase.

According to Schröer et al., (2021), CRISP-DM (Cross-Industry Standard Process for Data Mining) is a standard and an industry-independent process model for managing data mining projects. CRISP-DM methodology used for this project involves six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. Below is the description of each phase of the project.

Business Understanding

The first phase of CRISP-DM methodology is business understanding. As the name suggests, at this stage, business problems are understood. With the business Understanding, business objectives are formulated. The business objective should be simple, crisp, and easy to

understand. Next, assess the situation by checking what needs to be done in terms of inventory of resources, the requirements, assumptions, and constraints of the project, the risks and contingencies, terminologies used and to be used, and then the costs and benefits of the project.

Further, determine the goals and objectives for the project in technical terms and define the success criteria. After thorough research on the business understanding and objectives, a project plan is produced describing the specific project stages, the duration, resources needed, and the costs, and an initial assessment of the tools and techniques to be used.

Data Understanding

The second phase of CRISP-DM methodology is Data understanding. This phase involves Collecting the Initial Data or acquiring the data and its access to the data listed in the project's resources. Then, describe the Data by examining the properties of the data acquired and provide a description report regarding the dataset's format, quantity, and metadata. The following steps are doing exploratory data analysis to understand the patterns by querying, visualizing, and generating summary report. In this stage, find the first or initial hypothesis and its impact on the project.

Data Preparation

After the data understanding phase, the data is prepared for a model to generate outputs. Typically, in this phase, cleaning the data, integrating the data, conduct data encoding, feature engineering, and selecting variables and features is done. One of the criteria for determining the data is that it should be relevant to the data science goal identified in the Business Understanding phase. Then, after cleaning the data, construct it by including derived attributes, entire new records, or transformed values for existing attributes. Next, the data is integrated by combining data from multiple datasets to create new datasets or values.

Modeling

In the modeling phase, select the modeling technique to be used. Identified in the Business Understanding phase. Next, the test design is generated by creating a procedure to test the model's quality and validity. Then, describe the intended plan for training and testing and how to evaluate the models. The model is built by running it using the prepared dataset. Once the model is tested, list the parameters, and choice of parameter setting.

Further, the model is assessed by interpreting the models according to domain knowledge and criteria. At this point, score the quality of the produced models and summarize their output. Examine the confusion matrix, graphs, and model performance metrics.

Evaluation

The fifth phase of the CRISP-DM methodology is evaluation. In this phase, test the models on test applications and determine how well the model satisfies the business's objective in this phase of outcomes evaluation. Further, review the process by conducting a more detailed review to identify if there is any essential factor or task that has somehow been overlooked during the process, identify any quality assurance issues, summarize the process review, and highlight activities that have been missed and should be repeated. Then, the following steps are determined by assessing how to proceed with the project. In this part, include a list of possible next steps, reasons for and against each one, and instructions on how to proceed.

Deployment

The final phase of CRISP-DM methodology involves determining the resources needed to deploy the solutions, identifying the stakeholders affected by the deployment, and setting a timeline. Once the planning is complete, a plan is created for deployment. This plan outlines the steps involved in deploying the solutions, the resources required, and the timeline for

deployment. The model is tested on a test environment before finally pushing to the production environment, and testing the solutions to ensure that they work as expected.

Once the model is tested, it is deployed to the production environment. This includes installing the solutions, configuring the required infrastructure, and integrating them with the current systems. The final Deployment phase step is monitoring and maintaining the deployed solutions. This involves ensuring the solutions function correctly, fixing issues, and making necessary updates.

Project Organization Plan

The project management plan serves as a navigational tool that steers the team through the project's implementation, oversight, and management phases. It outlines the methodology for project development and specifies the project management tools. For this project, CRISP-DM methodology for project organization and management is used. The work breakdown is done using the WBS (work breakdown structure) diagram, which helps manage the project by breaking down the project into six phases as per the CRISP-DM methodology.

Work Breakdown Structure

The Work Breakdown Structure (WBS) provides an overview by breaking the project into specific stages. For this project, the WBS encompasses several key phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation and Optimization, and Deployment.

The first level of the WBS contains six phases as per CRISP-DM methodology:

Business Understanding

In this phase, understand, identify, and discuss project objectives and requirements, determining the project scope and functionality. Also, define the problem domain, how the

proposed solution works, and what the expected outcome can be. This phase guides the entire project and its design.

Data Understanding

In this phase, collect data from the Quora Question Pair dataset and explore the data characteristics. Perform EDA by analyzing and visualizing the data to find patterns. Check and understand word corpus.

Data Preparation

In this phase, clean and construct dataset to get it ready to be modeled.

Clean data. Check and remove missing rows, extra spaces, extra new line characters, punctuation, and unnecessary capitalization.

Construct Data. Each sentence in the dataset is modeled as a sequence of words. Convert each word to a number by assigning it a token. Then learn an N-dimensional embedding associated with each token by either learning it from dataset or using pre-learned embeddings from other models like GRU, FastText, and GloVe. Then split the data into training, testing, and validation sets.

Modeling

In this phase, 4 different types of models are built using Siamese network architecture. Each model will also be trained using trainable and fixed embeddings. All the models and their variations will be evaluated on various metrics using a validation set. Tune hyper-parameters of the best models. Accuracy, precision, f1 score, and recall are calculated and reported for each model as part of the training phase.

Evaluation

In this phase, evaluate the models and access them using metrics like accuracy, precision, recall, and f1 score, which are calculated and reported for each model. Identify the performance using the test data set and compare the results with the existing solutions from the reference papers.

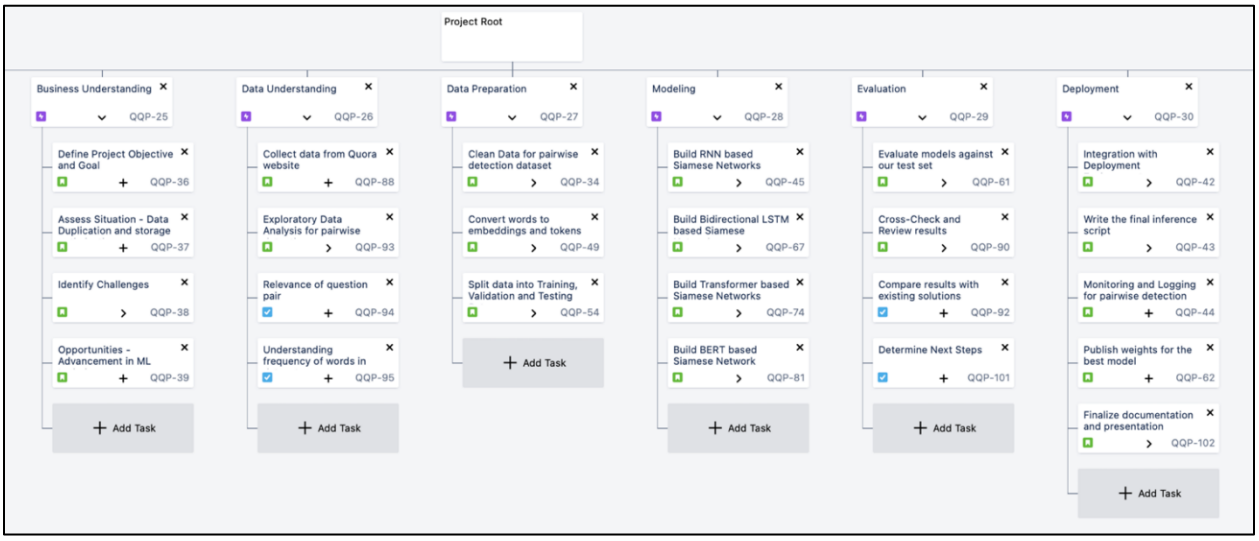
Deployment

Here, integrate and deploy the model into the target environment. Also monitor and log its performance. Then, consolidate data from each phase for the final report and presentation. The inference script will be loaded for the best-performing model and its optimal weights to demonstrate the identification of test question pairs.

Below is Figure 4, which represents the complete Work Breakdown Structure of the entire project.

Figure 4

Work Breakdown Structure of the Entire Project



The above work breakdown structure is created in JIRA. These phases are defined as Epics in JIRA. Each phase is further broken down into manageable stories, tasks, and subtasks

based on the project's requirements. It also serves as the base for creating GANTT and PERT charts.

Project Resource Requirement and Plan

This section outlines the hardware, software, tools, and licenses required for paraphrase identification using the Quora question pairs project. The project will employ Python programming language combined with various ML and deep learning libraries. Data preprocessing will use Python packages from NumPy, sci-kit learn, tokenization, and NLP processing, which will be done using NLTK (Natural Language Toolkit) and spaCy, considering their fast-processing capabilities. TensorFlow library features will be especially advantageous when building machine learning and deep learning models because of their automatic gradient computation feature. Harness GloVe or Embeddings from Language Models like GRU for text Vectorization, capturing the semantics. GloVe is trained on over 4.3 million Wikipedia articles. To compute the loss functions, utilize the contrastive loss or triplet loss functions in Siamese training networks, measuring the similarity of two inputs that have distinctions, thereby optimizing the network to better distinguish between similar and dissimilar pairs.

Python 3.7 is the most suitable version, encompassing all necessary features while maintaining compatibility across these Libraries. Python Integrated Development Environment (IDE) like Jupyter Notebook leverages Data preprocessing and EDA while Google Colab implements model training and evaluation. The original dataset of 55.5MB and its backup will be stored on Google Drive and Local storage. It will be accessible exclusively to the team members exhibiting storage infrastructure for the project while providing data reliability and fault tolerance regardless of geographic location. Apart from the technical requirements, project planning tools such as JIRA will be used to track issues, dependencies, deadlines, overall

progress, resource allocation, and utilization. Team Members conduct scheduled meetings, providing updates and discussing and addressing issues through Zoom meeting sessions. Collaborative documentation through Microsoft 365 ensures real-time cooperation among team members, enhancing productivity and facilitating efficient project recordkeeping.

These libraries, tools, and infrastructure comprehensively fulfill the paraphrasing identification using Quora question pairs.

Software Requirements

Table 6 illustrates the recommended software or library versions for successfully implementing paraphrase identification of Quora question pairs. Python's NumPy library (Version 1.26) is used for mathematical computations in the project. These include operations on matrices and arrays, among others, useful in activities like data manipulations and computation of similarity measures.

Scikit-learn (Version 1.4) builds machine learning and statistical models. This library includes many machine-learning algorithms and tools for classification, regression, clustering, and dimensionality reduction.

Besides this, Python NLTK (Natural Language Toolkit) (Version 3.8.1) is necessary for text mining/analysis, which involves tokenizing texts into sentences or words and semantic reasoning capability. The toolkit supports numerous Natural Language Processing (NLP) tasks like text preprocessing such as stemming and lemmatization, part of speech tagging, and named entity recognition feature extraction, among others. Stemming is a process that truncates a word by removing the suffix, leading to sometimes wrong meaning and spelling errors. Lemmatization considers the context in which it is used and changes a word into its meaningful base form, called Lemma. SpaCy (version 3.7.4) is similarly compatible with NLTK, as it has advanced

natural language processing capabilities. This allows for efficient tokenization, part-of-speech tagging, and dependency parsing, contributing to semantic analysis and understanding of text data.

TensorFlow (Version 2.16.1) is used to build machine learning and deep learning frameworks like Siamese Based network in the project. Its powerful computational graph abstraction and automatic differentiation abilities enable efficient model training and deployment for tasks like paraphrase identification.

These modules are used with Google Colab Notebook, a cloud-based Python Integrated development environment (IDE) that provides a collaborative and interactive computing environment that enhances data exploration, visualization, and code experimentation, which makes it suitable for data preprocessing and exploratory data analysis (EDA). Also using Google collab for model training and evaluation. It will be employed to design machine learning workflows, including data preparation, model development, hyperparameter tuning, and model deployment.

Table 6

Required Software's

Software/Libraries	**Version**	**Purpose**
Python - NumPy	1.26	Mathematical operations
Python - scikit learn	1.4	Machine learning models and statistical modelling

Software/Libraries	Version	Purpose
Python- NLTK (Natural Language Toolkit)	3.8.1	Analysis of text data, Tokenization, and sematic reasoning.
Python- spaCy	3.7.4	Supports word vectors on top of NLTK
Python- TensorFlow	2.16.1	Supports Machine learning and Deep learning frameworks
Google Colab Notebook	-	Cloud based Python IDE for Model training and evaluation

Every software or library version in Table 3 is meticulously selected to ensure compatibility and best performance within the project's environment, which ranges from data processing techniques, modeling approaches, and evaluation measures for paraphrase identification through Quora question pairs.

Hardware/Software Requirements

The section on hardware requirements details the basic equipment settings of various models for tasks such as documentation management, data preprocessing and storage, model training and evaluation, and dataset storage.

Table 7

Hardware/ Software requirements, Configuration, and Goals.

Objective	Hardware/Software	System Configuration	Goal
Data Preprocessing and Backup	Local Machine	8-core CPU and GPU	Data preprocessing on local machine leveraging the machine's computational resources and maintaining control over data privacy and accessibility and backup storage.
Dataset Storage	Google Shared Drive	55MB Cloud Storage	Centralized and accessible repository for storing project dataset, facilitating seamless collaboration while providing secure storage infrastructure.

Objective	Hardware/Software	System Configuration	Goal
Model training and evaluation	Google Colab	Python 3 Compute Resource, RAM 1.03/12.67 GB Disk 24.41/107.72	Cloud based platform for streamlined model development, leveraging cloud-based resources for efficient training, hyperparameter tuning, and evaluation.
Documentation	Google Shared Drive	~50MB Cloud Storage	Centralized and accessible platform for collaborative project document management.

Table 7 provides a system configuration of hardware and software that will be used during various project processes. Preprocessing data requires taping into much computation power from the local machine (8-core CPU and GPU, 16 GB RAM, Storage 512 GB) for purposes related to maintaining integrity in cases of data privacy, particularly while dealing with

this kind of processing. Storing datasets on Google Shared Drive is helpful as it has enough capability to handle large volumes of information for collaborative task force. The GloVe Vectorizations file of size 1.04GB will also be stored on Google shared Drive for centralized access. Using Google Colab for model training and evaluation allows leveraging cloud-based resources that provide Python 3 compute resources with 1.03/12.67 GB RAM and 24.41/107.72 GB disk space, thus facilitating streamlined development and hyperparameter tuning (Google Colab, 2020). Furthermore, documentation management on Google Shared Drive provides a central platform where each member can access project documentation for collaboration among peers. These specifications are relevant for all types of project activities aimed at achieving optimum performance and efficiency.

Licenses and Tools

Incorporating different tools into a project always comes with costs, but these expenses are known as investments, which can bring magnificent benefits for realizing project goals. The costs assigned to each tool contribute to how it helps the project succeed.

Table 8

Tools and their objectives.

Tool	Cost/Subscription	Objective
JIRA	Free	Project progress tracking and management
	Free	Hierarchical organization of project tasks to integrate their scope and deliverables.

Tool	Cost/Subscription	Objective
WBS Move and organize		
Gantt Chart	Free	Provides macro and perspective into all tasks tracking their completion rates while catering to the business requirement.
Draw.io	Free	To design PERT chart and other diagrams required to be included in the project documentation.
Zoom	Student Subscription	To Facilitate Team meetings allowing effective communication, cooperation, and decision-making.
MS Office	Student Subscription	Productivity tools for referencing, recordkeeping and consolidating all details at once place.

This project uses JIRA's free tier subscription, ensuring that resources are allocated effectively, workflows streamlined, and projects coordinated well, leading to timely delivery of project milestones. The Move and Organize tool for WBS is free and will be used as a Project

Management tool. Through its hierarchical organization functionalities, this device can help align tasks within a project with goals or deliverables, thus enhancing the process of planning and executing projects more effectively. By investing time in using this tool appropriately, the team working on the project can have much better control over their scope of work and clearer task interdependencies, thus contributing to the overall venture's successful completion.

Although it is free, the Gantt Chart tool provides significant insights into project schedules and milestones. Investing time in creating and updating Gantt charts gives the whole project team a view of what is happening at a macro level, which helps them allocate resources better, prioritize tasks more effectively, and manage risks appropriately. Investing in visualizing project schedules improves project planning, monitoring, and decision-making, resulting in timely project delivery and stakeholder satisfaction.

The draw.io software has been licensed for use by different project teams so that they can avoid extra costs when developing PERT charts, among other diagrams. It helps design clear and comprehensive visual representations of project workflows and processes. This fosters better decision-making while reducing misunderstandings during collaborations that lead to project success.

However, Zoom has numerous advantages in promoting effective communication and collaboration, a student subscription is utilized to use its features Specifically, through Zoom meetings, the team ensures frequent productive interactions, thus a faster issue resolution process, enhanced alignment regarding the project goals, and improved teamwork spirit.

Similarly, a student subscription is used for MS Office, which provides essential productivity tools for referencing, record-keeping, and consolidating project details. When used appropriately to develop project documents, it illuminates, organizes, and makes document

retrieval possible for the project team. In this case, documentation management promotes transparency, accountability, and knowledge sharing, leading to efficient and practical projects. The investments mentioned in Table 8 contribute to delivering successful projects and achieving their goals.

Costs, Specifications and Justifications

Tasks like data pretreatment, model building, evaluation, and deployment require a local machine. The estimated cost includes the cost of buying a computer system that meets the requirements for these activities and has enough RAM and processing capability. The GloVe Vectorizations are available for free to download. Team members can collaborate and access project data more efficiently when Google Shared Drive is a central repository. The free cost ensures accessibility without any additional expenses.

Python code is executed on Google Colab's cloud-based platform, making it appropriate for tasks like data pretreatment, model creation, evaluation, and deployment. The free price makes seamless collaboration and cost-free access to resources possible.

Project management with JIRA software includes progress tracking, task tracking, and problem management. The free cost ensures effective project management and coordination, which relieves further financial strain. Project meetings are conducted via Zoom, which helps team members communicate and work together in real-time. The free cost facilitates efficient decision-making and communication without needing to be budgeted for.

Table 9

Cost Estimation

Function	Resource Type	Resource	Duration	Estimated Cost
Local Machine	Hardware	Minimum 64-bit	3 Months	$1,315
Data Storage	Software	Google Shared Drive	3 Months	Free
Data Preprocessing	Software	Google Collab	3 Months	Free
Model Development	Software	Google Collab	3 Months	Free
Model Evaluation and Validation	Software	Google Collab	3 Months	Free
Model Deployment	Software	Google Collab	3 Months	Free
Project Management	Tool	JIRA	3 Months	Free
Project Meetings	Tool	Zoom	3 Months	Free
Total Cost				$1,315

The breakdown of overall cost $1,315 mentioned in Table 9 accounts for both the anticipated hardware expenditures and any potential project-related extras. Using free tools and resources ensures effective project execution while reducing total costs.

Project Schedule

Gantt Chart

The Gantt chart is like a calendar for the project. It is used to manage projects to see what tasks need to be done, when to start and finish each, and how tasks depend on each other. It is a visual to-do list with a timeline. On this chart, each task is shown as a bar; the length of the bar shows how long the task will take. If some tasks need to be done after others, dependency arrows link them to show the order.

For effective project planning and tracking, the WBS is utilized in addition to Gantt Charts. These charts present a comprehensive schedule that includes tasks, timelines, responsible team members, and the real-time status of deliverables. Gantt charts will help us track and plan the project and its dependencies to see a bigger picture of project progress.

This project is being worked on in one week sprint with weekends being non-working, just like a regular work schedule. The team has chosen to work through the spring break and keep working, ensuring tasks are given out all the time, with continuous task allocation throughout the project duration, unless there is a need to wait for the dependent task to complete. Tasks are assigned for three days maximum, and sub-tasks take 1-2 days to complete based on their complexity. Each part of the project is carefully planned, showing exactly what needs to be done.

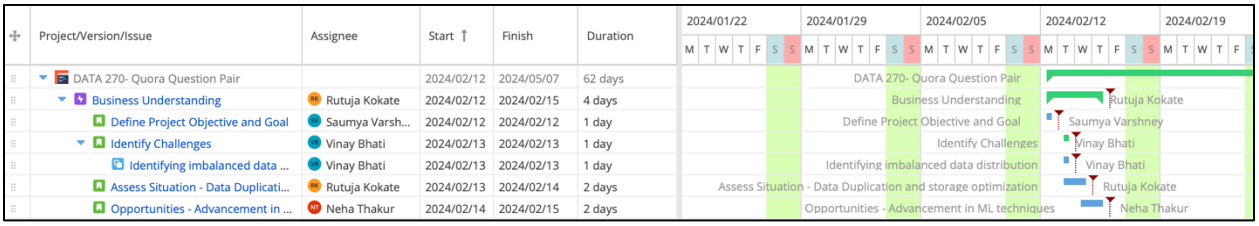
Business Understanding

Business understanding is an epic; here the key objective is to understand the project scope and objectives in this phase. The next-level division of tasks in this epic is called a story. There are multiple stories, such as defining project objectives and identifying challenges and opportunities. These stories have tasks associated with them. The business understanding phase started on February 12th, 2024, and ended on February 15th, 2024. This phase lasted four days.

Below is the Gantt chart Figure 5, which represents the tasks, timelines, and progress of business understanding.

Figure 5

Gantt Chart – Business Understanding Phase



Data Understanding

In this phase, collect the data and perform exploratory data analytics on the data collected. The objective of the EDA is to understand any underlying outliers and unnecessary trends in the data. This phase started on March 8th, 2024, and ended on March 15th, 2024, spanning six days.

Below is the Gantt chart Figure 6, which represents the tasks, timelines, and progress of data understanding.

Figure 6

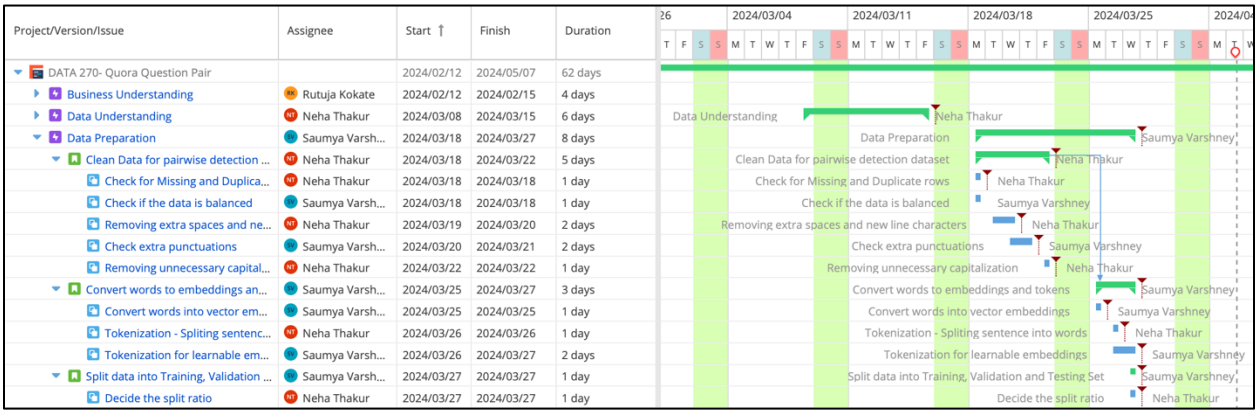
Gantt Chart – Data Understanding Phase

Data Preparation

The main objective of this phase is preprocessing and cleaning the data set to ensure the data is ready for the next phases. Once the data is ready, it is split into training, testing, and validation. This phase starts on March 18th, 2024, and ends on March 27th, 2024, spanning eight days. The Gantt chart below, Figure 7, represents the tasks, timelines, and progress of the Data preparation phase.

Figure 7

Gantt Chart – Data Preparation Phase



Modeling

In the Modeling epic, implement the proposed algorithms like LSTM, transformer, GRU, and RNN, and train the models. There are multiple stories and tasks, such as designing the model,

implementing the baseline DL models, training them with embeddings, and evaluating the model using the standard metrics on training data. The start date is March 28th, 2024, and the end date is April 11th, 2024, spanning 11 days. The flow of the modeling phase is shown in Figure 8**.**

Figure 8

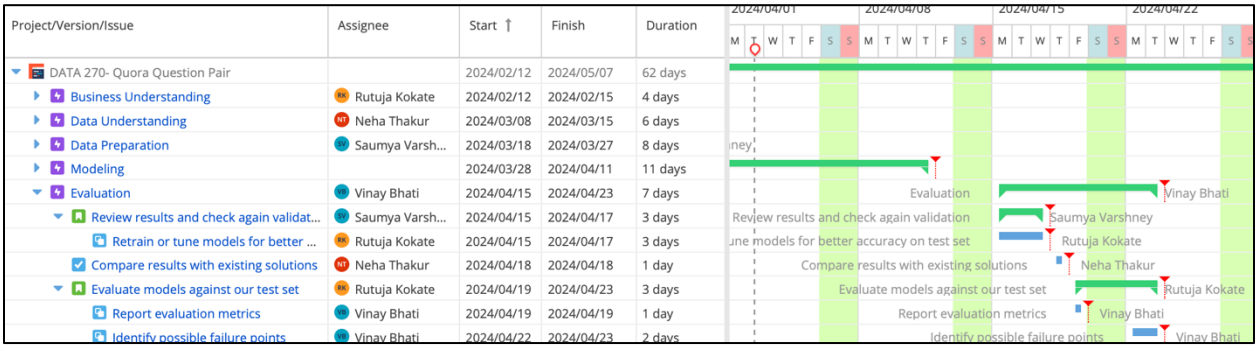
Gantt Chart – Modeling Phase

Evaluation

In the Evaluation phase, take the best-performing model from all four models trained and compare its performance with existing approaches. Then identify areas for improvement and make enhancements. This phase, which spans seven days, starts on April 15th, 2024, and ends on April 23rd, 2024. The Gantt chart Figure 9 below represents the tasks, timelines, and progress of the Evaluation phase.

Figure 9

Gantt Chart – Evaluation Phase

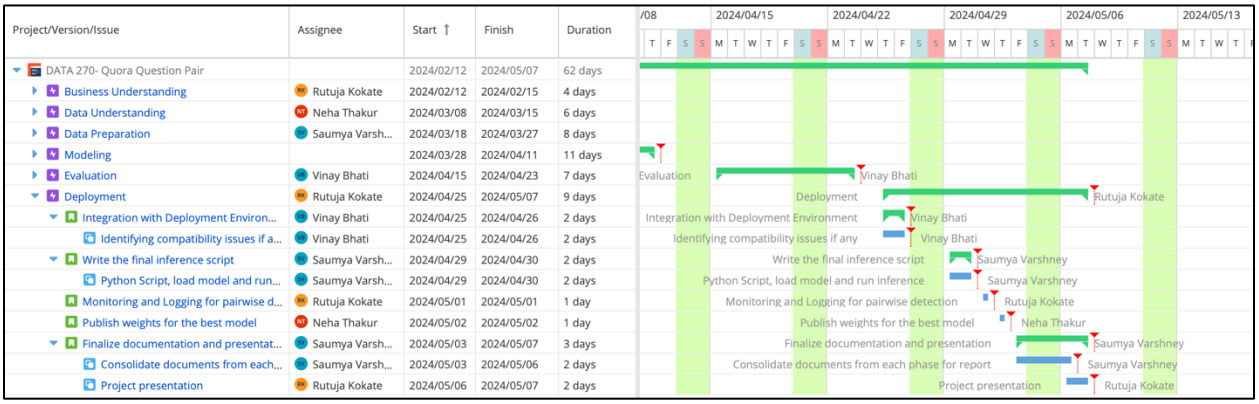


Deployment

This is the final phase, where everything done as part of this project is aligned by the team members. After evaluating and reviewing the process, create an inference script and deploy the model. The entire process, including the group and individual reports, is clearly documented, and reported. This phase, which spans nine days, starts on April 25th, 2024, and ends on May 07th, 2024. The Gantt chart Figure 10 below represents the tasks, timelines, and progress of the Deployment phase.

Figure 10

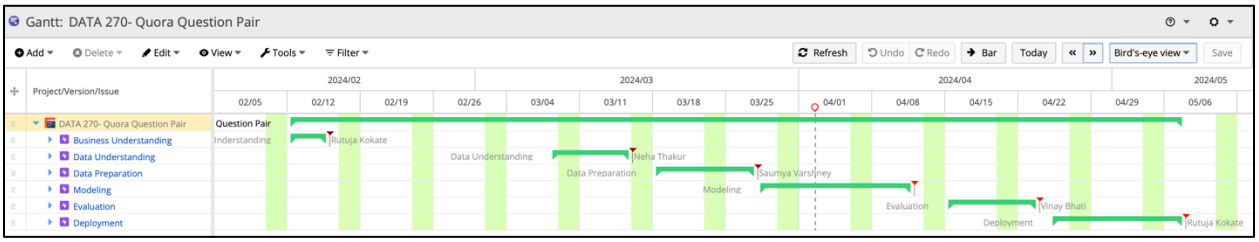
Gantt Chart – Deployment Phase



Below is the bird's eye view of Gantt chart of the project in Figure 11

Figure 11

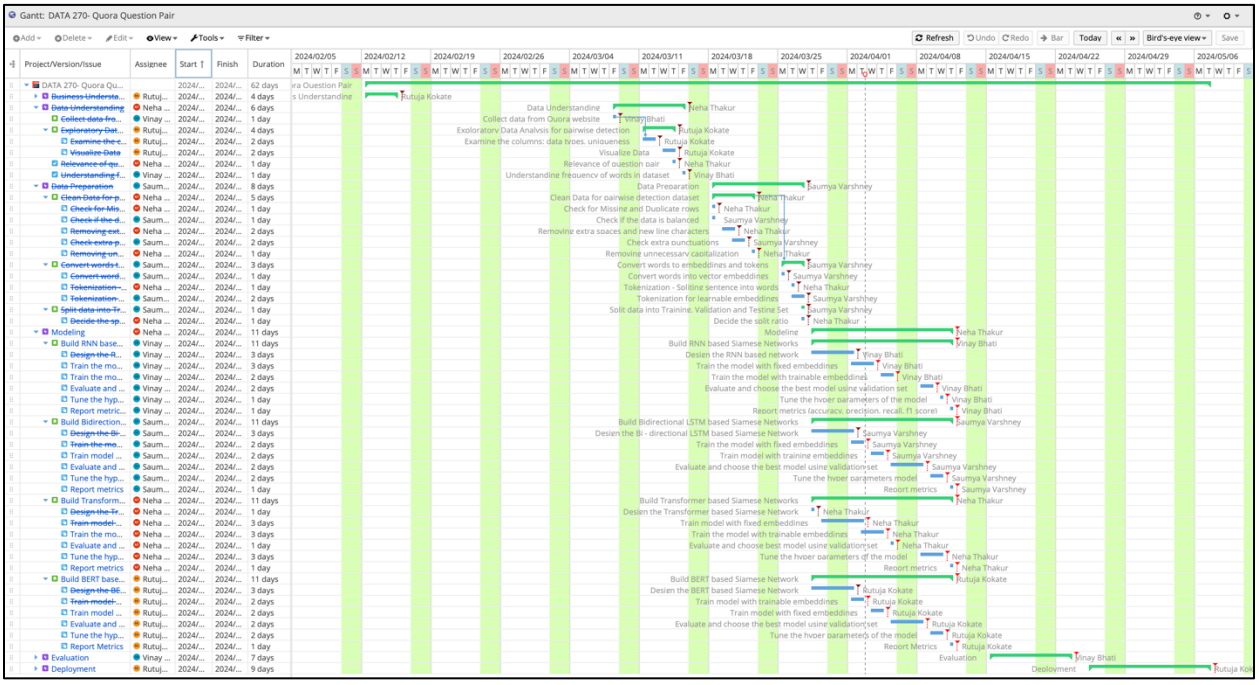
Bird's Eye View of Gantt Chart



Below, Figure 12 is the complete view of the Gantt chart, which represents all the epics, stories, tasks, and sub-tasks with their timelines and dependencies, as well as the team members assigned to each task.

Figure 12

Full View of Gantt Chart – Epics, Stories, Tasks



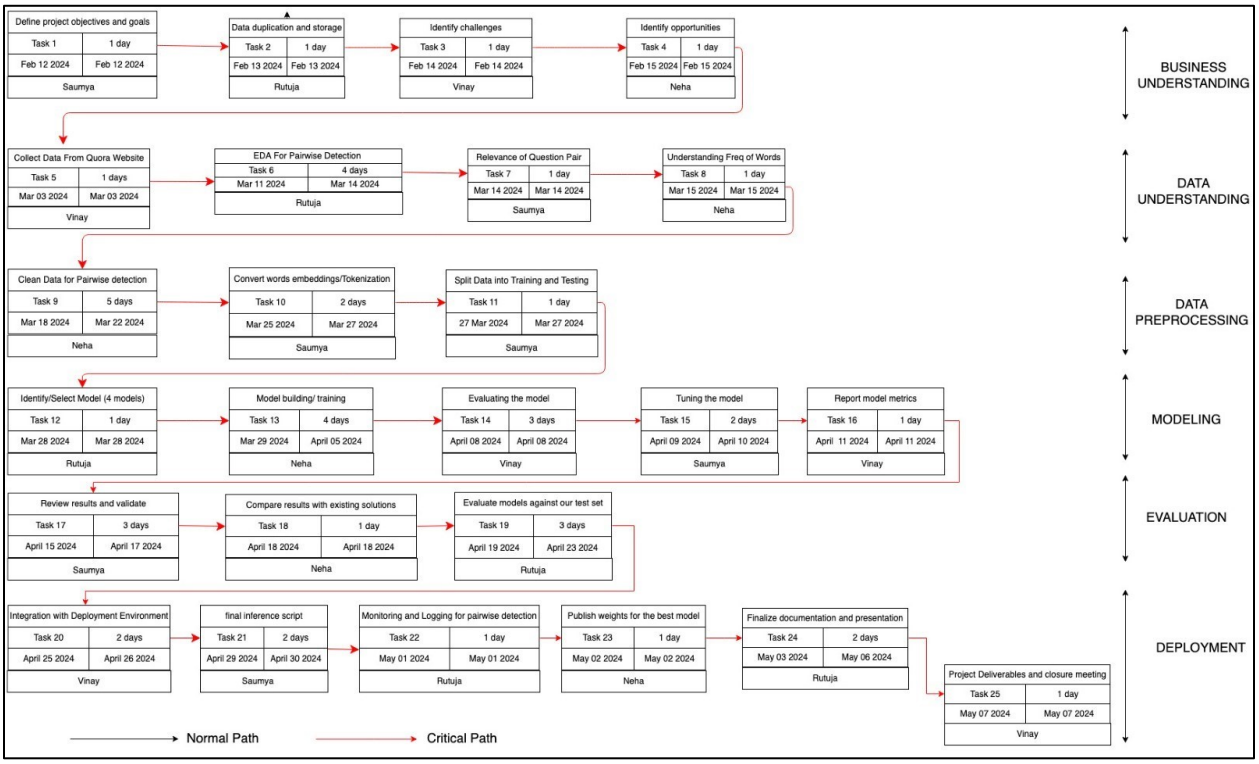
PERT Chart

The Program Evaluation and Review Technique (PERT) chart is a project management tool used to schedule, organize, and coordinate tasks within a project (Visual Paradigm Online, para. 1). PERT charts are particularly useful in planning the tasks that make up a project, estimating their time, understanding the dependencies, and determining a critical path to estimate the minimum amount of time needed to execute all the project tasks.

Figure 13 shows the PERT Chart created in draw.io based on tasks in Jira. It has six phases or lanes, each corresponding to one of the six phases of CRISP-DM. Each block represents a task with details like task name, task ID or task number, duration in days, start and end dates, and member responsibility since this is a task-based PERT chart. Since this is a task-oriented PERT chart, almost every task depends on the previous task to start, and each phase is dependent on the previous phase. Assignees in the Figure 12 below are responsible for the task assigned. However, this is a task-based PERT chart, the sub tasks would be done in collaboration. The critical path is shown in red, along with the sequence of tasks that must be completed for the project to be completed. The project timeline spans over two months starting from 12th February, 2024 to 07th May, 2024.

Figure 13

PERT Chart (Task-Oriented)



Data Engineering Plan

Data Process

A complete data engineering process is carried out on Quora Question Pair (QQP) dataset to prepare it for analysis and modeling on the paraphrase identification task. The QQP dataset is collected from the official website of Quora i.e., Data@Quora. To clean the data, incomplete or missing entries are identified and removed especially in “question1” and “question2” fields where gaps can greatly affect analysis and modeling. Inconsistent data is merged and corrected systematically, including questions with the same text but different question id, and duplicate pairs of questions.

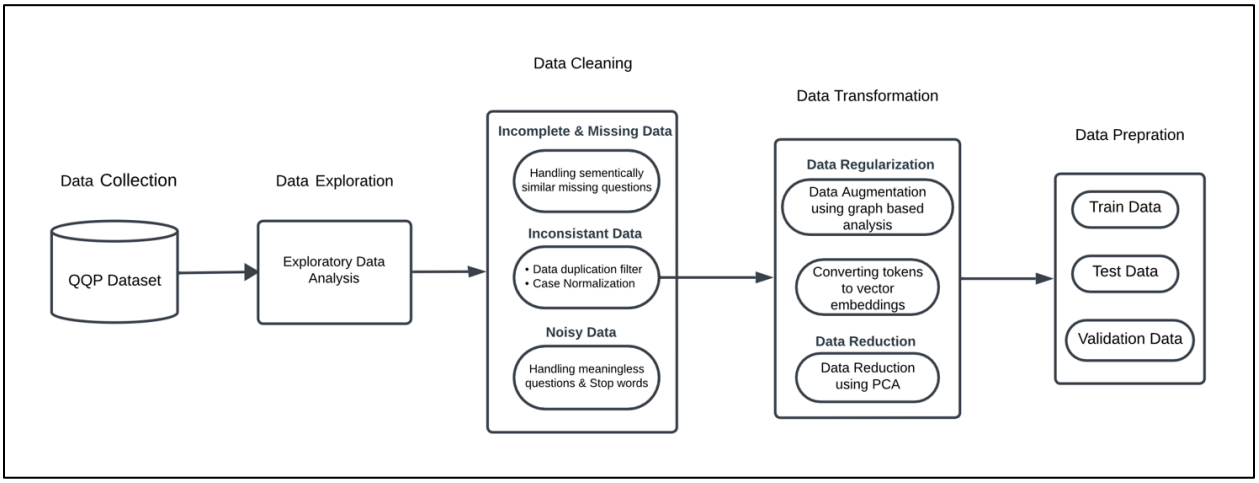
Moreover, noisy data such as meaningless questions, inconsistent capitalization and stop words have been dealt with to improve quality as well as relevance of the dataset. Furthermore, for data augmentation, some more duplicate question pairs are created through graph-based analysis to correct class imbalance. To ensure a balance between memory utilization and preservation of information, the dimensionality of FastText (Grave et al., 2018) embeddings have been reduced from 300 to 100 using Principal Component Analysis (PCA).

To prepare the data for modeling, the cleaned and transformed data set is split into training, testing and validation parts. An extensive exploratory data analysis is conducted to reveal any underlying patterns, trends or associations that may guide further data processing or modeling. Moreover, words are converted into vectorized forms using GloVe (Pennington et al., 2014) and FastText (Grave et al., 2018) embeddings which helped represent texts numerically for ease of computation. Database normalization approaches are applied to make all records comparable with each other across the entire dataset thereby guaranteeing accuracy in machine learning models.

As shown in Figure 14, each step has been taken with continual refinements made towards improving various aspects while ensuring that this collection remains most useful when it comes to creating a reliable system for detecting paraphrases among Quora question pairs.

Figure 14

Overview of Data Process



Data Collection

Data Collection Description and Requirement

The data is obtained from Quora website as shown in Figure 15. The dataset is made up of pairs of questions with a label indicating whether they are duplicates or not, available courtesy of Iyer et al (2017). Each record includes the ids of both the questions, text for 'question1', the text for 'question2', and a binary label ‘is_duplicate’ that denotes if both questions' texts are same. This indicates that all instances of data must have these two components - question texts and corresponding duplication label - to facilitate accurate identification of paraphrases.

For data collection procedure, this dataset is collected by taking it from Quora website since they do not provide any API for programmatically querying and retrieving such information. It comes in form of single .tsv (Tab-Separated Values) file which is about 52 MBs in size containing five feature variables along with one target variable having nearly 400k instances each.

If the dataset had been available via HTTP API, it would have been rate-limited to a certain number of queries per minute. This would mean that the dataset would be downloaded in

batches or chunks, and then all the chunks/batches would be merged to provide the final dataset. Figure 16 shows a detailed data collection plan for the QQP dataset, explaining all key variables utilized in the dataset.

Figure 15

Data collection website page

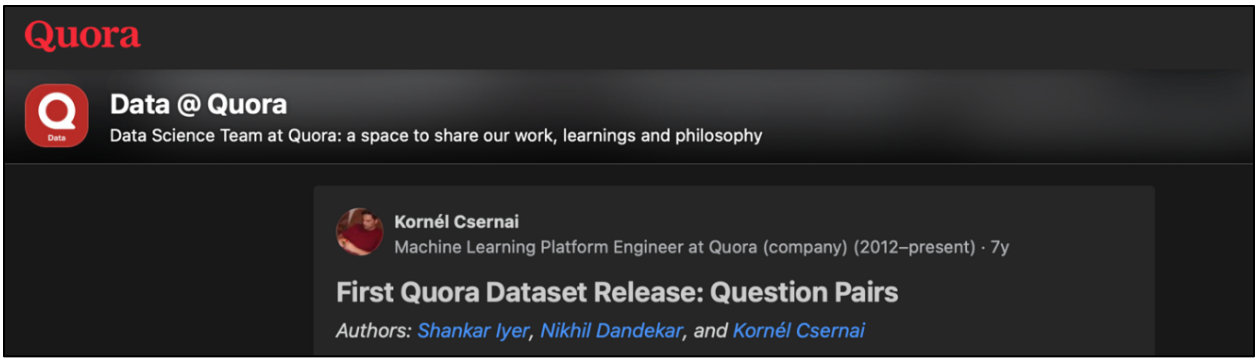


Figure 16

Data Collection Plan

What will be done with the data once it has been collected?

Yes	Identify how well the process is meeting **customer** needs	Yes	Analyze a **problem**, or identify the causes of variation
Yes	Obtain **exploratory view** of the process	Yes	Test a **hypothesis** about the process output
Yes	Evaluate the **measurement system**	Yes	Test a **hypothesis** about the effects of one or more inputs
Yes	Check the **stability** of the process (is it in control?)	Yes	**Control** a process input or monitor a process output
Yes	Conduct a **capability** study		**Other reason…**

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)

		1	2	3	4	5	6
What?	**Variable title**	id	qid1	qid2	question1	question2	is_duplicate
	Input (X) or output (Y) variable?	X	X	X	X	X	Y
	Unit of measurement	N/A	N/A	N/A	N/A	N/A	N/A
	Data type	Integer	Integer	Integer	String	String	Boolean
	Collection method	Automated	Automated	Automated	Automated	Automated	Automated
	If manual						
MSA	Gauge/instrument	N/A	N/A	N/A	N/A	N/A	N/A
	Location	N/A	N/A	N/A	N/A	N/A	N/A
	Gauge calibrated?	N/A	N/A	N/A	N/A	N/A	N/A
	Measurement system checked?	N/A	N/A	N/A	N/A	N/A	N/A
	Precision (R&R) adequate?	N/A	N/A	N/A	N/A	N/A	N/A
	Accuracy adequate?	N/A	N/A	N/A	N/A	N/A	N/A
Historical data	Historical data exist?	No	No	No	No	No	No
	Source of historical data	N/A	N/A	N/A	N/A	N/A	N/A
	Historical data representative/reliable?	N/A	N/A	N/A	N/A	N/A	N/A
	Mean	N/A	N/A	N/A	N/A	N/A	N/A
	Upper specification limit	N/A	N/A	N/A	N/A	N/A	N/A
	Lower specification limit	N/A	N/A	N/A	N/A	N/A	N/A
	Standard deviation	N/A	N/A	N/A	N/A	N/A	N/A
	Target	N/A	N/A	N/A	N/A	N/A	N/A
Sampling	Minimum sample size (MSS)	Entire Dataset	Entire Dataset	Entire Dataset	Entire Dataset	Entire Dataset	Entire Dataset
	Sampling frequency	Only once	Only once	Only once	Only once	Only once	Only once
	Sub-grouping needed?	No	No	No	No	No	No
	Sub-group size	N/A	N/A	N/A	N/A	N/A	N/A
	Stratification needed? (time, shift)	N/A	N/A	N/A	N/A	N/A	N/A
Who?	**Data collector**	Neha, Rutuja, Saumya, Vinay	Neha, Rutuja, Saumya, Vinay	Neha, Rutuja, Saumya, Vinay	Neha, Rutuja, Saumya, Vinay	Neha, Rutuja, Saumya, Vinay	Neha, Rutuja, Saumya, Vinay
	Operational definition exist?	Yes	Yes	Yes	Yes	Yes	Yes
	Data collector trained?	Yes	Yes	Yes	Yes	Yes	Yes
	Resources available for data collector?	Yes	Yes	Yes	Yes	Yes	Yes
When?	**Start date**	8-Mar	8-Mar	8-Mar	8-Mar	8-Mar	8-Mar
	Due date	8-Mar	8-Mar	8-Mar	8-Mar	8-Mar	8-Mar
	Duration (in days)	0	0	0	0	0	0

Samples from Raw Dataset

Below is Table 10, which provides detailed information about all the variables in the dataset.

Table 10

Data Description

Variable Name	Variable Type	Variable Description	Data Type
id	Feature	Unique identifier for each question pair	integer
qid1	Feature	Question ID for first question in pair	integer
qid2	Feature	Question ID for second question in pair	integer
question1	Feature	Textual content of first query	Text/ string
question2	Feature	The second question's text	Text/ string

Variable Name	Variable Type	Variable Description	Data Type
is_duplicate	Target	One means both questions are duplicate while zero means they're not duplicates	Boolean

The entire dataset published by Data@quora is sampled and since the dataset is static, it needs to be sampled only once. Data@quora team has not published any information about how the data has been sampled. If they had published information like, the time from when they collected the data or whether they are sampling questions from a specific topic, it would have provided lot of useful insight for handling this dataset. Below Figures 17 and 18 are the Sample view of the dataset.

Figure 17

Screenshot referenced from the website quoradata.quora.com

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 18

Screenshot below taken from EDA

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

Data Pre-processing

Once the data is collected, there is a need to check whether it is clean to derive features from a complex raw data set so that it can be used for data analysis and modeling. The following steps need to be performed on the data as part of the data pre-processing plan for this project.

Data Cleaning

Here the methods used to clean the Quora Question Pairs dataset are discussed, which is important for paraphrase identification model. Data cleaning includes working with incomplete, noisy, and inconsistent data. Each strategy that is used to fix various kinds of problems with data will be discussed here.

Handling Incomplete and Missing data. Incomplete or missing data is a common issue in many datasets and can significantly compromise the accuracy and reliability of machine learning models. In this project, missing values in critical fields such as "question1" or "question2" directly impact the ability to perform accurate paraphrase detection. These fields contain essential information for comparing text pairs, and any gaps in data could lead to incorrect model training and biased outcomes.

To address these challenges, a structured approach to identify and remove incomplete or missing entries from the dataset is implemented. The process can be summarized in two main steps: Identifying Missing Values and Removing Missing Data.

Identifying Missing Values. The first step is to identify any missing values across key text columns using the Pandas isnull() function. Missing entries in "question1" or "question2" could lead to inaccurate paraphrase detection because these fields contain the data necessary for comparison. Figure 19 is the list of null values in the dataset. Figure 20 shows missing data.

Figure 19

Check for nulls in the data set

```
nulls in each column:
id              0
qid1            0
qid2            0
question1       1
question2       2
is_duplicate    0
```

Figure 20

Missing data from the dataset

	id	qid1	qid2	question1	question2	is_duplicate
105780	105780	174363	174364	How can I develop android app?	NaN	0
201841	201841	303951	174364	How can I create an Android app?	NaN	0
363362	363362	493340	493341	NaN	My Chinese name is Haichao Yu. What English na...	0

Removing Missing Data. The question id corresponding to the question which is null is checked in the dataset to make sure if there is any other entry of the same id with text available. In the below Figure 21, question1 is null and, upon checking further, there is no other instance of corresponding id in the dataset. Hence the row is deleted.

Figure 21

The row that needs to be removed

	id	qid1	qid2	question1	question2	is_duplicate
363362	363362	493340	493341	NaN	My Chinese name is Haichao Yu. What English na...	0

Verifying Data Completeness. The dataset is queried again to check for nulls.

Figure 22

Check for nulls in the data set post cleaning

```
nulls in each column:
id              0
qid1            0
qid2            0
question1       0
question2       2
is_duplicate    0
```

As seen in Figure 22, there are still two missing values in question2 column. This will be handled while handling the inconsistent data because it contains questions that are semantically similar as shown in Figure 23.

Figure 23

Semantically Similar rows

	id	qid1	qid2	question1	question2	is_duplicate
105780	105780	174363	174364	How can I develop android app?	NaN	0
201841	201841	303951	174364	How can I create an Android app?	NaN	0

Handling Inconsistent Data. In this project, ensuring the consistency of data is crucial. Data inconsistencies, if unaddressed, could significantly affect the accuracy of the machine

learning models. Several types of inconsistencies are identified in this dataset, which are addressed systematically to ensure the integrity and usability of the data.

Resolving Null Values. From Figure 22, it is discovered that some entries in the "question2" field are missing. It can be seen from the figure that two questions with missing texts have the same question id. These two rows can be combined into a single row since they are semantically similar. Once the rows are combined the is_duplicate value is changed to 1 and the other row is deleted as seen in Figure 24 below.

Figure 24

Result after resolving inconsistent null value

	id	qid1	qid2	question1	question2	is_duplicate
105780	105780	174363	303951	How can I develop android app?	How can I create an Android app?	1

Figure 25, image of the dataset before cleaning the missing values on the left and post cleaning the missing values in the data set on the right.

Figure 25

Before and After Cleaning the Missing or Null values

```
nulls in each column:
id              0
qid1            0
qid2            0
question1       1
question2       2
is_duplicate    0
```

```
nulls in each column:
id              0
qid1            0
qid2            0
question1       0
question2       0
is_duplicate    0
```

Eliminating Duplicate Data. The dataset contained duplicate entries that could skew the model's learning process by over-representing certain features. To mitigate this, a strategy is

implemented to identify and remove these duplicates. A unique identifier for each question pair is generated, allowing to ensure that each pair is unique as shown in Figure 26. This step is critical in maintaining the quality and diversity of the training data, thereby helping to prevent the model from overfitting.

Figure 26

Checking duplicates by generating a new column uuid

	id	qid1	qid2	question1	question2	is_duplicate	uuid
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0	(1, 2)
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0	(3, 4)
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0	(5, 6)
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0	(7, 8)
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0	(9, 10)

Database Normalization of the dataset. To efficiently query the dataset and understand further inconsistencies the dataset is split into two tables. The first table named question pairs contains ids and is_duplicate field. The second table named question bank contains the question id along with its text.

Standardizing Question IDs. There are multiple instances where different IDs are associated with identical questions as shown in Figure 27. Such discrepancies could potentially confuse the model during training. To streamline this, multiple IDs for identical questions are consolidated into a single ID as shown in Figure 28. This involved creating a mapping from each question text to a unique ID and adjusting the question pair and question bank accordingly. This step reduced complexity and improved the model's ability to learn from a cleaner, more consistent dataset.

Figure 27

Duplicate values in the question bank

	qid	question
13016	25026	?
20794	39204	?
47056	84068	?
96725	161071	?
104101	171925	?
134403	214814	?
208485	312495	?
273065	391451	?
402423	535899	?
36829	42085	? to be deleted
59369	103982	? to be deleted
277629	396761	? to be deleted

Figure 28

Unique duplicate values after consolidating into a single id

	qid	question
13016	25026	?
260779	376791	Why?
25228	47035	What?
17682	33561	deleted
87861	147899	Deleted.

Handling Noisy Data. Eliminating noise and irrelevant information is a crucial step in this project for improving the dataset's clarity and usefulness when training models. It is necessary to make sure that only valuable features are considered by the machine learning algorithms rather than confusing it with unrelated parts of data sets. This section discusses the various types of noise found in this dataset and the methods used to remove that noise.

Removing Meaningless Questions. Identified and eliminated questions that are too short, or which did not make any sense, or which are unlikely to be useful in training an effective

model as shown in Figure 29. These entries included questions with fewer than four characters and those composed only of symbols. Removing these entries helped to refine the dataset, ensuring that the model trained on data that is both meaningful and contextually rich. This cleanup is pivotal in enhancing the overall quality and reliability of the training dataset.

Figure 29

Meaningless questions

	qid	question
102512	169595	ok ?
180461	276676	spam
54029	95429	I'm
216861	323090	Aaas
208199	312129	Edit
164553	255663	Nana
208798	312898	Can?
236655	347631	Spam
109311	179423	What
260779	376791	Why?
226925	335646	????
130637	209607	Does?
25228	47035	What?
189396	288142	Which
327206	453611	Life:
205947	309237	Null
72844	125092	lol ?
297461	419830	civil
325200	451410	Error
141281	224327	null
20072	37899	.....
263134	379636	delete
144506	228688	Delete
329933	456631	Who is
57484	101005	grammar
161082	251091	Who is?
17682	33561	deleted
400153	533491	What is
322705	448565	Cloning?
231151	340885	Marriage

	qid	question
3306	6553	.
13016	25026	?
23884	44699	HH
44619	80055	Na
86457	145814	Is?
108978	178936	i
109009	178982	Hh
115347	188110	o
158778	247989	A
169290	262028	111
175282	269923	Hh
181695	278282	Why
190570	289688	'
199110	300509	I
245880	358814	sss
257077	372328	hi
270146	387976	Ok
301583	424494	.....
325530	451788	My
326297	452592	How
328745	455319	lol
351788	480660	Q?
357127	486520	H
381124	512812	no

qid	question
25026	?
42085	? to be deleted
487791	To be deleted?
247924	? (To be deleted)
376791	Why?
47035	What?
33561	deleted
147899	Deleted.
279869	[removed]
308701	Not needed
388799	Deleted question.

Case Normalization. This dataset included text with inconsistent capitalization, which could lead to the same words being counted as distinct tokens based on case differences alone.

Implementation. All the text data is converted to lowercase to unify the text format, reducing complexity for the models, and ensuring that words such as "Answer" and "answer" are processed as the same token. As shown in Figure 30 below the left figure is before converting it to small case and right is after converting it to small case.

Figure 30

Question text converted to small case

	qid	question
0	1	What is the step by step guide to invest in sh...
0	2	What is the step by step guide to invest in sh...
1	3	What is the story of Kohinoor (Koh-i-Noor) Dia...
17296	4	What would happen if the Indian government sto...
2	5	How can I increase the speed of my internet co...

	qid	question
0	1	what is the step by step guide to invest in sh...
0	2	what is the step by step guide to invest in sh...
1	3	what is the story of kohinoor (koh-i-noor) dia...
17296	4	what would happen if the indian government sto...
2	5	how can i increase the speed of my internet co...

Text Tokenization. An important part of any natural language processing task is to convert sentences into a sequence of words or tokens. Tokenization is the process of splitting sentences or text into individual words or tokens, which are easier for models to process and analyze. This step is also part of data transformation, but this is included here as part of noise removal because it is the base step for removing stop words in the next section.

Implementation. The Natural Language Toolkit (NLTK) is utilized to tokenize the text data as shown in Figure 31 with before and after transformation. This step transformed continuous text strings into lists of words, enabling more detailed and effective linguistic analysis.

Figure 31

Text converted to tokens

	qid	question
0	1	what is the step by step guide to invest in sh...
0	2	what is the step by step guide to invest in sh...
1	3	what is the story of kohinoor (koh-i-noor) dia...
17296	4	what would happen if the indian government sto...
2	5	how can i increase the speed of my internet co...
2	6	how can internet speed be increased by hacking...
3	7	why am i mentally very lonely? how can i solve...
3	8	find the remainder when [math]23^{24}[/math] i...
4	9	which one dissolve in water quikly sugar, salt...
4	10	which fish would survive in salt water?
5	11	astrology: i am a capricorn sun cap moon and c...
5	12	i'm a triple capricorn (sun, moon and ascendan...
6	13	should i buy tiago?
6	14	what keeps childern active and far from phone ...
7	15	how can i be a good geologist?
7	16	what should i do to be a great geologist?

	qid	question
0	1	[what, is, the, step, by, step, guide, to, inv...
0	2	[what, is, the, step, by, step, guide, to, inv...
1	3	[what, is, the, story, of, kohinoor, (, koh-i-...
17296	4	[what, would, happen, if, the, indian, governm...
2	5	[how, can, i, increase, the, speed, of, my, in...
2	6	[how, can, internet, speed, be, increased, by,...
3	7	[why, am, i, mentally, very, lonely, ?, how, c...
3	8	[find, the, remainder, when, [, math,], 23^, ...
4	9	[which, one, dissolve, in, water, quikly, suga...
4	10	[which, fish, would, survive, in, salt, water, ?]
5	11	[astrology, :, i, am, a, capricorn, sun, cap, ...
5	12	[i, 'm, a, triple, capricorn, (, sun, ,, moon,...
6	13	[should, i, buy, tiago, ?]
6	14	[what, keeps, childern, active, and, far, from...
7	15	[how, can, i, be, a, good, geologist, ?]
7	16	[what, should, i, do, to, be, a, great, geolog...

Removing Stop Words. The text data included frequently used words (stop words) that do not contribute significant meaning to the sentences. These words, such as "is", "and", "the" can distract models from focusing on the meaningful keywords that are critical for detecting paraphrases. Examples of stop words are listed in Figure 32.

Figure 32

Stop Words

```
['other',
 'hasn',
 'to',
 'had',
 'be',
 'few',
 'won',
 'needn',
 'me',
 'we',
 'isn',
 'there',
 "didn't",
 'can',
 'they',
 'whom',
 'aren',
 'very',
 'here',
```

Implementation. A predefined list of stop words from NLTK is used to remove stop words from the tokenized text. Figure 33 represents data before on the left-hand side and after removing stop words on the right-hand side. This step helped in focusing the model's attention on significant words, enhancing both the efficiency and accuracy of the paraphrase identification process.

Figure 33

Post removing stop words

	qid	question
0	1	[what, is, the, step, by, step, guide, to, inv...
0	2	[what, is, the, step, by, step, guide, to, inv...
1	3	[what, is, the, story, of, kohinoor, (, koh-i-...
17296	4	[what, would, happen, if, the, indian, governm...
2	5	[how, can, i, increase, the, speed, of, my, in...
2	6	[how, can, internet, speed, be, increased, by,...

	qid	question
0	1	[step, step, guide, invest, share, market, ind...
0	2	[step, step, guide, invest, share, market, ?]
1	3	[story, kohinoor, (, koh-i-noor,), diamond, ?]
17296	4	[would, happen, indian, government, stole, koh...
2	5	[increase, speed, internet, connection, using,...
2	6	[internet, speed, increased, hacking, dns, ?]

Removing Meaningless Questions. Post-stop word removal, some sequences in the dataset had questions reduced to two or fewer meaningless tokens as shown in Figure 34. These entries are unlikely to provide valuable context for analysis and could degrade model performance.

Implementation. These sequences are filtered out and removed from the dataset. This is important to maintain the quality and relevance of the training and testing data.

Figure 34

More Meaningless Questions

	qid	question
493	984	[study]
151378	13922	[?]
9581	18608	[?]
9581	18609	[?]
29040	53770	[?]
29040	53771	[?]
35641	65095	[?]
35641	65096	[?]
39405	71436	[]
42947	77322	[?]
278091	82154	[?]
48233	86035	[possible]
65766	114086	[?]
67219	116374	[?]
68665	118624	[?]
252045	118625	[?]
69371	119713	[?]
74304	127361	[parisflatlist]

Data Transformation

Data transformation refers to changing data from one structure or format to another. It is a basic element of most data integration and data management tasks, such as data wrangling, data warehousing, application integration and data integration.

Data Regularization

Data Regularization is the process of adding new information to the dataset to prevent overfitting. This new information can be provided in the form of constraints like L1 and L2 regularization or by generating new data examples from the existing data using domain specific knowledge or by removing data examples to reduce bias. For Quora question pairs dataset, data augmentation strategy is used to introduce new duplicate question pair examples. The new question pairs are generated through graph-based analysis. This helps deal with class imbalance by creating additional instances belonging to minority group, which in case of this dataset are the

duplicate question pairs, so that they can be better recognized during classification process thus raising accuracy rates for both categories.

Data Augmentation. There is class imbalance in the dataset, since there are only 37% duplicate question pair samples. To remove this imbalance, new duplicate question pair examples are generated. The existing duplicate questions pairs can be represented as an undirected graph where the question ids are nodes, and an edge exists between the question ids if they are duplicate. After constructing the graph all the connected components are identified.

Each connected component contains a set of question ids that can be considered similar. For each connected component, all possible pairs of edges are generated between the nodes. Then the edges that already exist in the graph are filtered out. All the remaining edges are new question pairs that are duplicate of each other.

For each new edge discovered corresponding nodes is added as qid1, qid2 in the question pairs dataset and marked them as duplicates as shown in Figure 35. Now, there are 48% duplicate question pair samples in the dataset.

Figure 35

Augmented data after generating new edges

	qid1	qid2	is_duplicate
0	31	1100	1
1	31	2066	1
2	31	2067	1
3	31	6079	1
4	31	6080	1
5	31	6938	1
6	31	7751	1
7	31	7752	1
8	31	8577	1
9	31	8578	1

Converting word tokens in question text to individual token ids

Each unique word in the tokenized dataset is assigned a unique token ID. In the question bank, each word token is replaced by the individual token ID as shown in Figure 36.

Figure 36

Word token in Question bank replaced with token id

	qid	question
0	1	[step, step, guide, invest, share, market, ind...
0	2	[step, step, guide, invest, share, market, ?]
1	3	[story, kohinoor, (, koh-i-noor,), diamond, ?]
17296	4	[would, happen, indian, government, stole, koh...
2	5	[increase, speed, internet, connection, using,...

	qid	question
0	1	[95871, 95871, 48581, 55833, 91503, 64763, 542...
0	2	[95871, 95871, 48581, 55833, 91503, 64763, 11583]
1	3	[96193, 59590, 2019, 59587, 2020, 35327, 11583]
17296	4	[109506, 49465, 54299, 47739, 96101, 59590, 20...
2	5	[54136, 94759, 55511, 30115, 105170, 107026, 1...

Converting word tokens to Vector Embeddings

Words are represented as vector embeddings that permit the models a chance to deal with text using numerical representation and thus allow for the semantic sense of the words to be captured. This project used GloVe which stands for Global Vectors for Word Representation (Pennington et al., 2014) and FastText (Grave et al., 2018) approaches for these purposes.

The general process used to convert the tokens to their corresponding vector embeddings using GloVe and fasttext is as follow:

- Identify all the tokens in the dataset's vocabulary that are not present in the embedding's vocabulary.
- For each token found in the above step generate an embedding for this token using various strategies. For GloVe, the token embedding is the mean of all its sub tokens for which an embedding is present. The fasttext model provided a utility which would

automatically generate an embedding for a token that is previously not seen. This is possible because it is trained using continuous bag of words strategy.

- All the tokens from the embedding vocabulary and from the dataset vocabulary are stored in a new file along with their corresponding vector embeddings.

FastText Embeddings. Here, each word in the vocab is assigned a 300-dimensional vector. The 300 dimensions are chosen because it is the default embedding dimensions for fasttext. The fasttext embedding model is available courtesy of Grave et al (2018).
which contains all the words it has already learned and its corresponding embeddings and loaded it into the model. Below Figures 37 and 38 are the examples of vector embeddings of 300d fasttext.

Figure 37

Vector embeddings for token chill with 300d fasttext

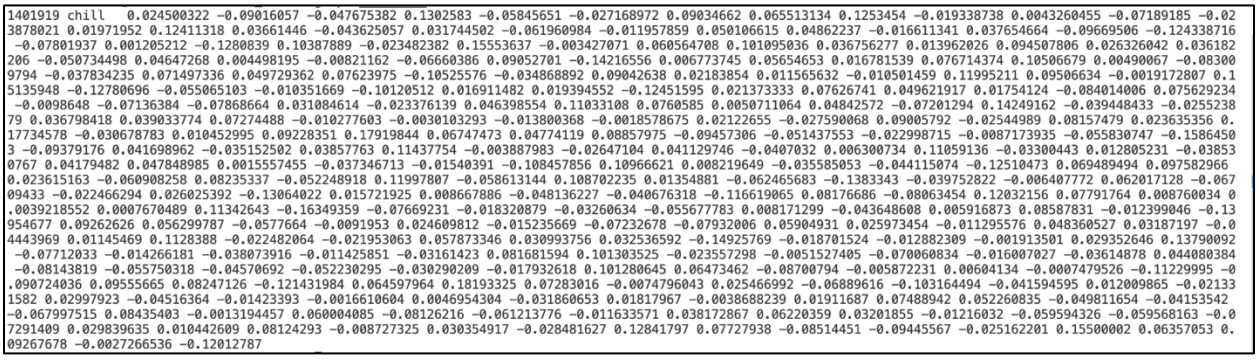
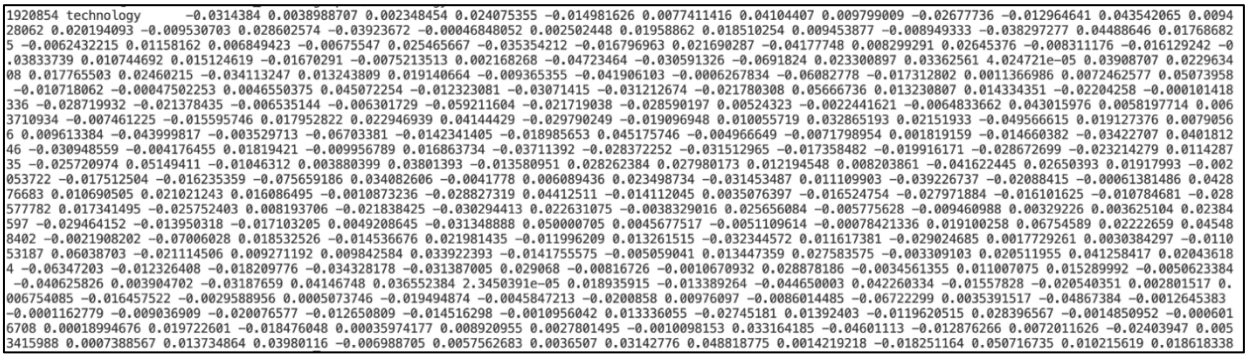


Figure 38

Vector embeddings for token technology with 300d fasttext



GloVe Embeddings. In this embedding as well, 300 dimensional embeddings are taken. The embeddings model is available courtesy of Pennington et al. (2014) and then embedded with vocab.csv to get the corresponding embedding for a token. Below Figures 39 and 40 are the examples of vector embeddings of 300d GloVe.

Figure 39

Vector embeddings for token chill with 300d GloVe

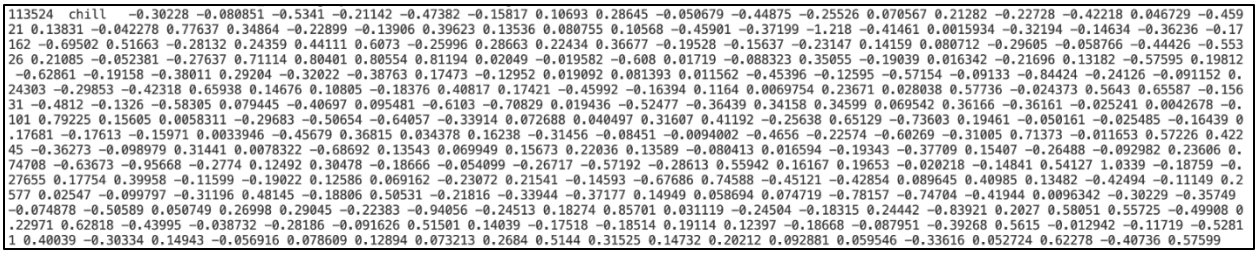
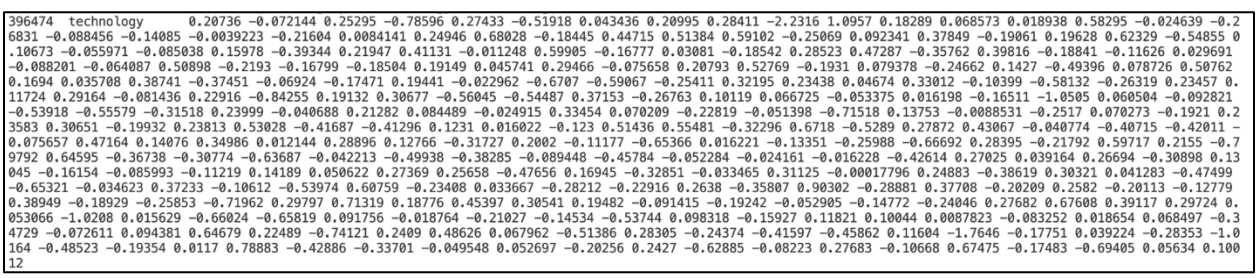


Figure 40

Vector embeddings for token technology with 300d GloVe



The major difference between GloVe and Fasttext embeddings is that FastText can handle OOV (Out of vocabulary) words more effectively because it constructs sub word representations. It can generate embeddings for OOV words by summing the embeddings of their sub words. GloVe on the other hand, has limitations in handling OOV words as it relies on the co-occurrence matrix. To overcome this a similar approach like fasttext is used to generate embedding for OOV vectors in GloVe. The strategy has been described previously.

Data Reduction using PCA

Data reduction is important because it decreases the time and memory needed for training models or making inferences. One approach commonly used for data reduction is using dimensionality reduction, wherein the dimensions of a vector are reduced to a lower dimension while preserving the information contained in the vector. PCA (Principal Component Analysis) is one such way of dimensionality reduction, where the dimensions of a vector are reduced by analyzing the variance in the dataset and transforming the axis to align with the variance present in the dataset.

The data reduction is done using PCA for both Fasttext and GloVe embeddings. Both Fasttext and GloVe provide 300-dimension embeddings for each token. Using PCA, this is reduced to 100-dimension. 100 is chosen as the final dimension for embeddings to provide a balance between memory and the amount of information that would be lost due to dimensionality reduction.

Fasttext does principal component analysis internally. With the reduce_model function of fasttext library, PCA has been done. Below Figures 41 and 42 are the examples of vector embeddings after PCA for fasttext.

Figure 41

Vector embeddings for token technology with fasttext before PCA

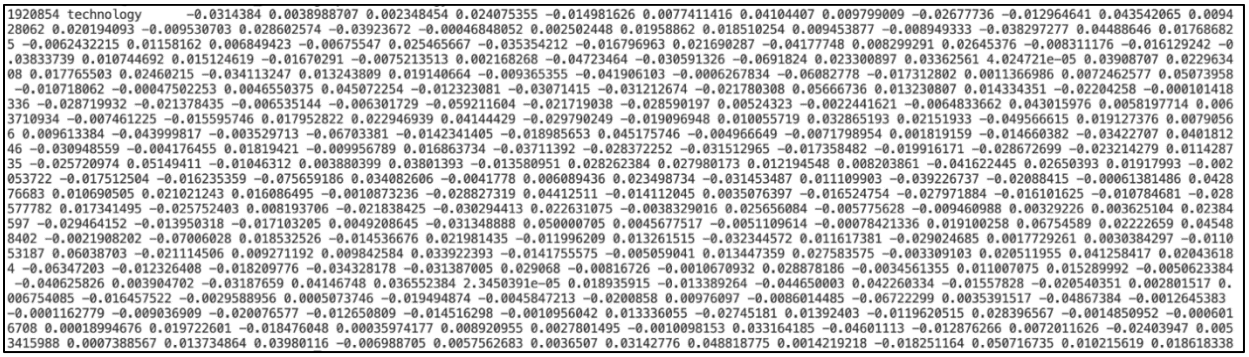
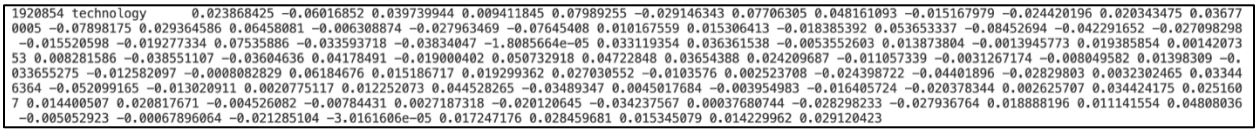


Figure 42

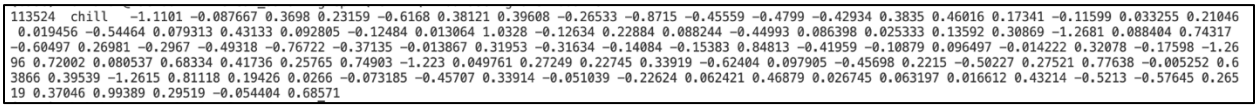
Vector embeddings for token technology with 100d fasttext after PCA



For GloVe, 100 dimensions vector embeddings are already available, and that has been taken. Below Figures 43 is an example of vector embeddings for 100d GloVe.

Figure 43

Vector embeddings for token technology with 100d GloVe



Data Preparation

To guarantee the strength and usefulness of training, validating, and testing processes required for creating powerful machine learning models, the datasets are carefully prepared

during the Data Preparation stage. Many tactical measures are taken during data preparation to maximize the usefulness of the information in various stages of model development.

Dataset Consolidation

The dataset, question_pairs.csv, has been cleaned-up and consolidated and contains pairs of questions tagged with whether they are duplicate. Each pair of questions consists of two question IDs (qid1 and qid2) and a flag indicating if the questions are duplicates (is_duplicate). This format is critical as it feeds directly into the training algorithm, enabling the training of models capable of correctly identifying paraphrased instances.

Data Splitting

The dataset is divided into three different sets: training, validation, and testing. This step is important as it allows for progressive training of the models, fine-tuning them under different situations to prevent overfitting and to ensure they generalize well to new, unseen data. The train_test_split function from the sklearn.model_selection module is utilized, which supports random shuffling and stratification of the data, an extremely helpful feature. Initially, 80% of the dataset is used for training, while setting aside 20% temporarily as a testing set. This temporary testing set is further bifurcated to create actual testing and validation sets, each accounting for 50% of the total temporary set as shown in Figure 45. In this way, every individual part represents the overall distribution more effectively than other methods.

Setting Proportions

Training Set. The largest portion, containing 80% of the data, is used to train the models. This dataset helps the model learn the underlying patterns without being exposed to the nuances of the validation and test datasets.

Validation Set. Half of the initial 20% split, now effectively 10% of the total data, is used to tune the models' hyperparameters and make decisions about model iterations, architecture, and features.

Testing Set. The remaining 10% serves as the final evaluation set, used only after model training and validation are complete. This set is crucial for assessing the final model's performance and its generalization capability for new data.

Storage and Accessibility. Figure 44 shows after splitting, each dataset is saved into separate CSV files within a clean data directory. This organization supports easy accessibility and maintainability of the data throughout the project lifecycle. The specific files created are:

- Figure 47 shows training data from train.csv,
- Figure 48 shows validation data from validation.csv, and
- Figure 49 shows testing data from test.csv.

These files are stored without indexing to maintain consistency and ease of use in subsequent data loading operations during the modeling phase.

Figure 44

Files After Splitting

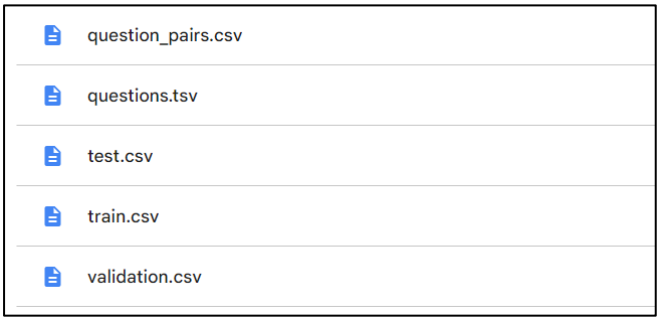


Figure 45

Data Splitting Proportion

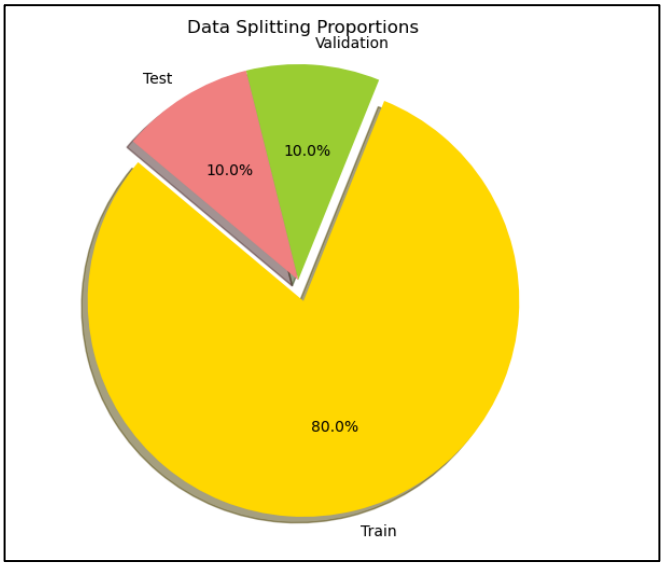


Figure 46

Samples in the dataset

```
Sample Distribution of Original Data Split
Number of samples in training set for original dataset: 387577
Number of samples in validation set for original dataset: 48447
Number of samples in test set for original dataset: 48448
```

Figure 47

Representation of Sample Train Data

	question1	question2	is_duplicate
2	[95871, 95871, 48581, 55833, 91503, 64763, 54248, 11583]	[95871, 95871, 48581, 55833, 91503, 64763, 11583]	0
3	[96193, 59590, 2019, 59587, 2020, 35327, 11583]	[109506, 49465, 54299, 47739, 96101, 59590, 2019, 59587, 2020, 35327, 19072, 11583]	0
4	[96193, 59590, 2019, 59587, 2020, 35327, 11583]	[31131, 54248, 58691, 59587, 88537, 11583]	0
5	[109506, 49465, 54299, 47739, 96101, 59590, 2019, 59587, 2020, 35327, 19072, 11583]	[24546, 69794, 54248, 46682, 59590, 35327, 19072, 11583]	0
6	[66143, 62583, 11583, 94127, 11583]	[43351, 85720, 11585, 65169, 11767, 6856, 111705, 6869, 111757, 11585, 2986, 11767, 36449, 6872, 11583]	0

Figure 48

Representation of Sample Validation Data

```
question1  question2  is_duplicate
[86313, 61016, 13448, 57036, 107880, 81648, 11583] [60997, 57036, 107880, 34771, 11583]  1
[70660, 104388, 22668, 35076, 14845, 87548, 11583] [80052, 87545, 109092, 104390, 22668, 11583]  0
[107751, 64073, 68101, 74303, 11583]  [38330, 68101, 74303, 11583]  1
[92948, 54945, 21408, 51689, 87431, 63584, 46682, 75232, 91634, 3338, 54049, 100272, 4390, 54049, 61301, 11583] [37356, 109329, 45899, 35042, 29477, 61841
[22508, 62090, 48259, 107648, 11583, 2178, 35494, 74187, 20920, 54554, 11583, 105099, 11583]  [50456, 102170, 20920, 62090, 44446, 80308, 25209, 44446,
```

Figure 49

Representation of Sample Test Data

```
question1  question2  is_duplicate
[42535, 78252, 61094, 43732, 34113, 11583] [35489, 92698, 61094, 43724, 5, 37667, 61094, 43724, 11583] 0
[29374, 101733, 56431, 105196, 11583]  [29374, 105196, 6, 101733, 81415, 11583]  1
[65664, 104944, 109287, 11881, 63741, 7, 11583] [65664, 98635, 102783, 11583, 104944, 109287, 11583]  0
[109506, 20920, 100984, 61694, 28592, 11583]  [105115, 90369, 61694, 28592, 85437, 11583] 0
[20819, 55200, 82079, 57519, 97547, 89042, 34774, 96572, 11583] [16254, 83360, 109616, 27020, 108469, 65876, 11583] 0
```

Data Statistics

Table 11 provides a comprehensive view of the data transformation process for the Quora Question Pairs dataset, beginning with an initial count of 404,290 question pairs. During the pre-processing phase, null values and inconsistencies are addressed, ensuring no data loss and retaining all original entries. The removal of noise and meaningless questions enhanced the quality of the data, maintaining the full count of question pairs.

During the transformation phase, the text data is normalized and tokenized to facilitate uniform processing and comparison across records. Additionally, tokenization helped in structuring the data more effectively for machine learning applications, maintaining the original count of records. These processed entries are then ready for the next stage without any loss in data quantity.

Figure 46 shows, In the preparation phase, stratified sampling is employed to split the dataset into training, validation, and test subsets to ensure a representative distribution of data across each set. This division resulted in 387,577 pairs for training, designed for robust model

training; 48,447 pairs for validation to fine-tune model parameters; and 48,448 pairs for testing, facilitating the objective assessment of the model's performance.

The detailed statistical breakdown in Table 11 will outline these subsets and their specific roles in the project lifecycle, emphasizing the strategic division and utilization of data for effective model training and evaluation. This structure ensures that each subset is optimally used for its intended purpose, from model tuning to performance testing, illustrating the methodical approach to data handling and model development.

Below Table 11, which provides detailed information about statistics in the dataset.

Table 11

Data Statistics

Stage	Process	Statistics
Raw Data	Initial Record Count	404,288 Question Pairs
	Null Values	404,288 Question Pairs (No Null Values)
Pre-processing	Handling Missing Value	1 missing in Q1 Question Pairs
	Handling Inconsistent Data	2 in Q2 Question Pairs
	Duplicate Removal	404,287 Unique Question Pairs
Data Split	Train-Test Split	Train: 387,577, Test: 48,448
	Test-Validation Split	Test: 24,224, Validation: 24,224
Transformation	Tokenization and Cleaning	Tokens mapped and cleaned (404,287)

Stage	Process	Statistics
	Stop Word Removal	Irrelevant tokens removed.
Data Output	Training Data Files	Saved as train.csv
	Validation Data Files	Saved as validation.csv
	Test Data Files	Saved as test.csv

Figure 50

Distribution of Question length with Frequency

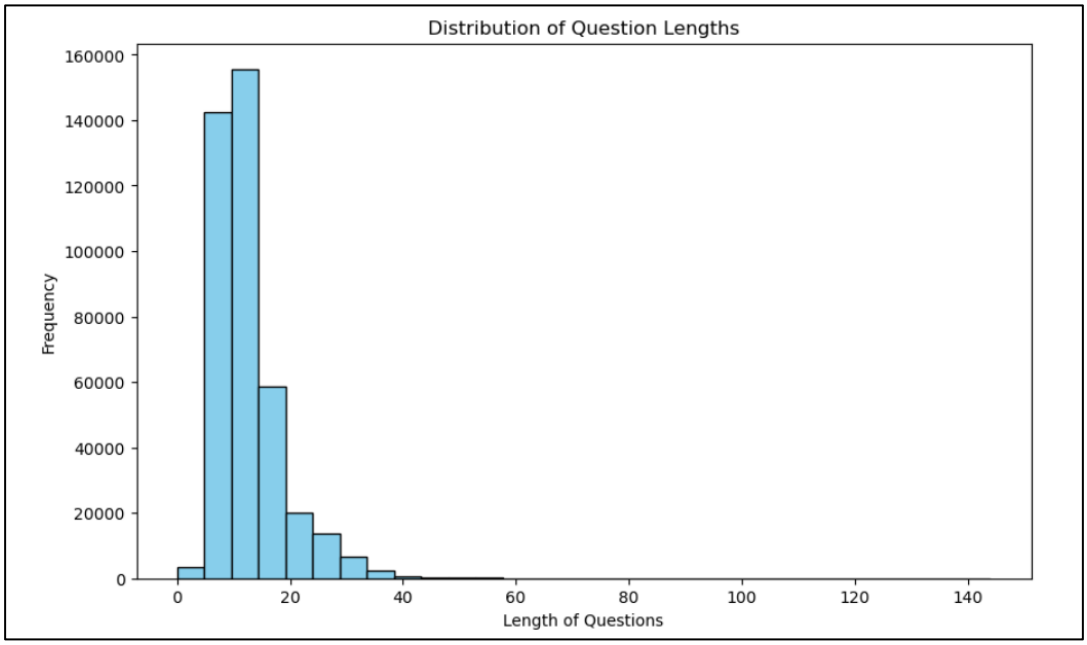
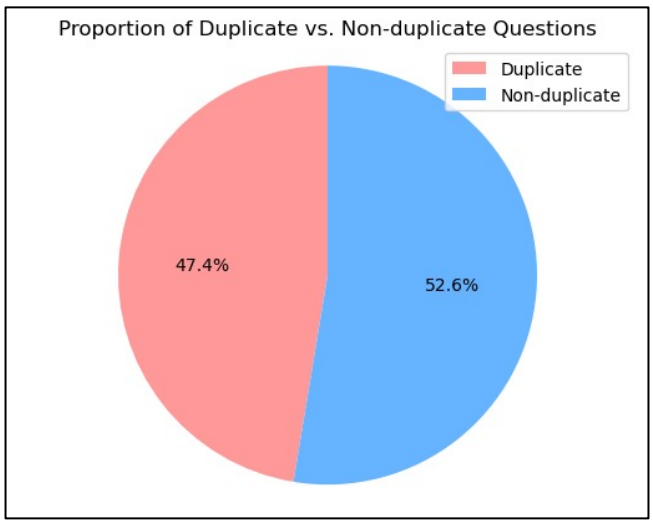


Figure 51 pie chart illustrates the balance achieved in dataset following the data transformation phase. As shown, the dataset exhibits a near equal distribution between duplicate and non-duplicate questions, with 47.4% labeled as duplicates and 52.6% as non-duplicates. This balanced distribution is critical for machine learning models, ensuring that they are not biased toward one class and can generalize well to new, unseen data. Such equilibrium in the dataset

aids in enhancing the accuracy and reliability of paraphrase detection system, thereby facilitating robust model training and validation.

Figure 51

Pie chart showing balanced data post Data transformation.



Data Analytics Results

The Results of Data Analytics part gives a detailed examination of various kinds of visualization, which offers us an in-depth understanding towards the features and intricacies of the dataset. Explored different facets of data through visualizations such as distribution between duplicate/non-duplicate question pairs, semantic similarity, and structural patterns. These visuals provide different viewpoints that help in knowing what should be considered when building models, evaluating them against real world use cases for the paraphrase identification system.

Figure 52 shows Distribution of Duplicate vs. Non-duplicate Questions is a basic visualization that tells us a lot about the dataset and how to prepare it in modeling. Working on

paraphrase identification using Quora question pairs; this visualization helps us understand the distribution of duplicate and non-duplicate question pairs.

It is a bar chart with two categories: '0' for non-duplicate pairs, and '1' for duplicate pairs. The dataset contains about 225k non-duplicate question pairs and roughly 150k duplicate question pairs after analyzing this chart.

The difference between these numbers shows how imbalanced the dataset is when it comes to classifying questions as duplicates or not duplicates. In other words, there are many more examples where both questions belong to different topics than cases where they ask about the same thing twice (around 75k).

This information is important since it means that most observations fall under the non-duplicates category rather than duplicates one.

Figure 53

*Bar chart with Distribution of Duplicate vs. Non-duplicate Question*s

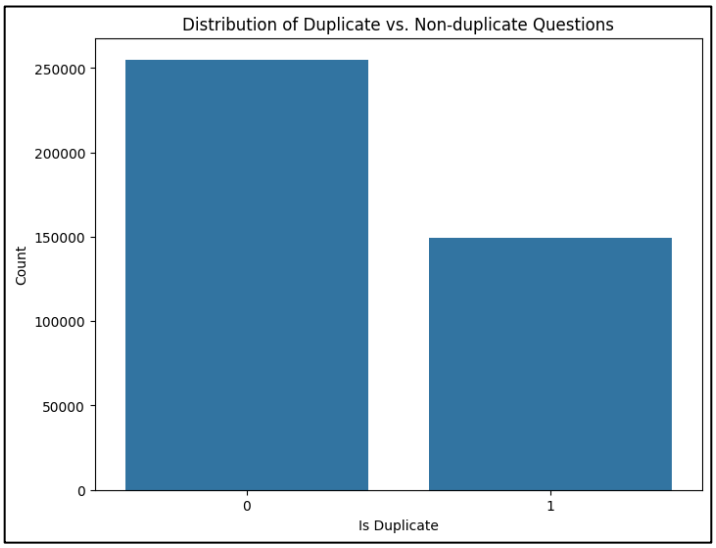
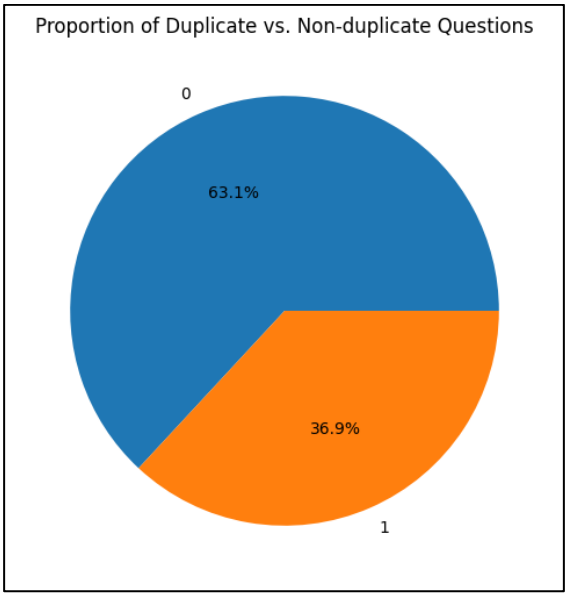


Figure 53 shows the relationship between the proportions of questions that are duplicates and those that are not is shown in a pie chart, and this gives an interesting summary of what the dataset represents. This pictorial representation brings out an issue of class imbalance in data where majority classes dominate with 63.1% being non-duplicate while only 36.9% represent duplication among question pairs. Such a skewedness can be problematic for machine learning algorithms which often have difficulties making accurate predictions about minority classes because they are biased towards the larger ones. This becomes very important in the task on paraphrase identification based on Quora question pairs where such observations should not be taken lightly at all. It allows us to know how duplicates correlate with non-duplicates throughout different stages during the development process of model considering thus help in selecting strategies about oversampling, using synthetic data generation or ensemble methods etcetera for improving generalization ability over unseen data by models affected due to under-representation. Furthermore, having a clear distinction on size relationships from pies serves as basic criterion used when evaluating various indicators suitable to determine accuracy levels attained when dealing with imbalanced classes while trying to recognize different forms of saying something.

Figure 54

Pie chart indicating the proportions of Questions



The dataset has questions of varying lengths. According to Figure 54 showing histogram, this is the most common type of inquiry on Quora, where most are less than 200 characters long. A graph can be used to compare frequencies and lengths between two different types of queries given that they have been labelled as 'Question 1' and 'Question 2'. It is fascinating that a great majority falls into the relatively short category which usually involves those with 200 or fewer words. This means that in Quora dataset there are many brief questions.

The number of longer questions drops considerably after they exceed the length of 200 characters which indicates their rareness. This decline should be considered while distinguishing between paraphrases during the identification process since it provides insight into how often people ask longer queries.

It is important to mention that these findings can also impact data cleaning before building models, for example neural networks where padding might need to be applied during training, so all inputs have equal lengths.

Figure 55

Histogram showing varying lengths of question pairs

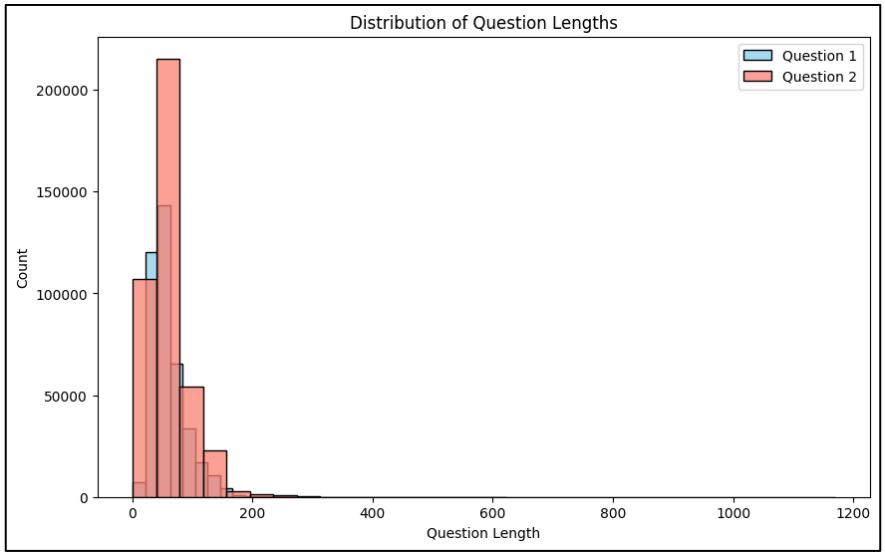
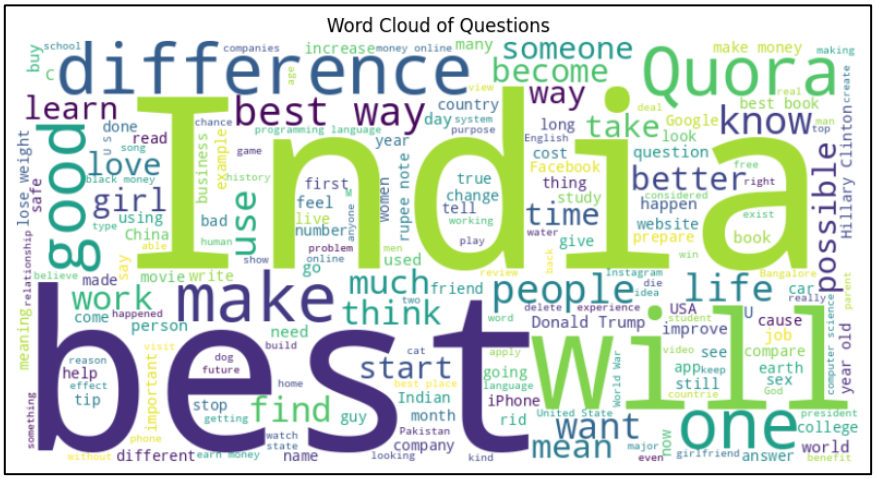


Figure 55 shows the word cloud of Quora question pairs dataset represents the frequently asked questions in this platform with larger fonts indicating higher number of times asked. Many frequently used words such as “difference”, “best”, “Quora” and “India” among others are indicative of the most popular themes or topics that users want to know about. This representation can be used to identify general subjects and key terms that may help in detecting duplicates or paraphrases. For instance, words like ‘make’, ‘people’ and ‘think’ appear quite often which implies personal advice giving or subjective inquiries asking for opinions; these types of questions are difficult to recognize as rephrased ones due their subtle differences in meaning based on context. Not only does it show what areas the dataset concentrates on, but it

also emphasizes how much comprehension relies on context and semantics when developing effective models for recognizing the same meaning statements.

Figure 56

Word cloud depicting the most common words in question pairs



The visualization shown in below Figure 56 for “Pairwise Relationships” does a deep dive into the Quora dataset’s question lengths in relation to their duplicate status. The axes have marginal histograms which show the distribution of lengths for ‘Question 1’ and ‘Question 2’. Both histograms are dominated by shorter questions but as length increases, there is a steep decline. Scatter plots compare lengths between ‘Question 1’ and ‘Question 2’ according to whether they are duplicates or not (orange for duplicates, blue otherwise). Many duplicate questions have similar lengths; this can be deduced from the fact that there is an aggregation of orange dots along the line of equality (where 'Question 1' length equals 'Question 2' length). Conversely, non-duplicates exhibit greater disparity in how their two questions’ lengths relate. Such a visual pattern suggests that models might find it helpful to treat closeness in length as

indicative when trying to predict if pairs of questions are copies, thus motivating better paraphrase identification through feature engineering around this signal.

Figure 57

Pairwise relationship between question pairs.

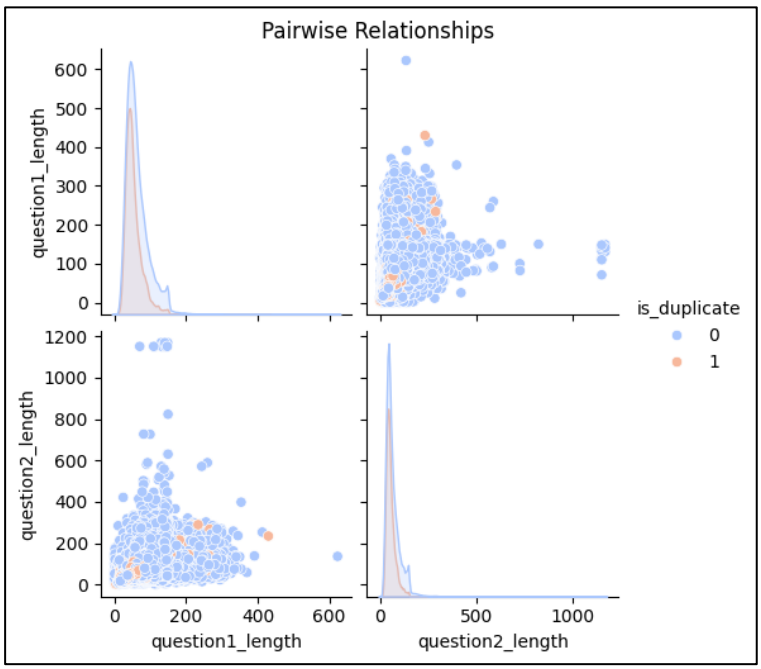


Figure 57 showing pie chart exhibits a nearly equal distribution between duplicate and non-duplicate questions in the Quora dataset after data augmentation; those account for 47.4% and 52.6% correspondingly. Before data augmentation, the fraction of duplicate questions is 36.9%, and non-duplicates are 63.1%. This is a significant change, meaning the two classes have been brought closer to balance by data augmentation.

Data synthesis may have been used for increasing the number of duplicated question pairs or even under sampling the non-duplicated class to make it less frequent so that machine learning models could benefit more from this dataset during its training process. Such an approach helps in avoiding bias towards any category since both will be equally represented

thereby enabling better comprehension of paraphrase identification tasks through models that have been trained using such balanced datasets as they tend to give higher levels of accuracy.

Model Development

Model Proposals

Siamese Network with RNN, GloVe and FastText embeddings

The proposed model employs a Recurrent Neural Network (RNN) for paraphrase identification. This model selection is dictated by the recurrent property of RNN which makes it good at handling series of data, suitable for this kind of natural language processing task. RNN can capture temporal dependencies and subtleties in textual data as they have memory, and this is crucial in distinguishing between paraphrased and non-paraphrased question pairs.

Mathematical Representation of RNN. The operation within a RNN unit can be mathematically represented and derived as shown in Figure 58.

Figure 58

Mathematical formula to represent RNN

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

Where,

- ht is the hidden state at time t,
- xt is the input at time t,
- Wxh and Whh are weights for the input and previous hidden state,
- bh is the bias, and
- σ denotes the sigmoid activation function.

In exploring natural language processing tasks, Agarwal et al. (2018) highlighted the efficacy of recurrent neural networks (RNNs) by integrating them with convolutional neural networks (CNNs) in their hybrid model for paraphrase detection. This combination utilized the RNN's ability to capture temporal dependencies, achieving a notable F1-score of 0.751 in diverse text environments. Similarly, leveraged RNNs, enhanced by LSTMs, for sentiment analysis, achieving an accuracy of 89%. These studies underscore the adaptability and precision of RNNs in handling complex linguistic patterns, validating their use in tasks that require deep linguistic understanding, such as the paraphrase identification in your RNN project.

Siamese Network with GRU (Gated Recurrent Unit), GloVe and FastText embeddings

The Gated Recurrent Unit (GRU) model is an effective choice for paraphrase detection because of its sophisticated ability to handle sequential data and capture long-term dependencies; this is especially true when using datasets like the Quora Question Pairs. Recurrent neural network (RNN) version GRU maintains performance advantages over long short-term memory (LSTM) networks and regular RNNs but simplifies the structure.

For this model, a literature survey delves into the efficacy of GRUs in text classification tasks, specifically focusing on their application to the Quora Question Pairs dataset. By examining recent research and comparative studies, this section aims to provide a comprehensive understanding of why GRUs are a preferred choice over other recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks for paraphrase identification.

Zulqarnain et al. (2019) highlight GRU's effectiveness in text classification tasks by showing it to perform better on benchmark datasets such as TREC and Google Snippets. The vanishing gradient problem is better managed by the GRU due to its architecture, which has fewer gates than the LSTM, which improves accuracy and lowers error rates. Moreover, by

including pre-trained word embeddings like GloVe, the GRU becomes more proficient at capturing contextual semantics between words, which makes it especially useful for comprehending and categorizing material according to its semantic content. In addition to outperforming conventional RNNs, the experimental results from Zulqarnain et al. (2019) confirm that the GRU model is a more computationally efficient option than LSTM, which makes it a perfect fit for paraphrase detection tasks are understanding subtle semantic

Ahmad et al. (2023) work in sentiment analysis task emphasizes the benefits of GRU over conventional Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. GRU's structure, which has fewer gates than LSTM, maintains great performance in capturing syntactic and semantic information while enabling faster training and less computing complexity. By adding word embeddings, part of speech (POS) tags, and contextual location information, the Attention-Based Multi-Channel GRU (Att-MC-GRU) substantially improves the model's performance and improves aspect extraction and sentiment classification accuracy. This hybrid technique focuses on the most important words in a sequence by utilizing the strengths of many input attributes and the attention mechanism to filter out extraneous information.

Ishmam and Sharmin’s (2019) work in hate speech domain provides evidence of the GRU model's advantages over conventional machine learning methods. The authors achieved a substantially higher accuracy than previous approaches, such Random Forest, which only attained 52.20%, by building a GRU-based deep neural network model for classifying user comments on Facebook pages. The GRU model demonstrated its capacity to grasp contextual information and subtle linguistic aspects by achieving 70.10% accuracy by utilizing word

embeddings such as Word2Vec. This is especially important for identifying paraphrases, when it's critical to grasp minute variations in meaning.

Putera Khano et al. (2023) distinguished the advantages of GRU over Long Short-Term Memory (LSTM) algorithms by emphasizing the former's ability to manage information flow and handle long-term dependencies without the need for more memory cells. Using two gates—the update gate and the reset gate to modify data flow and regulate how much historical information should be remembered or ignored, GRU streamlines the architecture of the RNN. With less computing complexity and faster training times, this simplified architecture retains outstanding performance in capturing contextual semantics and fine-grained linguistic nuances. The research also shows that GRU outperforms LSTM in many text classification tasks, indicating its applicability for tasks like paraphrase recognition where it is essential to comprehend subtle variations in meaning. The model is a reliable and effective option for natural language processing jobs since it can process vast amounts of textual input by utilizing pre-trained word embeddings and GRU's gating mechanisms.

Shiri et al. (2023) compared several deep learning models—such as CNN, RNN, LSTM, and GRU—across several data sets. Their investigation demonstrated GRU's superior performance, especially when dealing with big and unstructured datasets. Compared to LSTM, the GRU model had a simpler architecture with fewer gates, which allowed it to train more quickly and require less computing power while still achieving good accuracy. GRU proved its capacity to grasp temporal dependencies and contextual semantics by outperforming other models in their studies in terms of accuracy, precision, recall, and F1-score. These qualities are critical for identifying paraphrases, as it is necessary to recognize minute variations in meaning.

GRU is a good option for natural language processing tasks because of its efficiency and resilience, as demonstrated by the results of Shiri et al. (2023).

Thus, the literature review highlights the major benefits of the Gated Recurrent Unit (GRU) model for paraphrase detection tasks, especially when used with datasets such as the Quora Question Pairs. In comparison to other recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) models, studies like those by Shiri et al. (2023) and Zulqarnain et al. (2019) demonstrate GRU's superior performance in handling sequential data, capturing contextual semantics, and achieving high accuracy with reduced computational complexity. For natural language processing jobs that need accurate semantic understanding, the GRU is a very effective and appropriate option because of its effective design, quicker training times, and strong capacity to manage long-term dependencies.

Siamese Network with LSTM, GloVe and FastText embeddings

The modeling approach adopted here harnesses the power of Long Short-Term Memory (LSTM) networks using Siamese network architecture. LSTMs are good at handling tasks that require understanding long-term dependencies between words, which makes them an appropriate choice for the given word paraphrase identification task. Moreover, LSTMs deal with some common challenges faced when training neural networks like vanishing or exploding gradients thus ensuring a much more stable learning process (Shih et al., 2017). This project will investigate how these features of LSTM networks can enable reliable detection of semantically matching question pairs in the dataset.

The use of Siamese LSTM networks in semantic textual similarity and text categorization has recently been advanced. Attentive Siamese LSTM network is proposed by Bao et al., (2018).

As a result, this method enabled human beings to avoid manual feature engineering as its performance surpasses that of various corpora and languages. Mueller and Thyagarajan (2016) also came up with Siamese LSTM which incorporates synonymic information based on Manhattan distance metric for sentence similarity evaluation outperforming the current methods. In addition, Neculoiu et al. (2016) explored more on how Siamese recurrent networks work through text similarity when they tried to emphasize the words implying that their texts are more closely connected or not. Collective studies have indicated that Siamese Long Short-Term Memory architectures have potential for enhancing natural language processing tasks over conventional methods towards more sophisticatedly and semantically aware text analysis models.

Siamese Network with Transformer layers, GloVe and FastText embeddings

The Siamese Network architecture is used as the underlying architecture for all the different types of models built and trained in this project. In this paper, the Transformer model is selected as the building block for making the Siamese Network model to solve the Quora Question Pair detection problem. Since Transformer is introduced by Vaswani et al. (2017), it has revolutionized the field of Natural Language Processing, and has unlocked a lot of opportunities to solve many kinds of problems. It is well suited for the task of Quora Question Pair detection, and this section covers the theoretical background that motivated the choice to select Transformer and Siamese Neural Network architecture. Each model is trained with both GloVe and FastText embeddings to determine which vector embedding is well suited for the task of Quora Question Pair detection.

The concept of Siamese Neural Networks is introduced by Bromley et al. (1993) in paper titled "Signature Verification Using a 'Siamese' Time Delay Neural Network". It is an

architecture designed for tasks that involve finding similarities or relationships between two inputs, such as in face verification or signature recognition. This architecture consists of two identical subnetworks, each processing one of the input pairs. These subnetworks share the same parameters, ensuring that the same type of features is extracted from each input, thereby facilitating meaningful comparisons. The outputs of these subnetworks, often referred to as feature vectors, are then compared using a distance metric (like Euclidean distance or cosine similarity). The overall network is trained end-to-end in one shot. This training approach teaches the network to effectively map inputs that are semantically similar close to each other in the feature space, thus achieving the desired task of comparison or relationship detection.

The concept of the attention mechanism in neural networks is first introduced in the paper "Neural Machine Translation by Jointly Learning to Align and Translate" by Bahdanau et al (2014). This paper presented Attention as a solution to the problem of encoding the complete input sequence into a fixed-length vector in sequence-to-sequence learning, particularly in the context of machine translation. The attention mechanism allows the model to automatically learn to focus on different parts of the input sequence when predicting each word of the output sequence, essentially deciding dynamically which parts of the input are most relevant. Bahdanau et al. argued that using a fixed length context vector is inadequate for longer sentences and presented the attention mechanism as a solution to dynamically generate a context vector by focusing on relevant parts of the input sequence as needed.

The Bilateral Multi-Perspective Matching (BiMPM) model, introduced by Wang et al. (2017) is a technique designed for Natural Language sentence matching tasks, such as question answering, natural language inference, and paraphrase identification. Their model performed matching of two sentences in both directions – each sentence is matched against the other. This

bilateral approach helped capture the mutual influences and relationships between the sentences more effectively. Using a BiLSTM (Bidirectional Long Short-Term Memory) network, the model would generate context-aware representations for each word in the sentences. This is followed by a matching layer that performs the core task of matching words and phrases from one sentence to the other in various perspectives. The outputs from the matching layer are aggregated and synthesized to form a fixed-size representation that summarized the matchings. This layer is then connected to the final layer, which gave the result for the matching task, like classifying whether the sentences are equivalent, or determining the relationship between them. The BiMPM model showed strong performance across multiple tasks, reflecting its versatility and robustness in understanding complex sentence relationships.

The "Attention is All You Need" paper, introduced by Vaswani et al. (2017), presents the Transformer model, a novel architecture that has significantly impacted Natural Language Processing (NLP) and beyond. They introduced self-attention, a mechanism that allows the model to weigh the importance of different words in a sentence, regardless of their position. Their innovation, Transformer, uses a stack of self-attention and feed-forward layers for both encoder and decoder, eliminating the need for recurrence. The Transformer architecture is also highly parallelizable, which makes it more efficient than architectures based on convolutional or recurrent layers. The equation for Attention that they used in the Transformer can be formulated as shown in formula (1) below.

$$Attention(Q, K, V) = softmax(\frac{QK^t}{\sqrt{d_k}})V \quad (1)$$

Q stands for "Query" which are usually the representations of the input tokens that are transformed into a space where they can be compared against keys. **K** stands for "Key" which are also representations of the input tokens but are used to compute the amount of attention that

needs to be paid to each value. **V** stands for "Value" which are the representations of the input tokens that are combined based on the attention weights to produce the output. $\mathbf{QK^t}$ stands for dot product of **Q** and **K** and this operation is used to measure the similarity between all queries and all keys. $\mathbf{d_k}$ is the dimension of the key vectors and is used to scale the dot products. **Softmax** function is applied over each row of the scaled dot product matrix. This step converts the scores into probabilities that sum to one. The softmax essentially determines how much each part of the input should contribute to the output. A high score in the softmax output indicates that the corresponding value is important to consider for the given query.

Devlin et al. (2019) introduced BERT (Bidirectional Encoder Representations from Transformers) which further pushed the boundaries of Transformer to solve various kinds of problems. Unlike previous models which typically read text data in one direction (either left-to-right or right-to-left), BERT is designed to read the entire sequence of words at once. One of BERT's novel training approaches is the masked language model (MLM), where some percentages of the input tokens are randomly masked, and the goal for the model is to predict the original value of the masked words, based only on their context. This enabled the model to better understand the relationships and structures within sentences. The BERT model is first pre-trained on a large corpus of text in an unsupervised manner using MLM. Once pre-trained, BERT could be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering, sentiment analysis and paraphrase detection.

Model Supports

Environment, Platforms and Tools

Hardware/Software Requirements. The section on hardware requirements details the basic equipment settings of various models for tasks such as documentation management, data preprocessing and storage, model training and evaluation, and dataset storage.

Table 12

Hardware/ Software requirements, Configuration, and Goals

Hardware/Software	System Configuration	Goals
Local Machine	8-core CPU and GPU	Data preprocessing on local machine leveraging the machine's computational resources and maintaining control over data privacy and accessibility and backup storage.
Google Shared Drive	55MB Cloud Storage	Centralized and accessible repository for storing project dataset, facilitating seamless collaboration while providing secure storage infrastructure.
Google Colab	Python 3 Compute Resource, RAM 1.03/12.67 GB Disk 24.41/107.72, T4 GPU	Cloud based platform for streamlined model development, leveraging cloud-based resources for efficient training, hyperparameter tuning, and evaluation.

Google Shared Drive	~50MB Cloud Storage	Centralized and accessible platform for collaborative project document management.
Visual studio code	v1.89.1	For data cleaning and maintaining folder structure
Google Chrome	v124.0.6367.158	For research and accessing cloud services
Python	3.11.1	For data loading, model development and model training

Table 12 provides a system configuration of hardware and software that will be used during various project processes. Preprocessing data requires taping into much computation power from the local machine (8-core CPU and GPU, 16 GB RAM, Storage 512 GB) for purposes related to maintaining integrity in cases of data privacy, particularly while dealing with this kind of processing. Storing datasets on Google Shared Drive is helpful as it has enough capability to handle large volumes of information for collaborative task force. The GloVe Vectorizations file of size 1.04GB will also be stored on Google shared Drive for centralized access. Using Google Colab for model training and evaluation allows leveraging cloud-based resources that provide Python 3 compute resources with 1.03/12.67 GB RAM and 24.41/107.72 GB disk space, thus facilitating streamlined development and hyperparameter tuning (Google Colab, 2020). Furthermore, documentation management on Google Shared Drive provides a central platform where each member can access project documentation for collaboration among

peers. These specifications are relevant for all types of project activities aimed at achieving optimum performance and efficiency.

Tools. The list of tools used in this project is mentioned in Table 13.

Table 13

Tools and libraries used

Library		Method	Usage
NLTK	nltk.tokenize	word_tokenize	Tokenize the input sentence using NLTK's recommended word tokenizer
	nltk.corpus	stopwords	List of stopwords in the NLTK corpus
	nltk	download	Download the helper files for NLTK methods
Pandas	pandas	read_csv, concat, Series, Dataframe	Functions to read CSV files, combine them and model them as Dataframes and Series
NetworkX	networkx	Graph, connected_components, has_path	Create an undirected Graph using a list of vertices and edges, find connected components and identify paths
Numpy	numpy	asarray, load, expand_dims	Numpy functions to load, convert and operate on array

Library		Method	Usage
FastText	fasttext	load_model	Load a saved FastText embedding model to memory
	fasttext.util	reduce_model	Use PCA to reduce dimensions of 300-dimension FastText vector embeddings
MatplotLib	matplotlib.pyplot	subplots	Create a subplot to plot various visualizations
Seaborn	seaborn	lineplot	Create a line chart
Keras	keras	Model	Create a Keras model using an input and output layer object.
	keras.utils	plot_model, set_random_seed, pad_sequences, PyDataset	Keras utility function to plot model architecture, reproducibility and data processing
	keras.ops	shape, expand_dims, arrange, repeat	A numpy-compatible API to perform various operations on a Tensor during runtime
	keras.losses	BinaryCrossentropy	Use the cross-entropy loss to train a Keras model
	keras.regularizers	L2	Use L2 regularization for weights in a Keras layer

Library	Method	Usage
keras.initializers	RandomUniform, GlorotUniform	Use a specific distribution to sample the initial values for weights.
keras.layers	Layer, Embedding, Dense, Dropout, Subtract, MultiHeadAttention, LayerNormalization, GlobalAveragePooling, SimpleRNN, GRU, LSTM	Keras layers used as building blocks to build Siamese Network models
keras.callbacks	EarlyStopping, ModelCheckpoint, BackupAndRestore, CSVLogger	Callbacks to run at end of each epoch to persist and monitor Model training.

Model Architecture and Data Flow

Recurrent neural network (RNN)

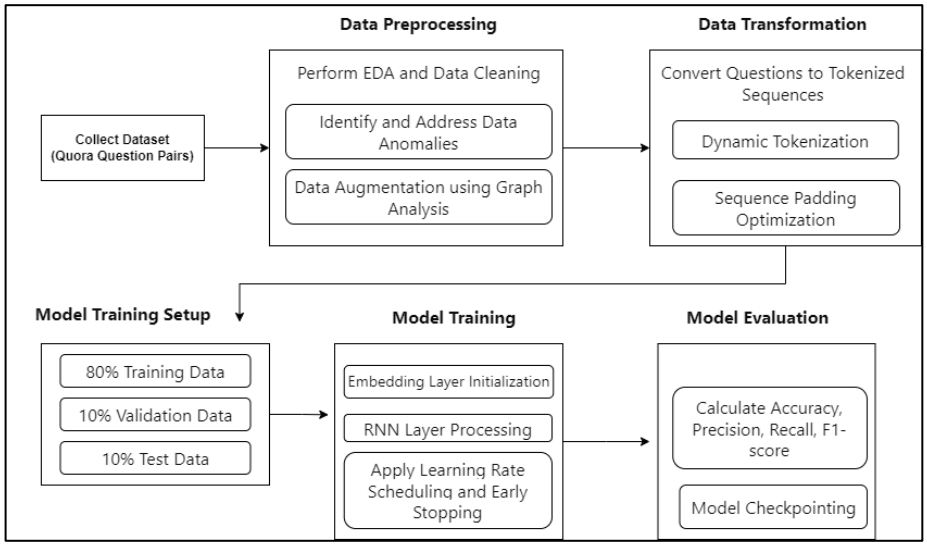
Figure 59 shows the Data flow diagram for RNN model. The dataset, comprising over 400,000 pairs of questions labeled for paraphrase identification, undergoes initial preprocessing with exploratory data analysis (EDA) to identify and address anomalies such as outliers, inconsistent data, and noise. Data augmentation, adding 91,227 duplicate question pairs through

graph analysis, balances class distribution. Dynamic tokenization methods are implemented to enhance model accuracy by effectively converting sentences into word tokens.

Feature extraction aims at changing text data into a form that is suitable for analysis and utilizes pre-trained embeddings like GloVe or FastText to convert tokenized sequences into embedded vectors. The Recurrent Neural Network (RNN) architecture was selected due to its efficiency in handling sequential data, it includes an embedding layer which takes the tokenized text, converting it into dense vectors followed by a RNN layer to process sequence data and catch temporal dependencies. Dense layers are then used on the output from the RNN layer for final classification. The division of the data into three groups ensures effective model training and evaluation; 80% goes to training, 10% for validation while the remaining 10% is reserved for testing. Learning rate scheduling, early stopping and model checkpointing were used while training to ensure convergence and avoid overfitting. Model performance was evaluated using metrics such as accuracy, precision, recall and F1-score across different models and embeddings to find out which model-embedding pair performed best. A flow diagram for this work shows how it went from data preprocessing all through model evaluation stages.

Figure 59

RNN data preprocessing



GRU (Gated Recurrent Unit)

Figure 60 shows the architecture diagram for GRU. The input tensors for the two questions to be compared are represented by the two input layers, question1 and question2, in the model architecture for paraphrase identification using Gated Recurrent Units (GRU) provided. It may be inferred from these input layers' (None, None) structure that they can handle multi-length sequences. An embedding layer, which uses fastText embeddings in particular, then receives the data from these input layers. As seen here (None, None, 100), where 100 is the dimensionality of the word embeddings, the embedding layer converts the input sequences into dense vectors of fixed size.

The embedded sequences are processed by a GRU layer (designated as gru_1) after the embedding layer. For the sequences to have contextual information and temporal dependencies, the GRU layer is essential. After receiving an input in the shape of (None, None, 100), this layer produces a tensor in the shape of (None, 128). The hidden state dimensions are represented by the 128 units in the GRU layer, which enables the model to gradually identify intricate patterns

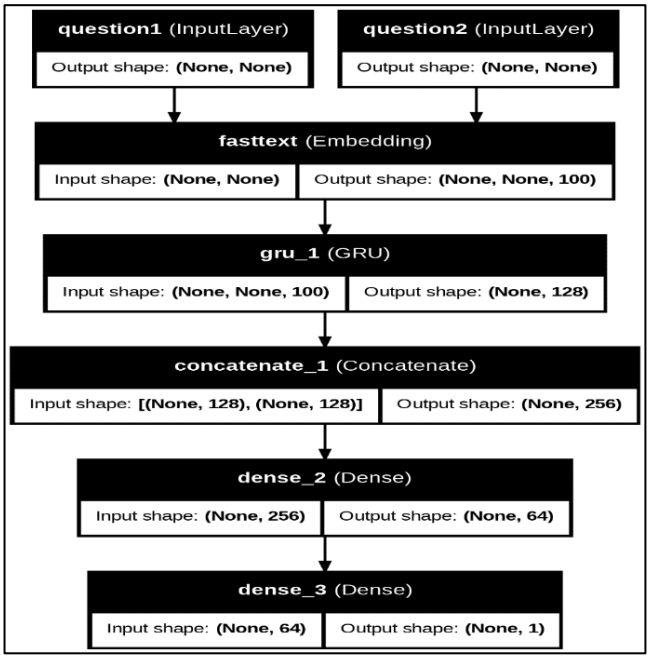
and relationships in the input sequences. With its gating mechanisms, GRUs effectively control information flow and preserve long-term dependence.

The outputs from gru_1, which correspond to questions 1 and 2, are concatenated after the sequences are processed with the GRU layer. The concatenation layer creates a single tensor of shape (None, 256) by joining the two output tensors, each of shape (None, 128). Concatenation is an important phase since it combines the data from the two queries, allowing the model to compare and comprehend the relationship between the two sequences. The last step in the classification process is passing this combined representation through more dense layers.

The output is concatenated and fed into two dense layers, one with 256 units (dense_2) and the other with 64 units (dense_3). To introduce non-linearity and help the network acquire the complex features required to discriminate between paraphrases, both dense layers apply a sequence of modifications. The solitary unit in the last dense layer, which produces an output between 0 and 1, has a sigmoid activation function. This result shows the likelihood that each of the two input questions is a paraphrase. By utilizing several thick layers, the model can improve its comprehension and generate predictions that are more precise by combining the information that is taken out of the GRU layer.

Figure 60

GRU Architecture diagram

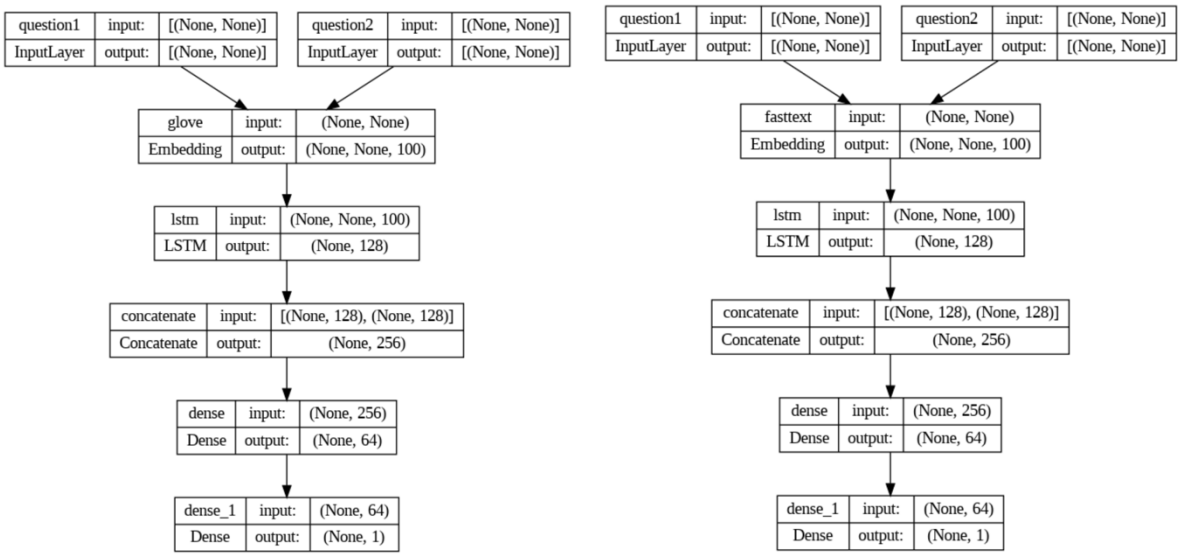


LSTM

Figure 61 depicts a Siamese LSTM network design intended to examine the semantic similarity of two text inputs using GloVe embeddings. Every input sequence is processed by similar subnetworks, which are organized in such a way that both inputs are encoded in a similar way for effective comparison. In the beginning, both question1 and question2 go through the Input layers that fit sequences of unspecified lengths. These inputs are then transformed from tokens to 100-dimensional vectors, which are the output of the GloVe embedding layer, that can capture the semantics in the vectors that are created by the pre-trained global vector representations.

Figure 61

Architecture of Siamese network with LSTM model



As a result, these embeddings are then used by the LSTM layers that output 128-dimensional vectors, which are designed to show the contextual relationships within the text sequences. The LSTM layers outputs from the two subnetworks are then combined into a one-dimensional vector of 256 dimension, which is a result of the features from both questions. This concatenated vector is fed to a dense layer which reduces its dimensionality to 64, after which it is fed to a final dense layer which gives a single value as the output. The above output shows the extent of semantic similarity between the two input questions, probably the scale of 0 to 1 is used to denote the dissimilarity and the perfection of similarity, respectively.

Transformer

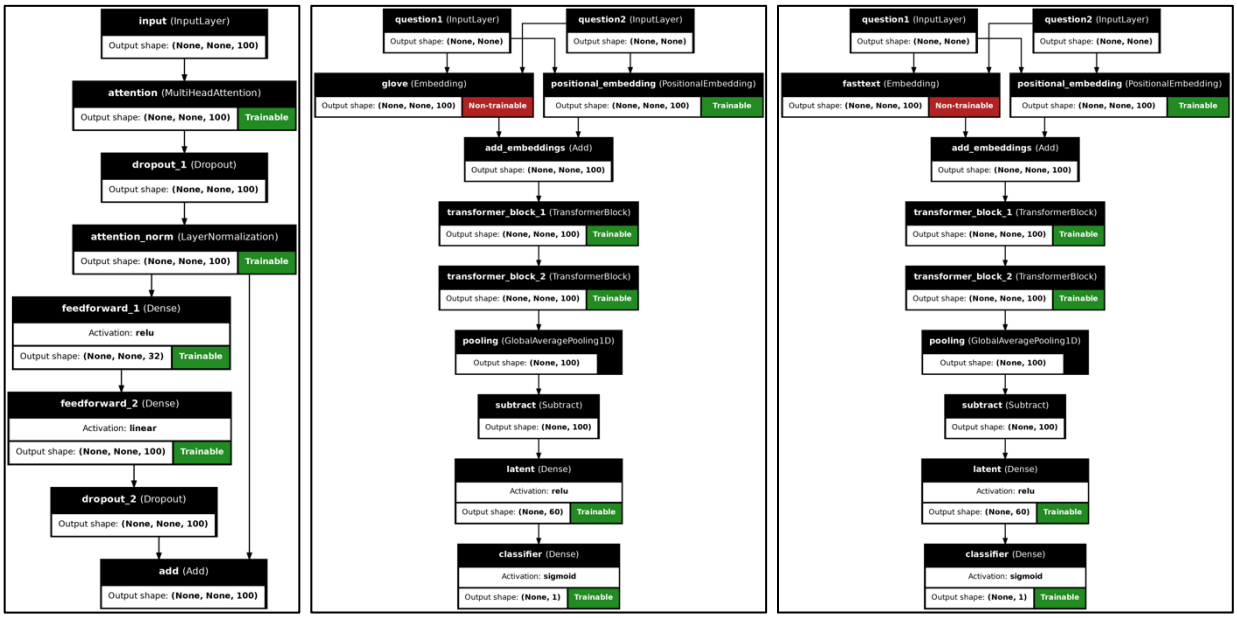
To visualize the model architecture and data flow as shown in Figure 62, plot_model utility provided by Keras is used. The same architecture is used to train two models – one with GloVe embeddings (shown in the middle of Figure 62) and one with FastText embeddings

(shown on the right of Figure 62). Both models follow the Siamese Network architecture and take as input 2 arrays of shape (batch size, sequence length), and output an array of shape (batch size, 1) containing the class probability for each sample in the batch. Batch Size and Sequence Length in the figures below are (None, None) since the Transformer model can technically take as input any size for each of these dimensions. In fact, the input sequences don't need to have the same sequence length. Question 1 can have the shape (32, 10) whereas question 2 can have the shape (32, 15). Some layers are marked as Trainable, which means the weights for these layers will be updated as part of the model training. Some layers are marked as non-trainable, which means that these layers have weights, but they are frozen and cannot be changed during model training. For other layers where this is not specified, it means that those layers don't have any weights associated with them, and only apply transformations or perform operations on the incoming data. Each layer also specifies the output shape of data that passes through that layer.

Each TransformerBlock, as shown in left of Figure 62, consists of a self-Attention Layer, whose parameters are trainable. This is followed by a Dropout layer with dropout rate of 0.1 and acts as a regularization mechanism. This is followed a LayerNormalization layer that helps in stabilizing the training of Attention layer before it. Then the architecture split into two branches. The first branch goes through an additional feed-forward Neural Network consisting of two fully connected layers. The first layer has a ReLu activation to enforce non-linearity, whereas the second layer has a linear activation. This branch then gets added back to the other branch and the output is passed to the next layer. This split branch architecture is also known as Residual network and is used to overcome the Vanishing Gradient issue in Deep Neural Networks. On the left is the architecture for TransformerBlock layer, in the middle is the architecture for GloVe mode, and on the right is the architecture for FastText model.

Figure 62

Overall architecture of Siamese Network with Transformer.



Both glove and fasttext layers contain the GloVe and FastText embeddings and are non-trainable. PositionalEmbedding layer, named as positional_embedding, are the positional embeddings that are learned during model training. TransformerBlock, described previously, is used in transformer_block_1 and transformer_block_2 layers. These two TransformerBlock layers are stacked one after the other. This allows the model to learn even more complex relationship between the input sequences and achieve higher accuracy. The number of TransformerBlock layers needed to be carefully considered, since stacking many such layers would lead to overfitting on the Train dataset.

Model Comparison and Justification

As far as image data go, RNNs are great at taking care of spatial patterns; however, they do not perform well with sequential, context dependent text data (Pascanu et al., 2013). Furthermore, there are challenges that come with working with RNNs such as vanishing and

exploding gradients, but LSTMs were designed to address these problems. Table 14 shows the comparison of proposed models based on several factors.

On the other hand, Transformers can process the sequences in parallel rather than sequential. As a result, it increases the rate at which models are trained by allowing for model operations to be run on multiple computational units at a time. GRUs are like LSTMs in that they address the vanishing gradient problem found in regular RNNs. They have these properties because they can keep track of dependencies longer than traditional RNNs and so may find them helpful whenever understanding context from larger text segments is important.

Table 14

Comparison of Models

Characteristics	**LSTM**	**Transformers**	**GRU**	**RNN**
Architecture	Parallel processing	Parallel processing	Parallel processing	Parallel processing
Data type	Can handle text, image, audio, and time series data	Highly effective on textual data	Can handle text, image, audio, and time series data	Can handle text, image, audio, and time series data

Characteristics	LSTM	Transformers	GRU	RNN
Data size	Performs well with large datasets but may not perform good with sparse data	Performs well with huge datasets	Performs well with large datasets but may not perform good with sparse data	Less effective with large datasets
Overfitting/ Underfitting	Can overfit on small dataset	Prone to overfitting unless trained on large datasets	Can overfit on small dataset	Prone to both overfitting and underfitting
Model complexity	Highly complexity	High Complexity	Lower complexity than LSTM	Low Complexity
Preprocessing Requirements	Require sequence padding, normalization, and embedding layers for text.	Requires positional encoding	Require sequence padding, normalization, and embedding layers for text.	Require sequence padding, normalization, and embedding layers for text.

Characteristics	LSTM	Transformers	GRU	RNN
Training time	Slower training because of sequential processing	Faster training on GPUs	Slower training because of sequential processing	Faster training
Computational complexity	High Computational complexity	Very high computational and space complexity	High Computational complexity	High Computational complexity
Strengths	Good at capturing dependencies on sequential data	Good in understanding of global dependencies	Efficient for moderate size dataset	Simple and effective for short sequences
Limitations	Complex in training	Requires a huge amount of data for training	Overfits on small datasets	It has a limited memory, and it underperforms

Model Evaluation Methods

Confusion Matrix

In binary classification, the outcomes can either be positive or negative, and predictions can be true or false. A confusion matrix is a table that allows visualization and evaluation of the performance of a classification model. (Ting , 2011).

TP (True Positives). These are the instances correctly predicted as positive.

TN (True Negatives). These are the instances correctly predicted as negative.

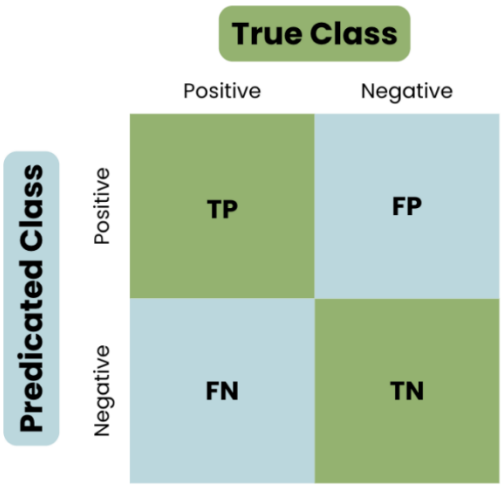
FP (False Positives). These are the instances incorrectly predicted as positive (also called Type I error).

FN (False Negatives). These are the instances incorrectly predicted as negative (also called Type II error).

The matrix is typically arranged as in figure 63.

Figure 63

Confusion Matrix



The confusion matrix provides a detailed analysis of the model performance, distinguishing between different types of errors and helps in understanding the model's behavior across different classes. At the same time, it is not very intuitive for datasets with an imbalanced class distribution; the matrix might be dominated by the majority class. It also does not provide insights into the costs of different types of errors, which could be crucial in some applications.

Accuracy

Accuracy is a metric used to evaluate classification models, specifically in binary classification problems where the outcomes are restricted to two classes (e.g., positive, and negative). It measures the proportion of correct predictions made by the model over the total number of predictions made. (Metz, October 1978). Accuracy can be calculated using the following formula (2).

$$Accuracy = \frac{Number\ of\ correct\ predictions}{\text{Total number of predictions}} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad (2)$$

The Advantages of using Accuracy is given below.

Simplicity. Accuracy is straightforward to understand and calculate. It provides a quick measure of how well a model is performing.

Effectiveness in Balanced Datasets. When the classes are balanced, accuracy can be a reliable measure of model performance.

The shortcomings of Accuracy are given below.

Misleading in Imbalanced Datasets. In cases where there is a significant class imbalance, accuracy can be misleading. For example, if 95% of the data belongs to one class, a model that naively predicts this major class for all inputs would still achieve a 95% accuracy rate, despite not learning anything useful about the minority class.

Doesn't Account for the Costs of Different Types of Errors. Accuracy treats all types of errors the same way. However, in many real-world applications, the cost of a false positive might be very different from the cost of a false negative.

Precision

Precision is a performance metric used in binary classification to evaluate the accuracy of the positive predictions made by a model. It answers the question: "Of all the instances labeled as positive by the model, how many are actually positive?" Precision is particularly important when the cost of a false positive is high. (Taylor, 1999). Mathematically, it is expressed as below formula (3)

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \tag{3}$$

The Advantages of using Precision are given below.

Focus on Positive Class. Precision is critical in scenarios where the cost of a false positive is high, such as in email spam detection, where a legitimate email classified as spam can be problematic.

Useful in Imbalanced Datasets. Precision can be a useful metric in datasets where the positive class is less frequent but more important to predict correctly.

The shortcomings of Precision are given below.

Does Not Consider False Negatives. Precision does not consider false negatives (FN), i.e., actual positives that the model incorrectly labels as negative. This can be a problem in critical applications like disease screening.

Not a Complete Measure. Precision alone does not provide a complete picture of a model’s performance. It must be considered alongside other metrics like recall and accuracy.

Recall

Recall is a crucial metric used in the evaluation of binary classification models. It provides insight into the model's ability to identify all relevant instances from the dataset. Recall, also known as Sensitivity or True Positive Rate, measures the proportion of actual positives correctly identified by the model. (Boutell, Luo, Shen, & Brown, 2004) The formula for recall is given in formula (4).

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{4}$$

The Advantages of using Recall is given below.

Focus on Positives. Recall is particularly valuable in contexts where the cost of missing a positive instance (e.g., not diagnosing a disease) is much higher than the cost of a false positive.

Useful in Imbalanced Datasets. It helps to evaluate the model's performance in scenarios where one class (positive) is much less prevalent than the other.

The shortcomings of Recall are given below.

Does Not Consider False Positives. High recall can sometimes be achieved at the expense of accepting many false positives, which can be problematic in some scenarios (like spam detection).

Not a Standalone Metric. Recall needs to be balanced with precision (which measures the accuracy of the positive predictions) for a comprehensive evaluation. The trade-off between recall and precision is often visualized through the Precision-Recall curve.

F1 Score

The F1 score is a widely used metric to evaluate the performance of binary classification models. It considers both the precision and the recall of the test to provide a single score. It is the

harmonic mean of precision and recall. A higher F1 score is generally better, indicating both high precision and high recall. Mathematically, it can be represented as shown in formula (5).

$$F1 = 2 \text{ x } \frac{\text{Precision x Recall}}{\text{Precision+Recall}} = \frac{2\text{TP}}{2\text{TP+FP+FN}} \quad (5)$$

The Advantages of using F1 is given below.

Balance Between Precision and Recall. F1 Score provides a balance between precision and recall, which is particularly useful when the class distribution is uneven.

Useful in Unequal Class Distribution. F1 Score is more informative than accuracy when there are unequal numbers of observations in each class, which is common in real-world data.

The shortcomings of F1 are given below.

Not Intuitive Understanding. the F1 Score and how it is derived can be less intuitive than metrics like accuracy.

No Insight into Specific Errors. While F1 Score gives a single measure of performance, it doesn't tell you about the types of errors (like false positives vs. false negatives).

Assumes Equal Importance of Precision and Recall. The regular F1 Score weights precision and recall equally, which may not align with specific business needs where one might be more important than the other.

Receiver Operating Characteristic and Area Under Curve

The ROC curve (Receiver Operating Characteristic curve) and the Area Under the ROC Curve (AUC) are important tools used in evaluating the performance of binary classification models. The ROC curve is a graphical representation of a classifier's performance across all classification thresholds. It plots two parameters.

True Positive Rate (TPR). Also known as sensitivity or recall, it measures the proportion of actual positives correctly identified by the model having formula (6).

False Positive Rate (FPR). Measures the proportion of actual negatives incorrectly classified as positives with formula (7).

$$TPR = \frac{\text{TP}}{\text{TP+FN}} \tag{6}$$

$$FPR = \frac{\text{FP}}{\text{TN+FP}} \tag{7}$$

Graphically, this curve is plotted with FPR on the X-axis and TPR on the Y-axis. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold.

The AUC measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1). It provides an aggregate measure of performance across all possible classification thresholds. The AUC can be computed through various numerical integration techniques, often based on the trapezoidal rule given in formula (8)

$$AUC = \int_0^1 ROC(t)dt \tag{8}$$

AUC can be interpreted as AUC = 1 The model has perfect discrimination. 0.5 < AUC < 1 The model is better than random guessing. Higher values indicate better performance. AUC = 0.5 The model has no discriminative ability, equivalent to random guessing. AUC < 0.5 The model performs worse than random guessing.

Precision Recall Curve and Area under Precision Recall Curve

The Precision-Recall curve is a graph with Precision values on the y-axis and Recall values on the x-axis. The threshold of classifier is varied from 0 to 1, and for each threshold, a precision and recall value are calculated. The curve plots these values, starting from the threshold closest to zero to the threshold closest to one.

The Area Under the Precision-Recall Curve (AUPRC) represents a single summary statistic of the average precision across all recall levels, and it provides a measure of the overall quality of a binary classifier. It is particularly useful when dealing with imbalanced datasets, where the minority class (positive class) is of greater interest. The AUPRC can be calculated by integrating the area under the curve formed by plotting precision against recall. It does not have a simple analytical formula and is typically calculated numerically, often through a trapezoidal integration method in practice.

Model Validation and Evaluation

All the models are evaluated on the validation set first and the best of all the models is then trained on train plus validation data set and tested the results on test dataset.

Recurrent neural network (RNN)

The model is trained on training and validation datasets using both FastText and GloVe embeddings. The performance metrics for two LSTM models FastText and GloVe that were trained using various word embeddings are displayed below in Figure 64. Several metrics, including F1 Score, ROC AUC, PR AUC, Accuracy, Precision, and Recall, were used to evaluate these models' performance.

As shown in Figure 64, The Glove model outperforms the other one in terms of Accuracy, Precision, F1 Score, ROC AUC, and PR AUC, but there is not much of a difference in performance between the two models across these measures. This finding leads one to the conclusion that, although both embedding approaches might be useful sources of features for RNN -based text classification, FastText embeddings might be able to capture more useful semantic information required for this task, which would explain their marginally superior

performance. This could be explained by the subword data that FastText collected, which might help it cope with OOV words and uncommon keywords more effectively than GloVe does.

Figure 64

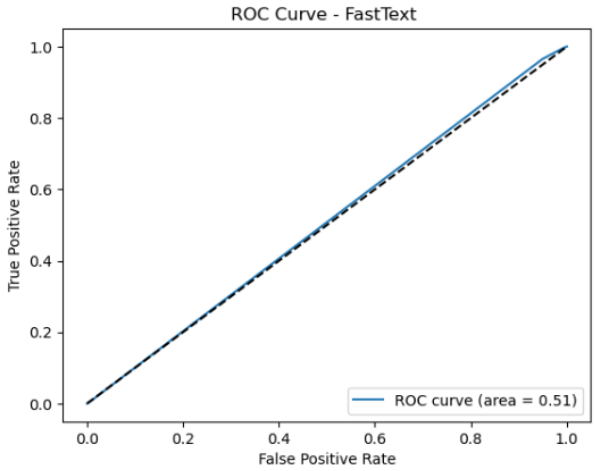
Comparison of metrices for models

Model	Accuracy	Precision	Recall	F1 - score	Area Under ROC curve	Area Under Precision-Recall Curve
RNN (GloVe)	0.6240	0.4406	0.4303	0.4354	0.5683	0.5106
RNN (Fasttext)	0.6249	0.3761	0.5331	0.5415	0.5075	0.3584

The ROC curve for the FastText model is very close to the diagonal line, which represents a random guess. An AUC of 0.51 indicates that the model performs slightly better than random guessing but is not a strong classifier. The true positive rate increases linearly with the false positive rate, suggesting limited discrimination between the positive and negative classes as shown in Figure 65.

Figure 65

ROC Curve for SimpleRNN using Fasttext embeddings

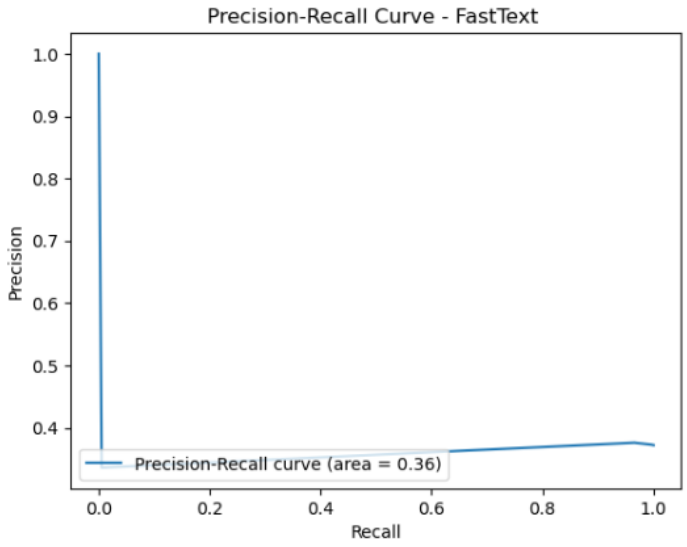


The Precision-Recall curve for FastText starts with high precision but rapidly declines as recall increases, which indicates a significant number of false positives as the model attempts to

increase its true positive rate. The low AUC of 0.36 highlights the model's difficulty in effectively distinguishing between classes, especially when trying to maximize recall as shown in Figure 66

Figure 66

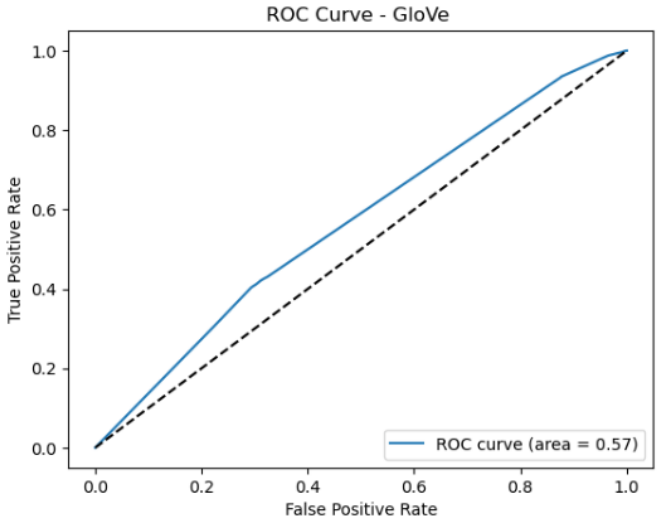
Precision Recall Curve for SimpleRNN using Fasttext embeddings



Similar to the FastText model, the GloVe model's ROC curve is also close to the diagonal, albeit slightly higher, which reflects marginally better performance. An AUC of 0.57 still indicates relatively poor performance, though it suggests that the GloVe embedding provides slightly more effective features for classification compared to FastText in this model configuration as shown in Figure 67

Figure 67

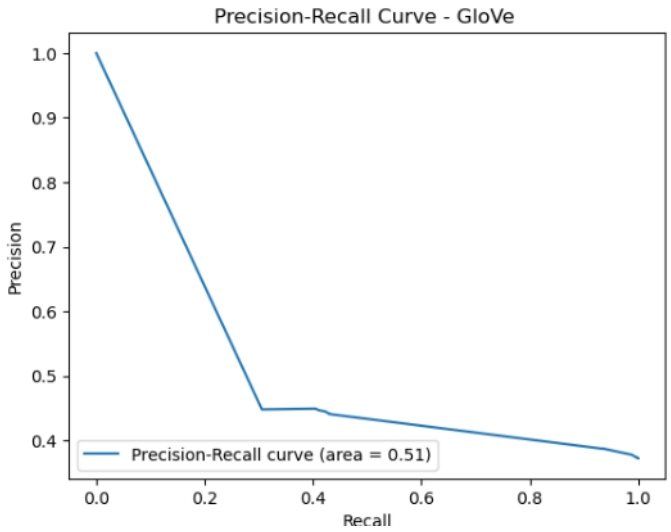
ROC Curve for SimpleRNN using Glove embeddings



The Precision-Recall curve for GloVe shows a better balance between precision and recall compared to FastText. Starting with high precision that decreases as recall increases, the curve maintains a moderate level of precision over a range of recall values. An AUC of 0.51, while not high, is more favorable than the FastText model, indicating GloVe embeddings might be more effective for this task as shown in Figure 68

Figure 68

Precision Recall Curve for SimpleRNN using Glove embeddings



Gated Recurrent Unit

The preprocessed Quora question pair dataset is split into ratios of 80:10:10 as training, validation, and testing dataset during the data preparation step. After developing the GRU model using the training dataset the model is evaluated using the testing dataset on the evaluation metric described in the previous section. GloVe and fastText embeddings are used after tokenization of questions to note paraphrasing identification. The model classifies if the test question pairs are semantically similar. This comprehensive evaluation strategy enables comprehensive assessment of the model's performance under varied situations and improvements.

GRU using GloVe Embeddings. Figure 69 shows that the model achieved an accuracy of 82.29%, indicating that most of its predictions were correct. Precision is 75.93%, meaning that when the model predicted a duplicate, it is accurate around 76% of the time. The recall rate stood at 76.79%, showing that the model successfully identified nearly 77% of actual duplicate pairs. The F1 Score, which balances precision and recall, is 76.36%, highlighting the model's effectiveness in both identifying and correctly predicting paraphrased pairs.

The ROC AUC, a measure of the model's ability to distinguish between classes, is 0.89. This high value reflects strong performance in differentiating between duplicate and non-duplicate pairs. The PR AUC, which measures precision and recall trade-offs, is 0.84, further demonstrating the model's robustness.

Figure 69

Classification Report for selected model

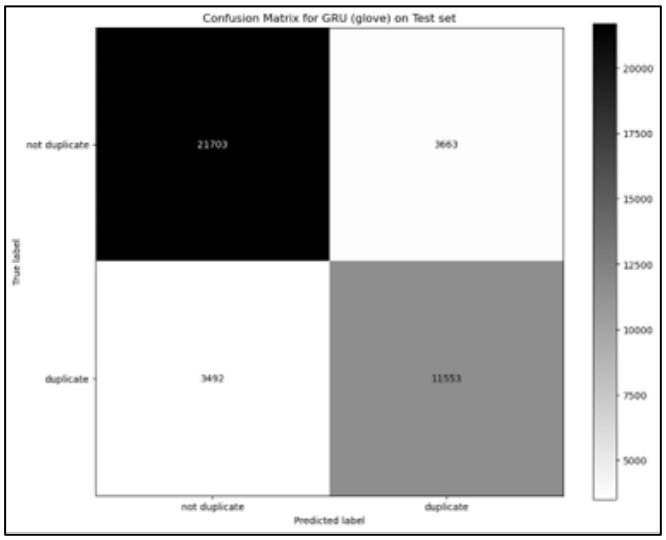
	Accuracy	Precision	Recall	F1	ROC AUC	PR AUC
GRU (glove)	0.822944	0.759267	0.767896	0.763557	0.891904	0.83806

A thorough understanding of the model's performance on the test set of the Quora Question Pairs dataset may be obtained from the confusion matrix shown in Figure 70 for the GRU model utilizing GloVe embeddings. The model properly recognized 11,553 duplicate question pairs as duplicates and 21,703 non-duplicate question pairs as non-duplicates in this matrix. It did, however, miss 3,492 duplicate pairings and classified them as non-duplicates while mistakenly labeling 3,663 non-duplicate pairs as duplicates. When assessing the accuracy and error rates of the model, these findings are important. A high ratio of true positives to true negatives suggests that the model can effectively discriminate between pairs that are and are not duplicates. On the other hand, false positives and false negatives indicate areas in need of

additional work.

Figure 70

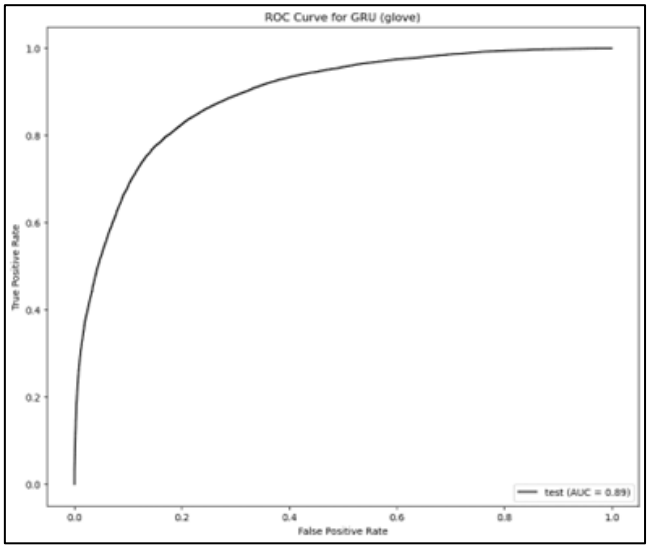
Confusion matrix for selected model



While assessing the performance of Binary classification algorithms ROC AUC curve is an important indicator shown in Figure 6. It explains the True Positive Rate (Sensitivity) versus the False Positive Rate (Specificity). Figure 71 helps to clarify the trade-offs between sensitivity and specificity. The Quora Question Pairs dataset's ROC curve illustrates how well the GRU model with GloVe embeddings can distinguish between duplicate and non-duplicate question pairs. The model is effective at detecting actual duplicates while reducing false positives, as evidenced by the excellent accuracy of the AUC (Area Under the Curve) value of 0.89. The curve's closeness to the top-left corner indicates that the model strikes a reasonable compromise between preventing inaccurate duplication identification and successfully identifying duplicates.

Figure 71

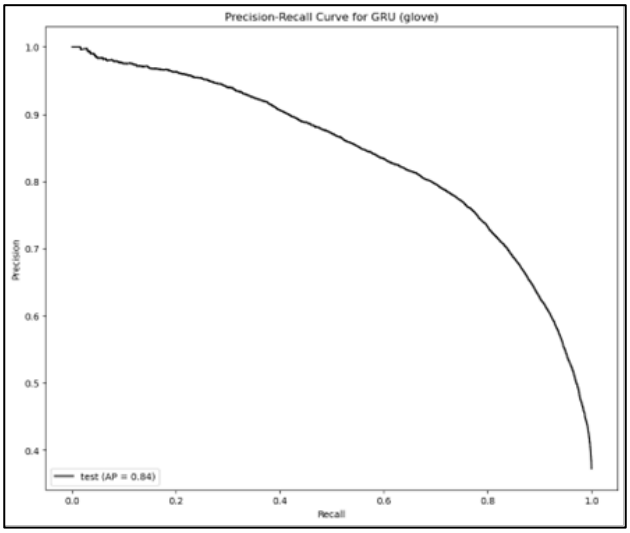
ROC Curve for selected model GRU using GloVe embeddings



When assessing binary classification models, the Precision-Recall (PR) curve is essential, particularly when dealing with imbalanced datasets. As shown in Figure 72 plotting accuracy against recall allows one to see how these measurements are traded off. The GRU model's PR curve, which displays the model's ability to balance recall and precision when identifying duplicate question pairs from the Quora Question Pairs dataset, is based on GloVe embeddings. As recollection rises, the precision and recall curve progressively decrease, as is common when recall is higher and lower thresholds imply lesser precision. With an Average Precision (AP) score of 0.84, the model can effectively balance recall and precision over a range of thresholds.

Figure 72

Precision Recall Curve for selected model GRU using GloVe embeddings



GRU using fastText Embeddings. Figure 73 summarizes the evaluation metrics for the GRU model with FastText embeddings. The model had an accuracy of 0.822004, indicating that 82.20% of its predictions were correct. Precision is 0.756971, which means that the model accurately predicted a duplicate 75.69% of the time. The recall is 0.768694, which means that the model correctly identified 76.86% of genuine duplicates. The F1 Score, which considers precision and recall, is 0.762787, showing a reasonable balance between these two metrics. The ROC AUC is 0.894752, suggesting the model's excellent discriminative ability. The PR AUC is 0.839048, indicating the model's durability, notably in discriminating between classes.

Figure 73

Classification Report for GRU (fastText)

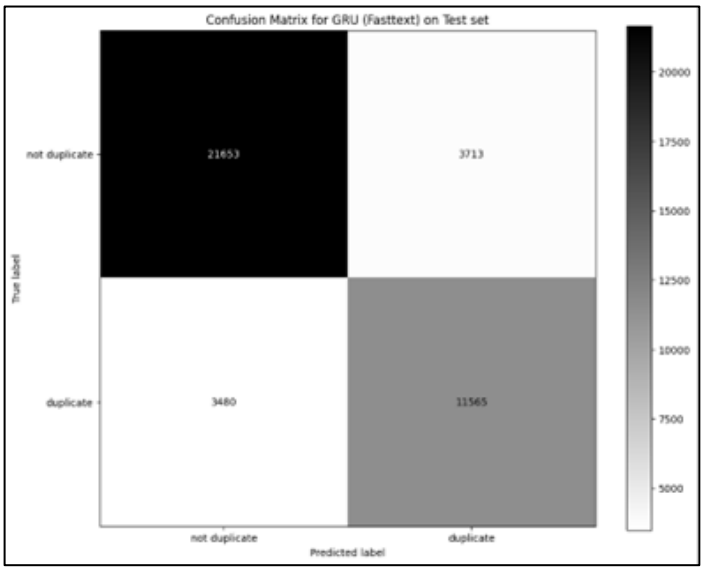
	Accuracy	Precision	Recall	F1	ROC AUC	PR AUC
GRU (Fasttext)	0.822004	0.756971	0.768694	0.762787	0.894752	0.839048

The confusion matrix in Figure 74 provides a thorough picture of model's strengths and limitations in Figure 73. The matrix breaks down the model's performance by indicating the number of True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN). For the validation set, the model correctly identified 21,653 non-duplicate pairs (TN) and 11,565 duplicate pairs (TP). It incorrectly classified 3,713 non-duplicate pairs as duplicates (FP) and 3,480 duplicate pairs as non-duplicates (FN).

The model's ability to recognize paraphrased question pairings is demonstrated by the large number of true positives and true negatives, while the false positives and false negatives offer information about possible areas for further optimization.

Figure 74

Confusion matrix for selected model GRU using fastText Embeddings



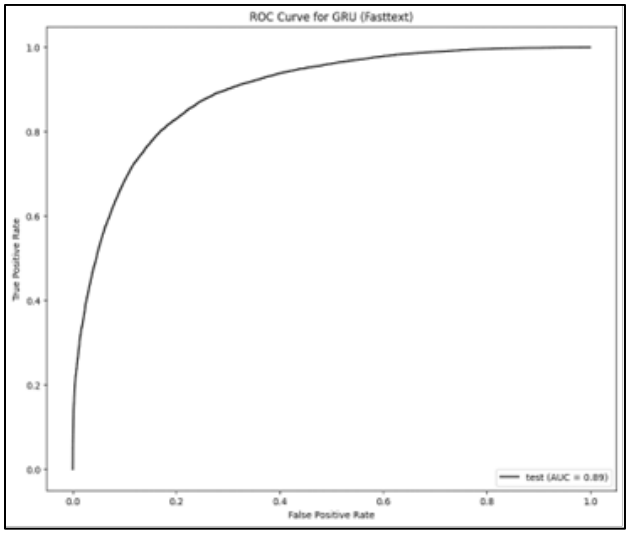
The diagnostic capacity of a binary classifier when its discrimination threshold is adjusted is depicted graphically by the ROC (Receiver Operating Characteristic) curve. Plotting the real positive rate (sensitivity) versus the false positive rate (specificity) at different threshold levels is what the curve represents. An area under the curve (AUC) of 1.0 indicates that a model

with perfect discrimination has a point in the upper left corner (100% sensitivity and 0% false positive rate). The ability of the GRU model to differentiate between duplicate and non-duplicate question pairs in the Quora Question Pairs dataset is shown by the ROC curve for the model employing FastText embeddings as shown in Figure 75.

The model shows a high degree of accuracy in predicting whether pairs of questions are paraphrased, as evidenced by the AUC value of 0.89. The model strikes a fair mix between sensitivity and specificity, as indicated by the curve's closeness to the top-left corner, which makes it a useful tool for paraphrase detection in this situation.

Figure 75

ROC Curve for selected model GRU using fastText Embeddings.

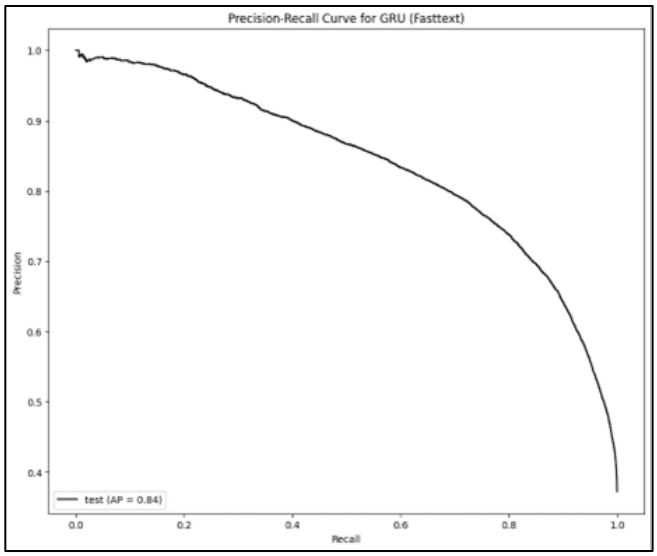


When assessing binary classification models, particularly when dealing with imbalanced datasets, the Precision-Recall (PR) curve is essential. Plotting precision versus recall helps to highlight the trade-offs between these measurements as shown in Figure 76. The GRU model's PR curve with GloVe embeddings illustrates how well the model balances recall and precision when detecting duplicate question pairs in the Quora Question Pairs dataset. Since higher recall frequently entails lower thresholds, which reduce accuracy, the curve typically starts out with

high precision and recall and gradually declines as recall increases. The model exhibits a fair balance between precision and recall across various thresholds, as indicated by the Average Precision (AP) score of 0.84.

Figure 76

Precision- Recall Curve for selected model GRU using fastText Embeddings.



LSTM

Using both FastText and GloVe embeddings, the model is trained and evaluated on training and validation dataset. Below Figure 77 shows the performance metrics for two LSTM models that were trained using different word embeddings which are FastText and GloVe. These models' performance was assessed using several metrics including Accuracy, Precision, Recall, F1 Score, ROC AUC and PR AUC.

There is not much difference in the performance of both models across these metrics, with FastText model being slightly better than the other one in terms of its Accuracy, Precision, F1 Score, ROC AUC as well as PR AUC. As a result of this finding, it is possible to conclude that while both embedding techniques could be good sources of features for LSTM based text classification but perhaps FastText embeddings manage to capture more helpful semantic

information necessary for this specific task thereby allowing them to perform slightly better than their counterparts. This may be attributed to the sub word data captured by FastText that may aid in dealing with OOV words and rare terms better than GloVe does.

Figure 77

Comparison of different metrices for model evaluation

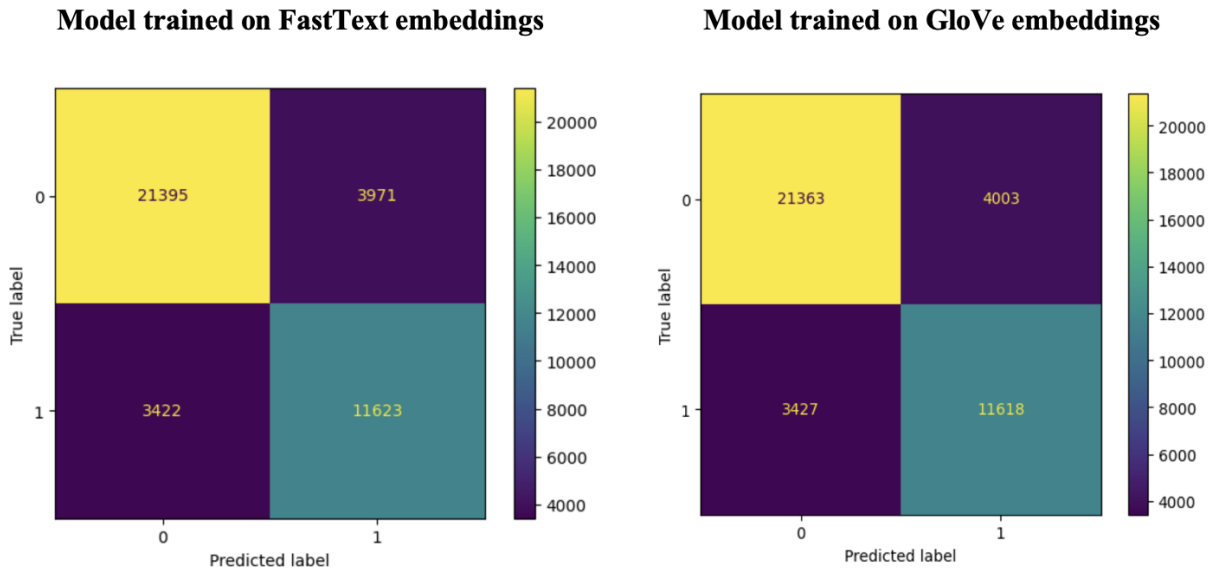
	Accuracy	Precision	Recall	F1	ROC AUC	PR AUC
LSTM (Fasttext)	0.817055	0.745351	0.772549	0.758706	0.890606	0.834036

	Accuracy	Precision	Recall	F1	ROC AUC	PR AUC
LSTM (GloVe)	0.816139	0.743742	0.772217	0.757712	0.883401	0.829656

Confusion Matrix. For understanding the prediction by model on test data, the confusion matrix is plotted for the models trained with both FastText and GloVe embeddings. Figure 78 shows the comparison of the confusion matrix for both models.

Figure 78

Confusion Matrix for LSTM model on FastText and GloVe embeddings



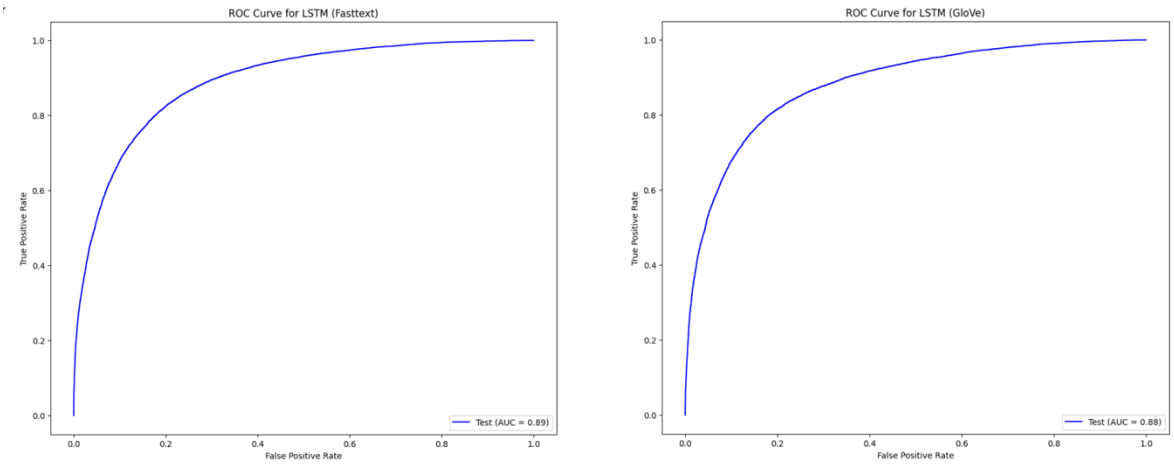
LSTM models on FastText and GloVe embeddings are almost on the same performance with minor differences in the number of false positives and true positives. The FastText model has slightly less false positives and more true positives than the GloVe model, hence it slightly outperforms the GloVe model in the field of reducing false positives and capturing more true positives. This kind of imaging makes the understanding of the trade-offs between different types of errors easier and can help the programmers to tune the models in further to improve their predictive accuracy.

Area under the Curve (AUC) of the Receiver Operating Characteristic (ROC). In Figure 79, the ROC curve for the model trained with FastText embeddings is shown on the right, and the model gets an AUC score of 0.89. This very high AUC value is proof of the excellent

performance of the model, where the classes are clearly separated with a high true positive rate and a low false positive rate as the threshold is changed.

Figure 79

ROC curve for LSTM Model on FastText and GloVe embeddings



In Figure 79, the ROC curve for the model trained with FastText embeddings is shown on the right, and the model gets an AUC score of 0.89. This very high AUC value is proof of the excellent performance of the model, where the classes are clearly separated with a high true positive rate and a low false positive rate as the threshold is changed.

The ROC curve for the model which was trained with GloVe embeddings is on the right, with an AUC of 0.88. This is a bit less than the FastText model but still shows a strong ability to classify the posts into the correct ones.

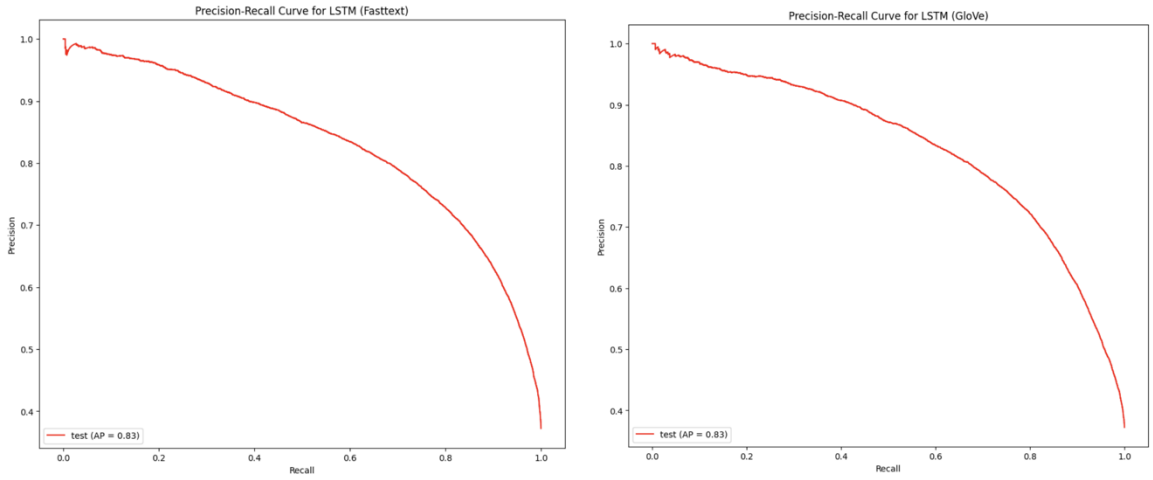
Both curves are described by a fast rise in TPR with a low growth of FPR, especially at the lower threshold settings. This feature demonstrates that both models, no matter what the embedding is used, can classify the objects with high accuracy without a high number of false positives, therefore they are good for tasks that require high discrimination sensitivity such as the semantic analysis of text. The minor variance in the AUC scores of the two may be caused by the

unique characteristics of the embeddings and how they manage to extract semantic subtleties from the training data.

Area under the Curve (AUC) of the Precision-Recall Curve. In Figure 80, Both models attain precisely the same area under the PR curve (AP score), which is 0.83. Thus, this resemblance shows that both embedding techniques achieve the same performance in the semantic features detection necessary for the LSTM to classify the tasks effectively. The PR curves are handy for showing the model's performance in situations where the positive class is more interesting than the negative class and where false negatives and false positives have different costs.

Figure 80

Precision-Recall curve for LSTM Model on FastText and GloVe embeddings



The quite simple way of the gradual decrease from high precision at low recall to low precision at high recall shows that the models keep the balance between the catching as many positives as possible and maintaining the accuracy, which is the most important in many practical applications such as information retrieval and ranking tasks.

Transformer (fasttext)

Figure 81

Training Curves

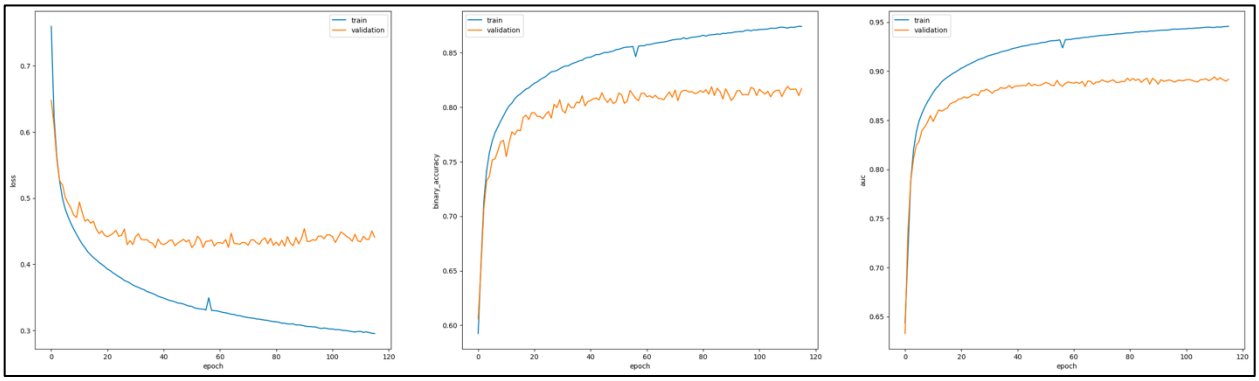
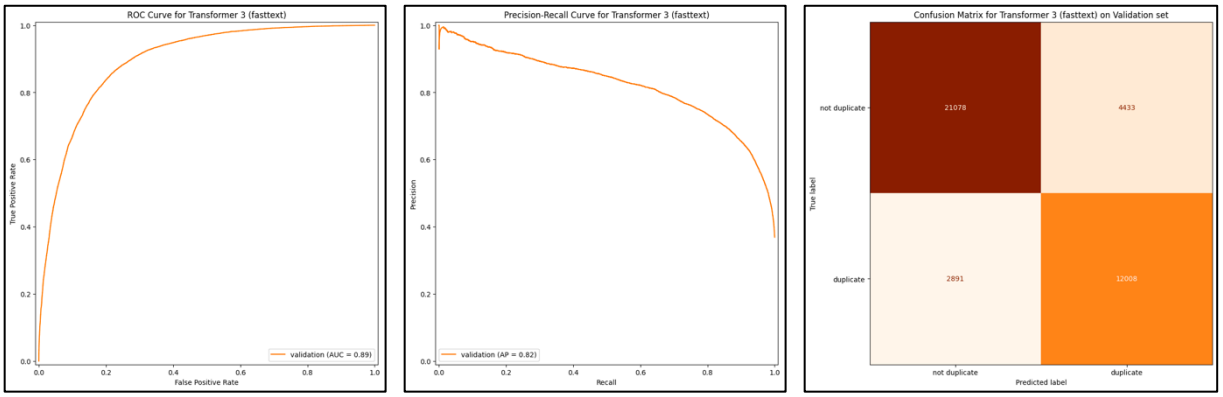


Figure 82

ROC Curve, Precision-Recall Curve and Confusion Matrix



From the training curves in Figure 82, the validation metrics started drifting from the training metrics as soon as epoch 5. The gap persisted till the time the training is stopped at epoch 100. The validation loss is almost flat, and the binary accuracy and AUC score for the validation set are still improving, but very slowly. Further training time might have improved the validation results, but at the risk of overfitting on the training dataset. The ROC Curve and Precision-Recall curve in Figure 82 have a much better shape compared to the previous models, and there are roughly 11% false positives and 7% false negatives, which are better than all the

models seen above. This model got an accuracy of 0.818758, precision of 0.730369, recall of 0.805960, F1 score of 0.766305, AUC of 0.894611 and AUPRC of 0.821879.

Transformer (GloVe)

Figure 83

Training Curves

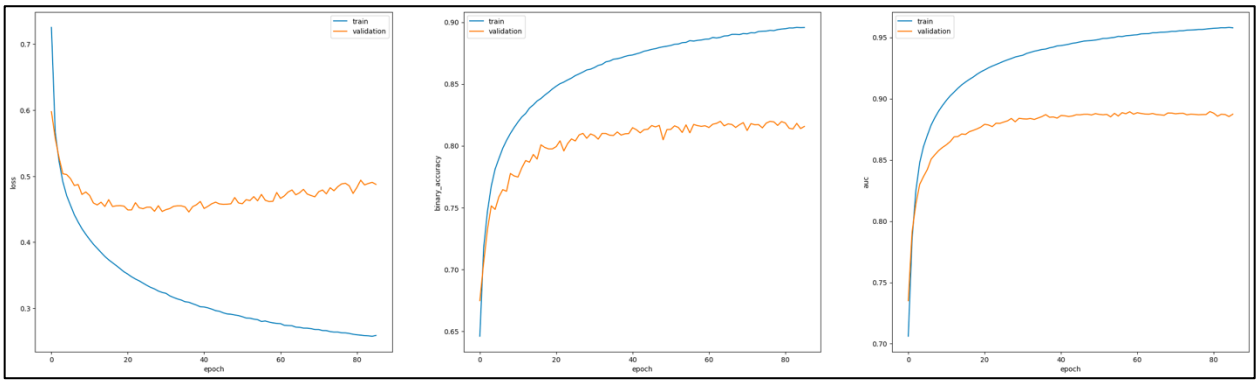
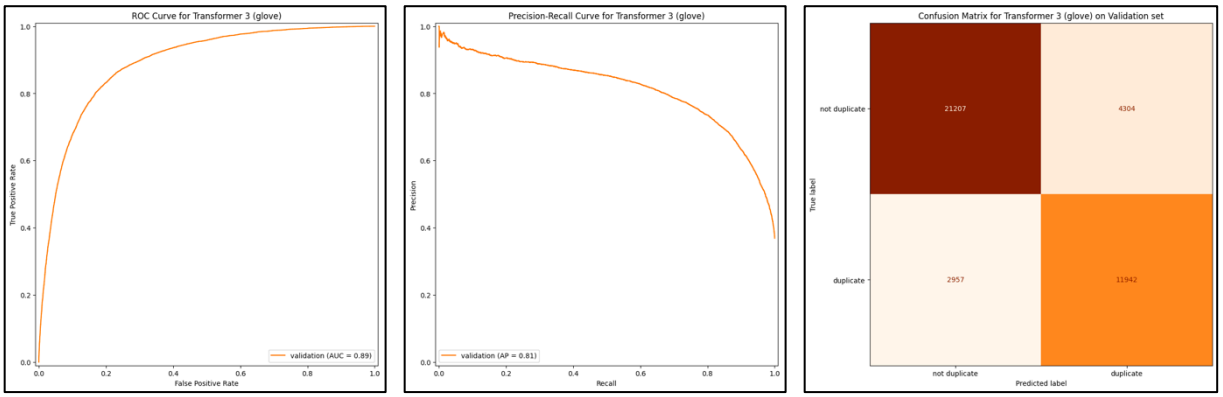


Figure 84

ROC Curve, Precision-Recall Curve and Confusion Matrix



From the training curves in Figure 83, the validation metrics started drifting from the training metrics as soon as epoch 5, but the gap kept increasing compared to Transformer (FastText). The gap persisted till the time the training stopped at epoch 100. The validation loss is increasing slowly, and the binary accuracy and AUC score for the validation set are still

improving, but very slowly. Further training time might have improved the validation results, but at the risk of overfitting on the training dataset. The ROC Curve and Precision-Recall curve in Figure 84, have a much better shape compared to the previous models, and there are roughly 10.6% false positives and 7% false negatives, which are better than all the models seen above. This model got an accuracy of 0.820317, precision of 0.735073, recall of 0.801530, F1 score of 0.766865, AUC of 0.889366 and AUPRC of 0.814508.

Table 15

Result Comparison

Model	Accuracy	Precision	Recall	F1	AUC	AUPRC
RNN (GloVe)	0.6240	0.4406	0.4303	0.4354	0.5683	0.5106
RNN (FastText)	0.6249	0.3761	0.5331	0.5415	0.5075	0.3584
GRU (Glove)	0.8198	0.7485	0.7698	0.7590	0.8895	0.8340
GRU (FastText)	0.8142	0.7284	0.7911	0.7584	0.8914	0.8325
LSTM (GloVe)	0.8189	0.7436	0.7768	0.7598	0.8885	0.8340
LSTM (FastText)	0.8202	0.7465	0.7757	0.7608	0.8911	0.8315
Transformer (GloVe)	0.8203	0.7350	0.8015	0.7668	0.8893	0.8145

Model	Accuracy	Precision	Recall	F1	AUC	AUPRC
Transformer (GloVe)	0.8187	0.7303	0.8059	0.7663	0.8946	0.8218

Evaluating Transformer with GloVe on Test set

The Transformer with GloVe Embeddings model is trained on the combined train and validation dataset, and then evaluated against the test dataset. The following are the results from that training.

Figure 85

Training curves

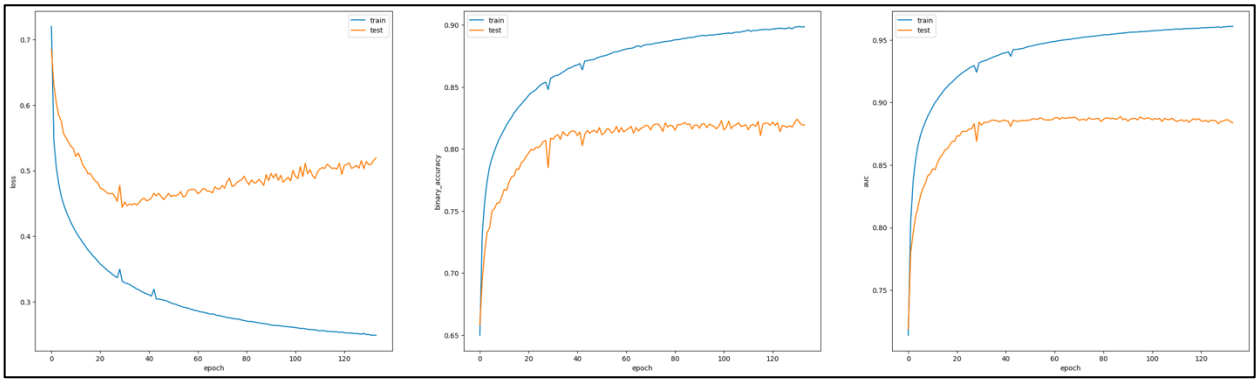
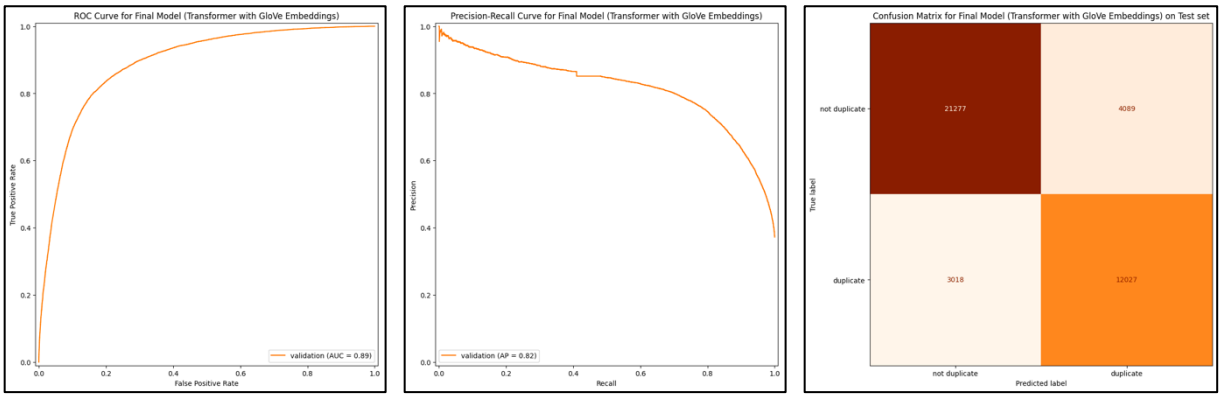


Figure 86

ROC Curve, Precision-Recall Curve and Confusion Matrix



From the training curves in Figure 85, the test metrics started drifting from the training metrics as soon as epoch 2 and the gap kept increasing throughout the training. The test loss is increasing steadily after the 30th epoch, and the binary accuracy and AUC score for the test set are still improving, but very slowly. Further training time might have improved the test results, but at the risk of overfitting on the training dataset. The ROC Curve and Precision-Recall curve in Figure 86 have a much better shape compared to the previous models, and there are roughly 10% false positives and 7% false negatives, which are better than all the models seen above. This model got a test accuracy of 0.8241, precision of 0.7462, recall of 0.7994, F1 score of 0.7719, AUC of 0.8895, AUPRC of 0.8170.

Upon retraining Transformer (Glove) using both training and validation datasets, the performance on the test set shows,

Improvement in all metrics. Notably, accuracy, precision, and F1 score all show slight improvements. This improvement could be due to the model having access to more training data, allowing it to learn a more generalized representation.

Stability in AUC and slight improvement in AUPRC. These metrics are crucial for understanding the model's ability to discriminate between classes and predict the positive class across various thresholds. The stability and slight improvement suggest that the model's ability to generalize has not been compromised despite the shift to potentially unseen test set.

GRU Model Evaluation

Using GloVe and fastText embeddings, the GRU models' performance comparison yields intriguing findings across a range of evaluation measures. The accuracy on the validation set for the GRU model with GloVe embeddings is 0.8198 in the first experiment and 0.822 in the second. The precision values of 0.7485 and 0.759, respectively, demonstrate the model's reliable

ability to distinguish between identical pairs. Recall values of 0.7698 and 0.763, which show a small fluctuation but still maintain strength, show how well the model captures most duplicate pairs. The 0.7590 and 0.763 F1-scores show a fair mix between recall and precision. Furthermore, both experiments' Area Under the ROC curve (AUC-ROC) values—0.891 in the second experiment and 0.8895 in the first—show excellent discriminative power.

The precision-recall curve's (AUC-PR) values of 0.8340 and 0.838 show consistent performance in differentiating between duplicates and non-duplicates. In contrast, the GRU model with fastText embeddings performs admirably with only minor deviations. In the first experiment, this model's accuracy is 0.8142; in the second, it increases to 0.822. The precision ratings of 0.7284 and 0.756, respectively, indicate a consistent improvement in the accurate identification of duplicates. Recall values of 0.7911 and 0.768 show a minor decline in duplicate pair identification, but it is still robust. The 0.7584 and 0.762 F1-scores show a well-balanced performance. AUC-PR scores of 0.8325 and 0.839 highlight the model's dependability in class distinction, while AUC-ROC values of 0.8914 and 0.894 indicate good discriminating capacity.

Both models perform well overall, with the fastText model demonstrating somewhat superior ROC AUC and PR AUC values in the second experiment, suggesting its possible benefit in specific paraphrase recognition scenarios.

Conclusion

In conclusion, the Quora Question Pairs dataset evaluation of multiple deep learning models emphasizes the significance of word embeddings and model architecture in attaining superior performance for paraphrase identification tasks. When it comes to important measures like accuracy, precision, recall, and F1-score, LSTM and Transformer models—especially those that use FastText embeddings—consistently beat other models, demonstrating their greater

capacity to grasp semantic similarities across queries. Additionally, GRU models show promise for this job with strong performance, particularly in precision and F1-score. On the other hand, less sophisticated RNN models using GloVe and FastText embeddings perform worse, indicating that more intricate architectures are required to accurately reflect paraphrase relationships.All things considered, the results highlight how important it is to use the right model topologies and embeddings in order to improve paraphrase identification systems' performance. Notably, the Transformer with GloVe embeddings is the best-performing model; it demonstrated remarkable robustness and adaptability for the Quora Question Pair identification task. The effectiveness of GRU-based models in precisely identifying duplicate questions is highlighted by their good performance with both GloVe and FastText embeddings, the impressive outcomes of LSTM networks with FastText, and the superior performance of Transformer models. These understandings are essential for creating applications that enhance user experience and information retrieval quality, such as automated question-answering systems and content recommendation engines.

Limitations

Even with the encouraging outcomes the existing research has few limitations. The model's ability to capture domain-specific or context-dependent semantics may be hindered by its dependence on pre-trained word embeddings such as GloVe and FastText. This could impact performance in situations where language usage deviates from the training corpus of the embeddings. Large computing resources are required for complex model architectures such as Transformers and LSTMs, which results in lengthy training times and restricted hyperparameter adjustment. This problem is made worse by the fact that Google Colaboratory lacks GPU acceleration. Furthermore, the Quora dataset's intrinsic biases could limit the models'

generalizability across other datasets or domains, and the absence of domain-specific fine-tuning restricts their suitability for specialized applications. AI decision-making processes and user trust depend on interpretability and explainability of the models; these factors were not considered while using standard performance indicators. In addition, the study did not fully investigate dataset bias, which prevented the development of impartial and equitable AI systems.

Future Scope

There are other intriguing avenues for future research in paraphrase identification systems. Accuracy and robustness can be improved by integrating sophisticated architectures like BERT or GPT, and better performance can be achieved by investigating ensemble methods that mix numerous models and take use of their individual strengths. Transfer learning can shorten training times and boost efficiency by fine-tuning pre-trained models using huge, varied datasets for paraphrase identification. Applying domain adaptation strategies can make content more applicable in situations outside of the Quora dataset. By improving the interpretability and explainability of the models, user confidence and transparency can be increased, which will make these systems more dependable and easier to use. Equitable AI systems will be achieved using fairness-aware algorithms and methods to reduce biases in training data. More comprehensive and varied datasets covering a greater variety of paraphrase methods and settings will aid in the training of more broadly applicable models. More complex and precise paraphrase recognition systems can be achieved by working with interdisciplinary teams to integrate linguistic, cognitive, and psychological insights into model building.

References

Abishek, K., Hariharan, B. R., & Valliyammai, C. (2019). *An Enhanced Deep Learning Model for Duplicate Question Pairs Recognition*. Retrieved 3 14, 2024, from https://link.springer.com/chapter/10.1007/978-981-13-0514-6_73

About Keras 3. (2023). Retrieved from Keras: https://keras.io/about/

Agirre, E., Màrquez, L., & Wicentowski, R. (2007). *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007).* Prague, Czech Republic: Association for Computational Linguistics.

Ahmad, W., Khan, H. U., Iqbal, T., & Iqbal, S. (2023). Attention-Based Multi-Channel Gated Recurrent Neural Networks: A Novel Feature-Centric Approach for Aspect-Based Sentiment Classification. IEEE Access, 11, 54408-54424. https://doi.org/10.1109/ACCESS.2023.3281889

Ansari, N., & Sharma, R. (2020). Identifying Semantically Duplicate Questions Using Data Science Approach: A Quora Case Study. ACM Conference. Retrieved from http://arxiv.org/abs/2004.11694

Choi, J., Kim, T., & Lee, S.-g. (2019). Cell-aware Stacked LSTMs for Modeling Sentences. *Proceedings of Machine Learning Research 101. ACML 2019*, pp. 1172-1187. Asian Conference on Machine Learning.

Dean, J., Monga, R., & Ghemawat, S. (n.d.). *TensorFlow: Large-scale machine learning on heterogeneous systems*. Retrieved 3 15, 2024, from Google Research: http://download.tensorflow.org/paper/whitepaper2015.pdf

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *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). Minneapolis: Association for Computational Linguistics.

Dolan, W. B., & Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. *Proceedings of the Third International Workshop on Paraphrasing (IWP2005).*

Godbole, A., Dalmia, A., & Sahu, S. K. (2018). Siamese Neural Networks with Random Forest for Detecting Duplicate Question Pairs. arXiv preprint arXiv:1801.07288. Retrieved from http://arxiv.org/abs/1801.07288

Graesser, L., Sharma, L., Nangia, N., & Evci, U. (2019). Natural Language Understanding with the Quora Question Pairs Dataset. *ArXiv, abs/1907.01041*.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A. & Mikolov, T. (2018). *Learning Word Vectors for 157 Languages.* Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) https://fasttext.cc/docs/en/crawl-vectors.html

He, R., Ravula, A., Kanagal, B., & Ainslie, J. (2021). RealFormer: Transformer Likes Residual Attention. *Findings of the Association for Compliterat*

Hochreiter, S., & Schmidhuber, J. (1997, November 15). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780.

Hosking, T., & Lapata, M. (2021, May 31). Factorising meaning and form for intent-preserving paraphrasing. arXiv.org. https://doi.org/10.48550/arXiv.2105.15053

Ishmam, A. M., & Sharmin, S. (2019). Hateful Speech Detection in Public Facebook Pages for the Bengali Language. In Proceedings of the 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 613-618). IEEE. https://doi.org/10.1109/ICMLA.2019.00104

Iyer, S., Dandekar, N., & Csernai, K. (2017). *First Quora Dataset Release: Question Pairs*. Retrieved from Data @ Quora: https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

Kumari, R., Mishra, R., Malviya, S., & Tiwary, U. S. (1970, January 1). Detection of semantically equivalent question pairs. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-030-68449-5_2

Li, G., Zhang, P., & Jia, C. (2018). Attention Boosted Sequential Inference Model. *arXiv: Computation and Language*.

Liu, F., Jiao Y., Massiah J., Yilmaz E., Havrylov, S. (2021). Trans-Encoder: Unsupervised sentence-pair modeling through self- and mutual-distillations. arXiv.org. https://doi.org/10.48550/arXiv.2109.13059

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, (pp. 56-61).

Neculoiu, P., Versteegh, M., & Rotaru, M. (2016). Learning text similarity with Siamese

recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP* (pp. 148-157).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics.

Pennington, J., Socher, R., & Manning, C. D. (2014). *GloVe: Global Vectors for Word Representation*. Glove: Global vectors for word representation. https://nlp.stanford.edu/projects/glove/

Putera Khano, M. N. A., Saputro, D. R. S., Sutanto, & Wibowo, A. (2023). Sentiment Analysis with Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) Algorithms. BAREKENG: Journal of Mathematics and Its Applications, 17(4), 2235-2242. https://doi.org/10.30598/barekengvol17iss4pp2235-2242

Schröer, C., Kruse, F., & Jorge, M. G. & Gómez, J. M. (2021). *A Systematic Literature Review on Applying CRISP-DM Process Model,* Retrieved 4 2, 2024, from https://doi.org/10.1016/j.procs.2021.01.199

Shih, C.-H., Yan, B.-C., Liu, S.-H., & Chen, B. (2017). Investigating Siamese LSTM networks for text categorization. In 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC) (pp. 641-646). Kuala Lumpur, Malaysia. https://doi.org/10.1109/APSIPA.2017.8282104

Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023). A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU. arXiv preprint arXiv:2305.17473.

Steiner, B., DeVito, Z., Chintala, S., Gross, S., Paszke, A., Massa, F., . . . Bai, J. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Retrieved 3 15, 2024, from https://nips.cc/conferences/2019/acceptedpapersinitial

Tan, C., Wei, F., Wang, W., Lv, W., & Zhou, M. (2018). Multiway Attention Networks for Modeling Sentence Pairs. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence Main track.* (pp. 4411-4417). International Joint Conference on Artificial Intelligence.

Tung, A., & Xu, E. (2023). Determining Entailment of Questions in the Quora Dataset. Stanford University. Retrieved from https://nlp.stanford.edu/projects/snli/.

Tay, V., Tran, V. Q., Ruder, S., Gupta, J., Chung, H. W., Bahri, D., . . . Metzler, D. (2022). Charformer: Fast Character Transformers via Gradient-based Subword Tokenization. *ICLR 2022 Poster.* International Conference on Learning Representations.

Vaswani, A., Shazeer, N., Niki, P., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *31st Conference on Neural Information Processing Systems (NIPS 2017).* Long Beach: Neural Information Processing Systems.

W. Bao, W. Bao, J. Du, Y. Yang and X. Zhao, "Attentive Siamese LSTM Network for Semantic Textual Similarity Measure," *2018 International Conference on Asian Language Processing (IALP)*, Bandung, Indonesia, 2018, pp. 312-317, doi: 10.1109/IALP.2018.8629212.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. Retrieved 3 14, 2024, from https://openreview.net/pdf?id=rj4km2r5t7

Wang, Z., Hamza, W., & Florian, R. (2017). Bilateral Multi-Perspective Matching for Natural Language Sentences. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence Main track* (pp. 4144-4150). Melbourne: ICJAI.

Runqi Yang, Jianhai Zhang, Xing Gao, Feng Ji, and Haiqing Chen. 2019. Simple and Effective Text Matching with Richer Alignment Features. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4699–4709, Florence, Italy. Association for Computational Linguistics.

Zhang, R., Zhou, Q., Wu, B., Li, W., & Mo, T. (2020). What Do Questions Exactly Ask? MFAE: Duplicate Question Identification with Multi-Fusion Asking Emphasis. *Proceedings of the 2020 SIAM International Conference on Data Mining* (pp. 226-234). Society for Industrial and Applied Mathematics.

Zulqarnain, M., Ghazali, R., Ghouse, M. G., & Mushtaq, M. F. (2019). Efficient processing of GRU based on word embedding for text classification. International Journal on Informatics Visualization, 3(4), 377-384.

Agarwal, B., Ramampiaro, H., Langseth, H., & Ruocco, M. (2018). A deep network model for paraphrase detection in short text messages. *Information Processing & Management*, *54*(6), 922–937. https://doi.org/10.1016/j.ipm.2018.06.005

Appendix A

Data Cleaning

Figure A1

Loading the data into a Pandas dataframe

Import packages

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import pandas as pd
import nltk
import os
```

```
RAW_DATA_FOLDER = "raw_data" # create raw_data folder before running this line
STAGING_DATA_FOLDER = "staging_data" # create staging_data folder before running this line
CLEAN_DATA_FOLDER = "clean_data" # create clean_data folder before running this line
nltk.download("punkt") # This downloads the helper files for word_tokenize
nltk.download("stopwords") # Helper files for stopwords
```

Read the Dataset into a Pandas Dataframe

```
df = pd.read_csv(os.path.join(RAW_DATA_FOLDER, "quora_duplicate_questions.tsv"), sep="\t")
# os.path.join helps to join raw_data folder with the dataset file name using / this is make sure this notebook runs on windows as well
```

```
df.head()
```

```
df.shape
```

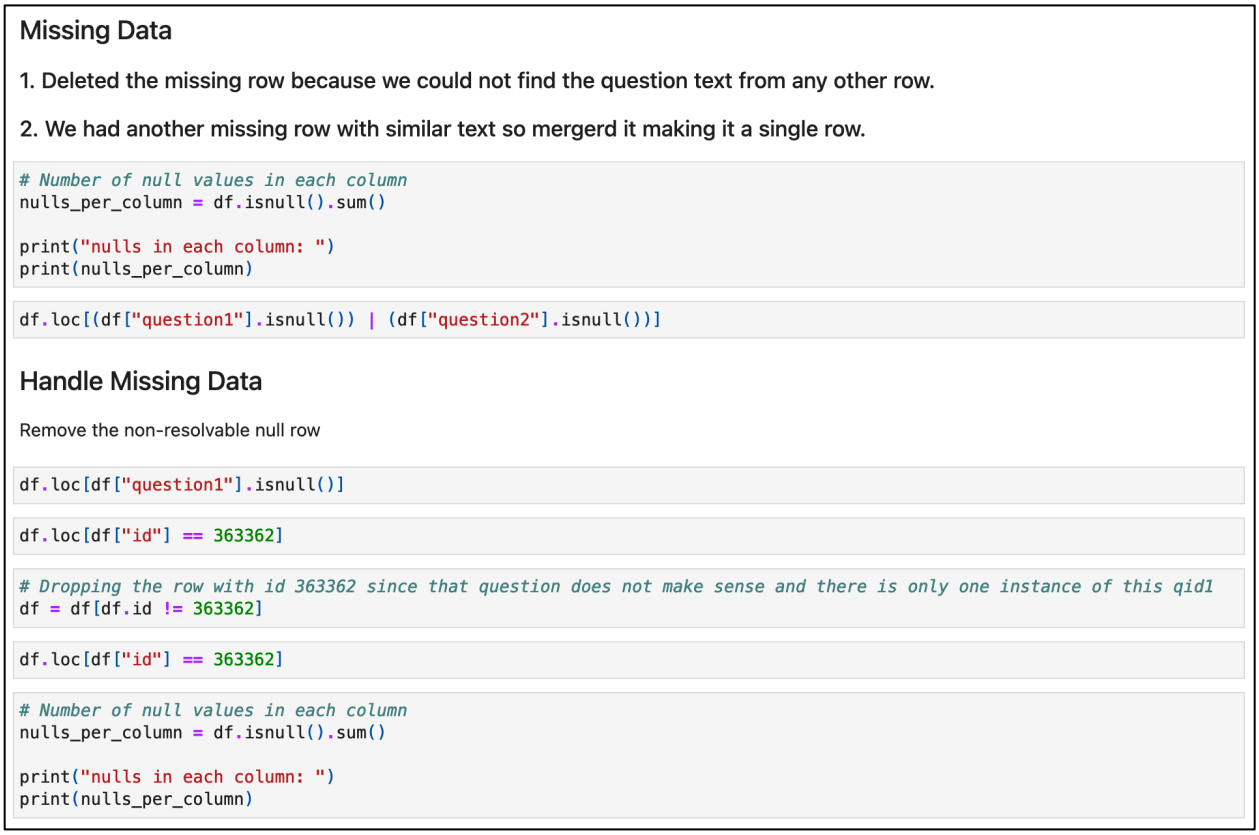
```
df["is_duplicate"].sum() # count of label 1
```

```
len(df)-(df["is_duplicate"].sum()) # Count of label 0
```

```
(len(df)-(df["is_duplicate"].sum()))/(len(df)) # the ratio of 0:1
```

Figure A2

Handling missing data



Missing Data

1. Deleted the missing row because we could not find the question text from any other row.

2. We had another missing row with similar text so mergerd it making it a single row.

```
# Number of null values in each column
nulls_per_column = df.isnull().sum()

print("nulls in each column: ")
print(nulls_per_column)
```

```
df.loc[(df["question1"].isnull()) | (df["question2"].isnull())]
```

Handle Missing Data

Remove the non-resolvable null row

```
df.loc[df["question1"].isnull()]
```

```
df.loc[df["id"] == 363362]
```

```
# Dropping the row with id 363362 since that question does not make sense and there is only one instance of this qid1
df = df[df.id != 363362]
```

```
df.loc[df["id"] == 363362]
```

```
# Number of null values in each column
nulls_per_column = df.isnull().sum()

print("nulls in each column: ")
print(nulls_per_column)
```

Figure A3

Handling resolvable nulls in Quora Question Pair data

Handle Data Inconsistency 1

Handle resolvable nulls - question text can be obtained from other example

```
df.loc[df["question2"].isnull()]
```

```
# we can see that qui2 is same for both since this id is NaN it is deleted from the data.
# Merging two rows with the similar question to make it one row
df.loc[105780, "qid2"] = df.loc[201841, "qid1"]
df.loc[105780, "question2"] = df.loc[201841, "question1"]
df = df[df.id != 201841]
```

```
df.loc[df["question2"].isnull()]
```

```
df.loc[df["id"] == 105780]
```

```
# Setting the is_duplicate value to 1 since it is duplicate
df.loc[105780, "is_duplicate"] = 1
```

```
df.loc[df["id"] == 105780]
```

```
# Again checking number of null values in each column
nulls_per_column = df.isnull().sum()

print("nulls in each column: ")
print(nulls_per_column)
```

```
df.shape # two rows has been removed now which were not necessary
```

Handle Data Inconsistency 2

Handle possible duplicate data

```
df[["qid1", "qid2", "question1", "question2", "is_duplicate"]].duplicated().sum()
```

```
df["uuid"] = df.apply(lambda row: tuple(sorted([row["qid1"], row["qid2"]])), axis=1)
df.head()
```

```
df[df["uuid"].duplicated()]
df = df.drop(columns="uuid")
```

Figure A4

Handling questions with duplicate text and multiple question IDs

Handle Data Inconsistency 3

Handle question texts with multiple question ids

```
# Identify duplicate question text in the questions dataframe and show the first 30 duplicates
qs[qs["question"].duplicated(keep=False)].sort_values(["question", "qid"]).head(n=30)
```

```
# Sorting the above questions on the length of the question
qs[qs["question"].duplicated(keep=False)].sort_values(
    ["question", "qid"]
).drop_duplicates("question").sort_values("question", key=lambda x: x.str.len()).head(n=20)
```

Figure A5

Removing duplicates from the question bank

Remove the duplicate questions from question bank, and generate mapping for question pairs

```
# Keep the first qid for each question
qs_dedup = qs.drop_duplicates(subset="question", keep="first")
# Mapping from question text to new qid (the one we kept)
qid_map = pd.Series(qs_dedup['qid'].values, index=qs_dedup['question']).to_dict()
# Reverse mapping to find duplicates and their corresponding kept qid
reverse_map = qs.set_index('qid')['question'].map(qid_map)
# Creating the final mapping from all old qids to the new (kept) qids
final_qid_map = pd.Series(reverse_map, index=qs['qid']).to_dict()
# rename qs_dedup to qs
qs = qs_dedup
```

Figure A6

Updating the question pairs with new question IDs after removing duplicates

Update question pairs dataframe by renaming the ids

for example 25026 - ? all the ? now has id 25026

```
q_pairs['qid1'] = q_pairs['qid1'].map(final_qid_map)
q_pairs['qid2'] = q_pairs['qid2'].map(final_qid_map)
```

Figure A7

Removing very short questions and saving the data to staging

Remove questions with length less than 4

```
# Checking questions with length less than 4
qs[qs.question.str.len() < 4]
```

```
# Adding the questions with length less than 4 to the remove quids list and deleting only questions with len less than 4
remove_qids += qs[qs.question.str.len() < 4]["qid"].to_list()
qs = qs[qs.question.str.len() >= 4]
```

Remove further questions with meaningless text

```
qs.sort_values("question", key=lambda x: x.str.len()).head(n=30) # Sorted the questions based on the length of the question
```

```
# remove the above questions as well
remove_qids += qs.sort_values("question", key=lambda x: x.str.len()).head(n=30)["qid"].to_list()
qs = qs[
    ~qs.qid.isin(
        qs.sort_values("question", key=lambda x: x.str.len()).head(n=30)["qid"]
    )
]
```

Make the question bank and question pairs consistent after deleting questions

```
for qid in remove_qids:
    qs = qs[qs.qid != qid]
    q_pairs = q_pairs[(q_pairs.qid1 != qid) & (q_pairs.qid2 != qid)]
```

Save the question bank and question pairs to staging data

```
q_pairs.to_csv(os.path.join(STAGING_DATA_FOLDER, "question_pairs.csv"), index=False)
qs.to_csv(os.path.join(STAGING_DATA_FOLDER, "questions.tsv"), sep="\t", index=False)

# Saving to a file
```

The data is now in 3NF and consistent.

Figure A8

Tokenizing the question text in question bank

Tokenization

This is a collection of all the words/tokens in our dataset. This is useful for tokenization and embeddings.

Need to install package nltk (natural language tool kit). pip install nltk

```
qs.head(30)
```

```
# converting to lower case

qs["question"] = qs["question"].str.lower()
```

```
qs.head()
```

```
# Here the each sentence is converted to a list of words aka tokenization.

qs["question"] = qs["question"].map(word_tokenize)
```

```
qs.head(30)
```

```
qs.tail(30)
```

Figure A9

Removing stop words from question text

Removing Stop Words

It is kind of making sense that is having the important or original meaning of the sentence preserved and removeing the rest.

```
stop_words = set(stopwords.words('english'))
```

```
list(stop_words)[:30]
```

```
qs["question"] = qs["question"].map(lambda row : [x for x in row if x not in stop_words])
# excludes all the words that are in stop_words from each question and includes rest.
```

```
qs.head(30)
```

```
qs.iloc[:, 1].tolist()[:5]
```

Remove questions that are potentially meaningless now

```
# questions with length less than 2 after tokenization, stop word removal
qs[qs.question.str.len() < 2]
```

```
for qid in qs[qs.question.str.len() < 2]["qid"].to_list():
    qs = qs[qs.qid != qid] # Removing questions with length less than 2 from question bank
    q_pairs = q_pairs[(q_pairs.qid1 != qid) & (q_pairs.qid2 != qid)] # Removing relevant qids from question pairs
```

Figure A10

Creating vocabulary of tokens using the tokenized question text

Create vocabulary

Vocabulary is created because these words will be used to find corresponding embeddings from pre trained embeddings like glove and fasttext.

```
vocab = set()
for line in qs.iloc[:, 1].tolist():
    for token in line:
        vocab.add(token)
vocab = pd.DataFrame({"token": sorted(vocab)})
vocab = vocab.reset_index().rename(columns={"index": "id"})
vocab["id"] = vocab["id"].map(lambda x: x + 1) # 0 id will be used for padding
vocab.to_csv(os.path.join(CLEAN_DATA_FOLDER, "vocab.tsv"), sep="\t", index=False)
```

```
vocab.head()
```

Figure A11

Convert question text to respective token IDs and save to staging

Convert question tokens to IDs

```
vocab_mapping = {row["token"]: row["id"] for _, row in vocab.iterrows()}
qs["question"] = qs["question"].map(lambda x: [vocab_mapping[token] for token in x])
```

Save the cleaned questions

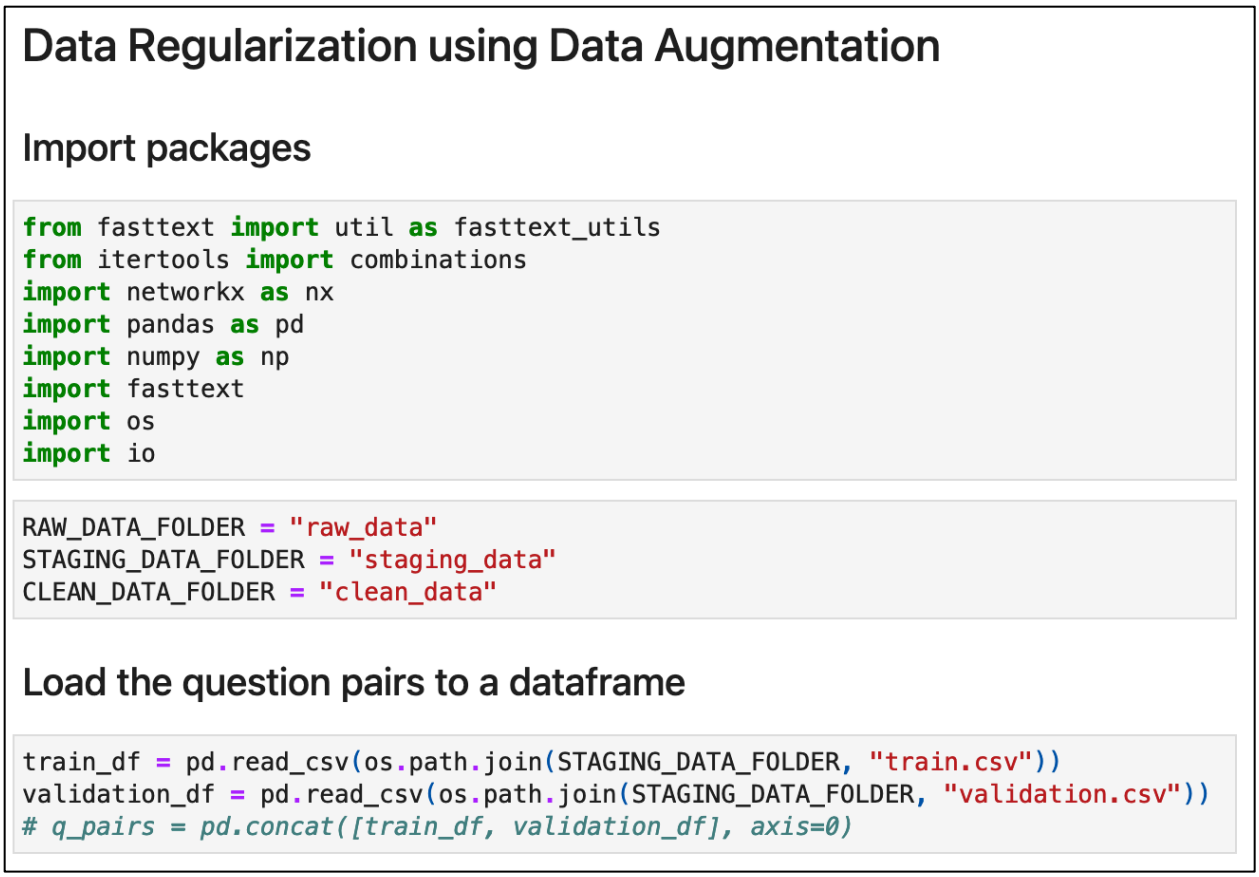
```
qs.to_csv(os.path.join(STAGING_DATA_FOLDER, "questions.tsv"), sep="\t", index=False)
q_pairs.to_csv(os.path.join(STAGING_DATA_FOLDER, "question_pairs.csv"), index=False)
```

Appendix B

Data Transformation

Figure B1

Load the train and validation dataset for augmentation



Data Regularization using Data Augmentation

Import packages

```
from fasttext import util as fasttext_utils
from itertools import combinations
import networkx as nx
import pandas as pd
import numpy as np
import fasttext
import os
import io
```

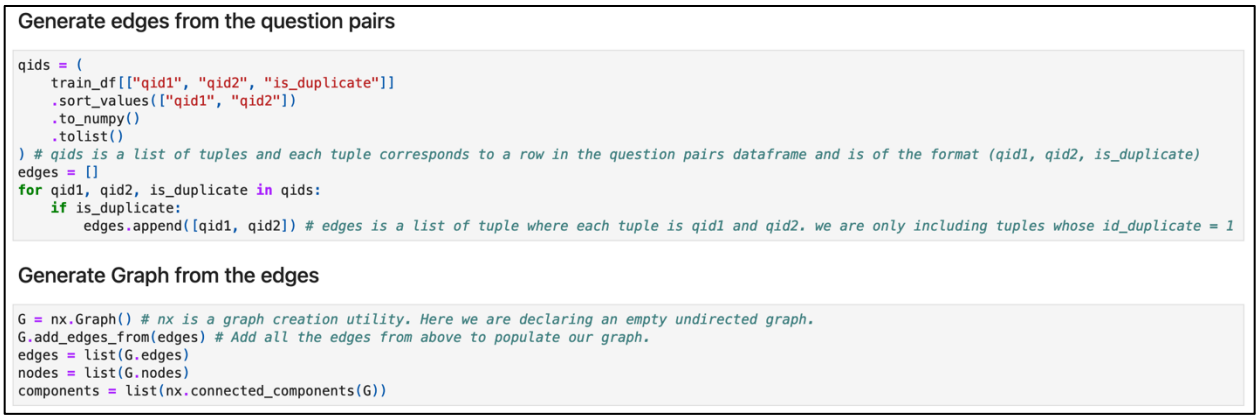
```
RAW_DATA_FOLDER = "raw_data"
STAGING_DATA_FOLDER = "staging_data"
CLEAN_DATA_FOLDER = "clean_data"
```

Load the question pairs to a dataframe

```
train_df = pd.read_csv(os.path.join(STAGING_DATA_FOLDER, "train.csv"))
validation_df = pd.read_csv(os.path.join(STAGING_DATA_FOLDER, "validation.csv"))
# q_pairs = pd.concat([train_df, validation_df], axis=0)
```

Figure B2

Identifying all connected components



Generate edges from the question pairs

```
qids = (
    train_df[["qid1", "qid2", "is_duplicate"]]
    .sort_values(["qid1", "qid2"])
    .to_numpy()
    .tolist()
) # qids is a list of tuples and each tuple corresponds to a row in the question pairs dataframe and is of the format (qid1, qid2, is_duplicate)
edges = []
for qid1, qid2, is_duplicate in qids:
    if is_duplicate:
        edges.append([qid1, qid2]) # edges is a list of tuple where each tuple is qid1 and qid2. we are only including tuples whose id_duplicate = 1
```

Generate Graph from the edges

```
G = nx.Graph() # nx is a graph creation utility. Here we are declaring an empty undirected graph.
G.add_edges_from(edges) # Add all the edges from above to populate our graph.
edges = list(G.edges)
nodes = list(G.nodes)
components = list(nx.connected_components(G))
```

Figure B3

Dropping trivial components

Drop trivial components (number of nodes in component is 2)

```
components = [x for x in components if len(x) != 2]
```

Figure B4

Generating new edges based on connected components

Generate all potential edges

for each component, identify all potential edges, and only choose edges which don't have a direct path in the component

```
def can_add_edge(G, u, v):
    if G.has_edge(u, v) or G.has_edge(v, u):
        # Edge already exists
        return False
    if nx.has_path(G, u, v) or nx.has_path(G, v, u):
        # Adding this edge doesn't connect different components
        return True
    return False
```

```
new_edges = []
for component in components:
    potential_edges = combinations(sorted(component), 2)
    for u, v in potential_edges:
        if can_add_edge(G, u, v):
            new_edges.append((u, v))
```

These are the number of new edges (duplicate question pairs)

```
len(new_edges)
```

```
91227
```

Figure B5

Adding the newly generated question pairs

Add the new edges to train question pairs

```
df = pd.DataFrame([{"qid1": u, "qid2": v, "is_duplicate": 1} for u, v in new_edges])
q_pairs = pd.concat([train_df[["qid1", "qid2", "is_duplicate"]], df], axis=0)
q_pairs = q_pairs.sort_values(["qid1", "qid2"]).reset_index().drop(columns="index")
```

```
q_pairs.head()
```

Figure B6

Checking the class balance after augmentation

```
q_pairs.to_csv(os.path.join(STAGING_DATA_FOLDER, "train.csv"), index_label="id")
```

```
total_pairs = len(q_pairs) + len(validation_df)
positive_pairs = q_pairs["is_duplicate"].sum() + validation_df["is_duplicate"].sum()
(total_pairs-positive_pairs)/total_pairs # the ratio of 0:1
```

```
0.5044656497778716
```

Figure B7

Loading the fasttext model

Data Transformation using Fasttext and Glove Embeddings

Data Transformation Using FastText

pip install fasttext - to install the helper code for fasttext

```
# 300 is the default embedding dimensions for fasttext.
# each word in the vocab will be assigned a 300-dimensional vector
embed_dimension = 300
```

```
# load the fasttext model which contains all the words it has already learned and the corresponding embeddings
ft_model = fasttext.load_model(os.path.join(RAW_DATA_FOLDER, "fasttext.cc.en.300d.bin"))
```

Figure B8

Loading the vocabulary

```
# Reading vocab.tsv which we created earlier
vocab = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER, "vocab.tsv"),
    sep="\t",
    index_col="id",
    keep_default_na=False, # n/a can be read/interpreted as null in pandas. So we are telling Pandas dont do it please
)
vocab = vocab["token"].tolist() # stroing all the tokens from vocab.tsv
```

Figure B9

Save the fasttext embeddings

```
embeddings = [[0.0 for _ in range(embed_dimension)]] # first entry for padding elements
for token in vocab:
    embeddings.append(ft_model.get_word_vector(token))

# This generates word embeddings/vectors for all the words/tokens in vocab variable using fasttext model (ft_model)
```

```
embeddings = np.asarray(embeddings)
np.save(os.path.join(CLEAN_DATA_FOLDER, f"fasttext.{embed_dimension}d.embeddings.npy"), embeddings)
```

Figure B10

Reducing fasttext dimensions using PCA

Data Reduction using PCA for FastText Embeddings

```
# each word in the vocab will be assigned a 100-dimensional vector
embed_dimension = 100
```

```
# this does PCA internally
if embed_dimension < 300:
    fasttext_utils.reduce_model(ft_model, embed_dimension)
```

```
# Reading vocab.tsv which we created earlier
vocab = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER, "vocab.tsv"),
    sep="\t",
    index_col="id",
    keep_default_na=False, # n/a can be read/interpreted as null in pandas. So we are telling Pandas dont do it please
)
vocab = vocab["token"].tolist() # stroing all the tokens from vocab.tsv
```

Figure B11

Save the 100 dimension embeddings of fasttext

```
embeddings = [[0.0 for _ in range(embed_dimension)]] # first entry for padding elements
for token in vocab:
    embeddings.append(ft_model.get_word_vector(token))

# This generates word embeddings/vectors for all the words/tokens in vocab variable using fasttext model (ft_model)
```

```
embeddings = np.asarray(embeddings)
np.save(os.path.join(CLEAN_DATA_FOLDER, f"fasttext.{embed_dimension}d.embeddings.npy"), embeddings)
```

Figure B12

Load the GloVe embeddings

Data Transformation using GloVe Embeddings

```
embed_dimension = 300
```

```
# Loading Glove embedding
glove = {}
fin = io.open(
    os.path.join(RAW_DATA_FOLDER, f"glove.6B.{embed_dimension}d.txt"),
    "r",
    encoding="utf-8",
    newline="\n",
    errors="ignore",
)
for line in fin:
    tokens = line.rstrip().split(" ")
    glove[tokens[0]] = list(map(float, tokens[1:]))
```

```
vocab = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER, "vocab.tsv"),
    sep="\t",
    index_col="id",
    keep_default_na=False,
)
vocab = vocab["token"].tolist()
```

```
# storing words/ tokens in oov which are not present in the glove embeddings
oov = []
for token in vocab:
    if token not in glove:
        oov.append(token)
oov = sorted(oov, key=len)
```

Figure B13

Generate embeddings for unknown words and save the glove embeddings

```python
def find_words_from_dict(dictionary, input_string):
    n = len(input_string)
    dp = [False] * (n + 1)
    dp[0] = True
    parent = [-1] * (
        n + 1
    )  # To store the starting index of the word that ends at position i

    # Filling dp array
    for i in range(1, n + 1):
        for j in range(i):
            if dp[j] and input_string[j:i] in dictionary:
                dp[i] = True
                parent[i] = j
                break

    # If dp[n] is False, it's not possible to construct the string
    if not dp[-1]:
        return -1

    # Recovering the words from the input string
    result = [0] * embed_dimension
    current = n
    count = 0
    while current > 0:
        start = parent[current]
        word = input_string[start:current]
        result = [result[i] + dictionary[word][i] for i in range(embed_dimension)]
        count += 1
        current = start

    return [x / count for x in result]
```

```python
#if the above function returns -1 then storing those words/ tokens in ooov
ooov = []
for oov_token in oov:
    embedding = find_words_from_dict(glove, oov_token)
    if embedding == -1:
        ooov.append(oov_token)
ooov = sorted(ooov, key=len)
```

```python
embeddings = [[0.0] * embed_dimension]
for token in vocab:
    if token in glove:
        embeddings.append(glove[token])
    elif token in ooov:
        embeddings.append([0.0] * embed_dimension)
    else:
        embeddings.append(find_words_from_dict(glove, token))
```

```python
embeddings = np.asarray(embeddings)
np.save(os.path.join(CLEAN_DATA_FOLDER, f"glove.{embed_dimension}d.embeddings.npy"), embeddings)
```

Figure B14

Load the data for splitting into train, validation and test

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split
STAGING_DATA_FOLDER = "staging_data"
q_pairs = pd.read_csv(os.path.join(STAGING_DATA_FOLDER, "question_pairs.csv"))
q_pairs.head()
train, test = train_test_split(q_pairs, test_size=0.2, random_state=42)
validation, test = train_test_split(test, test_size=0.5, random_state=42)
train.to_csv(os.path.join(STAGING_DATA_FOLDER, "train.csv"), index=False)
validation.to_csv(os.path.join(STAGING_DATA_FOLDER, "validation.csv"), index=False)
test.to_csv(os.path.join(STAGING_DATA_FOLDER, "test.csv"), index=False)
```

Figure B15

Combine the datasets and save to clean data folder

```
import pandas as pd
import os

STAGING_DATA_FOLDER = "staging_data"
CLEAN_DATA_FOLDER = "clean_data"
qs = pd.read_csv(os.path.join(STAGING_DATA_FOLDER, "questions.tsv"), sep="\t")
train = pd.read_csv(os.path.join(STAGING_DATA_FOLDER, "train.csv"))
validation = pd.read_csv(os.path.join(STAGING_DATA_FOLDER, "validation.csv"))
test = pd.read_csv(os.path.join(STAGING_DATA_FOLDER, "test.csv"))
train = (
    train.merge(qs, how="left", left_on="qid1", right_on="qid")
    .rename(columns={"question": "question1"})
    .drop(columns="qid1")
    .merge(qs, how="left", left_on="qid2", right_on="qid")
    .rename(columns={"question": "question2"})
    .drop(columns="qid2")[["question1", "question2", "is_duplicate"]]
)
validation = (
    validation.merge(qs, how="left", left_on="qid1", right_on="qid")
    .rename(columns={"question": "question1"})
    .drop(columns="qid1")
    .merge(qs, how="left", left_on="qid2", right_on="qid")
    .rename(columns={"question": "question2"})
    .drop(columns="qid2")[["question1", "question2", "is_duplicate"]]
)
test = (
    test.merge(qs, how="left", left_on="qid1", right_on="qid")
    .rename(columns={"question": "question1"})
    .drop(columns="qid1")
    .merge(qs, how="left", left_on="qid2", right_on="qid")
    .rename(columns={"question": "question2"})
    .drop(columns="qid2")[["question1", "question2", "is_duplicate"]]
)
train.to_csv(os.path.join(CLEAN_DATA_FOLDER, "train.tsv"), sep="\t", index=False)
validation.to_csv(os.path.join(CLEAN_DATA_FOLDER, "validation.tsv"), sep="\t", index=False)
test.to_csv(os.path.join(CLEAN_DATA_FOLDER, "test.tsv"), sep="\t", index=False)
```

Appendix C

Transformer Model

Figure C1

Load the packages and load the dataset

Quora Question Pairs (using Keras and Tensorflow)

Import packages

```
import os
os.environ["KERAS_BACKEND"] = "tensorflow"
import pandas as pd
import numpy as np
import keras
import math
from random import shuffle
keras.utils.set_random_seed(42)
```

Declare constants and dataset hyperparameters

```
CLEAN_DATA_FOLDER = "clean_data"
MODEL_DATA_FOLDER = "models"
BATCH_SIZE = 256
```

Prepare dataset (train and validation)

```
train_df = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER, "train.tsv"),
    sep="\t",
    converters={
        "question1": lambda x: [int(val) for val in x.strip("[]").split(", ")],
        "question2": lambda x: [int(val) for val in x.strip("[]").split(", ")],
    },
)
validation_df = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER, "validation.tsv"),
    sep="\t",
    converters={
        "question1": lambda x: [int(val) for val in x.strip("[]").split(", ")],
        "question2": lambda x: [int(val) for val in x.strip("[]").split(", ")],
    },
)
test_df = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER, "test.tsv"),
    sep="\t",
    converters={
        "question1": lambda x: [int(val) for val in x.strip("[]").split(", ")],
        "question2": lambda x: [int(val) for val in x.strip("[]").split(", ")],
    },
)
train_df = pd.concat([train_df, validation_df])
train_df["question1"] = train_df["question1"].apply(tuple)
train_df["question2"] = train_df["question2"].apply(tuple)
train_df = train_df.drop_duplicates()
train_df["question1"] = train_df["question1"].apply(list)
train_df["question2"] = train_df["question2"].apply(list)
```

Figure C2

Model hyperparameters

```
LEARNING_RATE = 1e-3
EMBED_NAME = "glove"
EMBED_DIMS = 100
ATTN_HEADS = 4
ATTN_DIMS = 64
FF_DIMS = 32
MAX_LEN = 85
FF_DROPOUT = 0.1
EPOCHS = 1000
```

Figure C3

Prepare the loader to feed data to the Transformer Model

Prepare Dataloader

```
class QuoraQuestionPairDataset(keras.utils.PyDataset):
    def __init__(self, x, y, batch_size, shuffling=True, **kwargs):
        super().__init__(**kwargs)
        self.x = x
        self.y = y
        self.batch_size = batch_size
        self.shuffling = shuffling

    def __len__(self):
        return math.ceil(len(self.y) / self.batch_size)

    def __getitem__(self, idx):
        batch_x_0 = self.x[0][idx * self.batch_size : (idx + 1) * self.batch_size]
        batch_x_1 = self.x[1][idx * self.batch_size : (idx + 1) * self.batch_size]
        batch_y = self.y[idx * self.batch_size : (idx + 1) * self.batch_size]
        batch_x_0 = keras.utils.pad_sequences(batch_x_0, padding='post')
        batch_x_1 = keras.utils.pad_sequences(batch_x_1, padding='post')
        batch_y = np.expand_dims(np.asarray(batch_y), axis=-1)
        # if idx % 2 == 0:
        #     return (batch_x_1, batch_x_0), batch_y
        return (batch_x_0, batch_x_1), batch_y

    def on_epoch_end(self):
        index = list(range(len(self.y)))
        if self.shuffling:
            shuffle(index)
        self.x = ([self.x[0][i] for i in index], [self.x[1][i] for i in index])
        self.y = [self.y[i] for i in index]

train_dataset = QuoraQuestionPairDataset(
    x=(train_df["question1"].tolist(), train_df["question2"].tolist()),
    y=train_df["is_duplicate"].tolist(),
    batch_size=BATCH_SIZE,
)
test_dataset = QuoraQuestionPairDataset(
    x=(test_df["question1"].tolist(), test_df["question2"].tolist()),
    y=test_df["is_duplicate"].tolist(),
    batch_size=BATCH_SIZE,
    shuffling=False
)
```

Figure C4

Defining the positional embeddings

```python
# reference - https://keras.io/examples/nlp/text_classification_with_transformer/
@keras.saving.register_keras_serializable(package="qqp")
class PositionalEmbedding(keras.layers.Layer):
    def __init__(self, embed_dims, max_len):
        super().__init__(name="positional_embedding")
        self.embed_dims = embed_dims
        self.max_len = max_len
        self.pos_embed = keras.layers.Embedding(
            input_dim=self.max_len,
            output_dim=self.embed_dims,
            embeddings_initializer=keras.initializers.RandomUniform(seed=42),
            name="positional_embedding_internal",
        )

    def call(self, x):
        shape_x = keras.ops.shape(x)
        positions = keras.ops.arange(start=0, stop=shape_x[-1], step=1)
        positions = keras.ops.expand_dims(positions, axis=0)
        positions = keras.ops.repeat(positions, shape_x[0], axis=0)
        return self.pos_embed(positions)

    def get_config(self):
        config = super().get_config()
        config.update({"embed_dims": self.embed_dims, "max_len": self.max_len})
        return config
```

Figure C5

Defining the GloVe embedding layer

```python
def get_embedding_layer(embed_name="fasttext", embed_dim=100):
    embedding_matrix = np.load(os.path.join(CLEAN_DATA_FOLDER, f"{embed_name}.{embed_dim}d.embeddings.npy"))
    embedding_layer = keras.layers.Embedding(embedding_matrix.shape[0], embed_dim, name=f"{embed_name}")
    embedding_layer.build((1,))
    embedding_layer.trainable = False
    embedding_layer.set_weights([embedding_matrix])
    return embedding_layer
```

Figure C6

Defining the Transformer block

```python
@keras.saving.register_keras_serializable(package="qqp")
class TransformerBlock(keras.layers.Layer):
    def __init__(self, attn_heads, attn_dims, embed_dims, ff_dims, ff_dropout, name):
        super().__init__(name=name)
        self.attn_heads = attn_heads
        self.attn_dims = attn_dims
        self.embed_dims = embed_dims
        self.ff_dims = ff_dims
        self.ff_dropout = ff_dropout
        self.name = name
        self.attn = keras.layers.MultiHeadAttention(
            num_heads=self.attn_heads,
            key_dim=self.attn_dims,
            kernel_initializer=keras.initializers.GlorotUniform(seed=42),
            name=self.name + "_attn",
        )
        self.ff1 = keras.layers.Dense(
            self.ff_dims,
            activation="relu",
            kernel_initializer=keras.initializers.GlorotUniform(seed=42),
            kernel_regularizer=keras.regularizers.L2(),
            name=self.name + "_ff1",
        )
        self.ff2 = keras.layers.Dense(
            self.embed_dims,
            kernel_initializer=keras.initializers.GlorotUniform(seed=42),
            kernel_regularizer=keras.regularizers.L2(),
            name=self.name + "_ff2",
        )
        self.ln1 = keras.layers.LayerNormalization(
            epsilon=1e-6, name=self.name + "_ln1"
        )
        self.ln2 = keras.layers.LayerNormalization(
            epsilon=1e-6, name=self.name + "_ln2"
        )
        self.do1 = keras.layers.Dropout(
            rate=self.ff_dropout, seed=42, name=self.name + "_do1"
        )
        self.do2 = keras.layers.Dropout(
            rate=self.ff_dropout, seed=42, name=self.name + "_do2"
        )

    def call(self, inputs):
        attn_output = self.attn(inputs, inputs)
        attn_output = self.do1(attn_output)
        out1 = self.ln1(inputs + attn_output)
        ff1_output = self.ff1(out1)
        ff2_output = self.ff2(ff1_output)
        ff_output = self.do2(ff2_output)
        return self.ln2(out1 + ff_output)

    def get_config(self):
        config = super().get_config()
        config.update(
            {
                "attn_heads": self.attn_heads,
                "attn_dims": self.attn_dims,
                "embed_dims": self.embed_dims,
                "ff_dims": self.ff_dims,
                "ff_dropout": self.ff_dropout,
                "name": self.name,
            }
        )
        return config
```

Figure C7

Defining the Siamese network using Transformer

```python
def create_transformer_model(
    attn_heads=4,
    attn_dims=32,
    embed_name="fasttext",
    embed_dims=100,
    max_len=85,
    ff_dims=32,
    ff_dropout=0.1,
    learning_rate=3e-4,
):
    # inputs for the model, question1 and question2
    question1 = keras.Input(shape=(None,), name="question1")
    question2 = keras.Input(shape=(None,), name="question2")
    # declare embedding layer, and then apply on question1 and question2
    embedding_layer = get_embedding_layer(embed_name, embed_dims)
    question1_embed = embedding_layer(question1)
    question2_embed = embedding_layer(question2)
    # declare positional embeddings layer, and then apply to input question1 and qusetion2
    pos_embedding_layer = PositionalEmbedding(embed_dims, max_len)
    question1_pos = pos_embedding_layer(question1)
    question2_pos = pos_embedding_layer(question2)
    # declare summing layer, and add the token and position embedding on question1 and question2
    summing_layer = keras.layers.Add(name="add_embeddings")
    question1_pos_embed = summing_layer([question1_embed, question1_pos])
    question2_pos_embed = summing_layer([question2_embed, question2_pos])
    # create transformer layer, and then apply on question1 and question2
    transformer_layer_1 = TransformerBlock(
        attn_heads, attn_dims // 2, embed_dims, ff_dims // 2, ff_dropout, name="transformer_block_1"
    )
    question1_output1 = transformer_layer_1(question1_pos_embed)
    question2_output1 = transformer_layer_1(question2_pos_embed)
    # create another transformer layer, and then apply on question1 and question2
    transformer_layer_2 = TransformerBlock(
        attn_heads, attn_dims // 2, embed_dims, ff_dims // 2, ff_dropout, name="transformer_block_2"
    )
    question1_output2 = transformer_layer_2(question1_output1)
    question2_output2 = transformer_layer_2(question2_output1)
    # global average pooling to collapse the sequence dimension
    pooling_layer = keras.layers.GlobalAveragePooling1D(name="pooling")
    question1_pool = pooling_layer(question1_output2)
    question2_pool = pooling_layer(question2_output2)
    # find the difference between 2 questions
    subtract = keras.layers.Subtract(name="subtract")([question1_pool, question2_pool])
    # latent layer
    latent = keras.layers.Dense(
        60,
        activation="relu",
        kernel_initializer=keras.initializers.GlorotUniform(seed=42),
        kernel_regularizer=keras.regularizers.L2(),
        name="latent",
    )(subtract)
    # output classification layer
    output = keras.layers.Dense(
        1,
        activation="sigmoid",
        kernel_initializer=keras.initializers.GlorotUniform(seed=42),
        name="classifier",
    )(latent)
    # create model
    model = keras.Model(
        name=f"final_model", inputs=(question1, question2), outputs=output
    )
    # compile the model with loss, optimizer and metrics
    model.compile(
        loss=keras.losses.BinaryCrossentropy(),
        optimizer=keras.optimizers.Adam(learning_rate=learning_rate),
        metrics=[keras.metrics.BinaryAccuracy(), keras.metrics.AUC()],
    )
    # return model
    return model
```

Figure C8

Create the Model and plot the model features and architecture

Create Siamese Network Model with Transformer

```
model = create_transformer_model(
    ATTN_HEADS, ATTN_DIMS, EMBED_NAME, EMBED_DIMS, MAX_LEN, FF_DIMS, FF_DROPOUT, LEARNING_RATE
)
model.summary()
keras.utils.plot_model(
    model,
    to_file=os.path.join(MODEL_DATA_FOLDER, f"final_model.png"),
    show_shapes=True,
    show_layer_names=True,
    rankdir="LR",
    show_layer_activations=True,
    show_trainable=True,
)
```

Figure C9

Train the Siamese Network model using Transformer

Train the Siamese Network Model

```
history = model.fit(
    train_dataset,
    epochs=EPOCHS,
    callbacks=[
        # keras.callbacks.EarlyStopping(monitor="val_binary_accuracy", mode="max", patience=10),
        keras.callbacks.ModelCheckpoint(
            os.path.join(MODEL_DATA_FOLDER, f"final_model.keras"),
            monitor="val_binary_accuracy",
            mode="max",
            save_best_only=True,
        ),
        keras.callbacks.BackupAndRestore("backup", delete_checkpoint=False),
        keras.callbacks.CSVLogger(
            os.path.join(MODEL_DATA_FOLDER, f"final_model_logs.csv"),
            append=True,
        ),
    ],
    validation_data=test_dataset,
)
```

LSTM Model

Figure C10

LSTM Model Importing libraries and defining folder structure and hyper-parameters.

Figure C11

LSTM Model Loading the train dataset

Figure C12

LSTM model Convert lists into TensorFlow ragged tensors

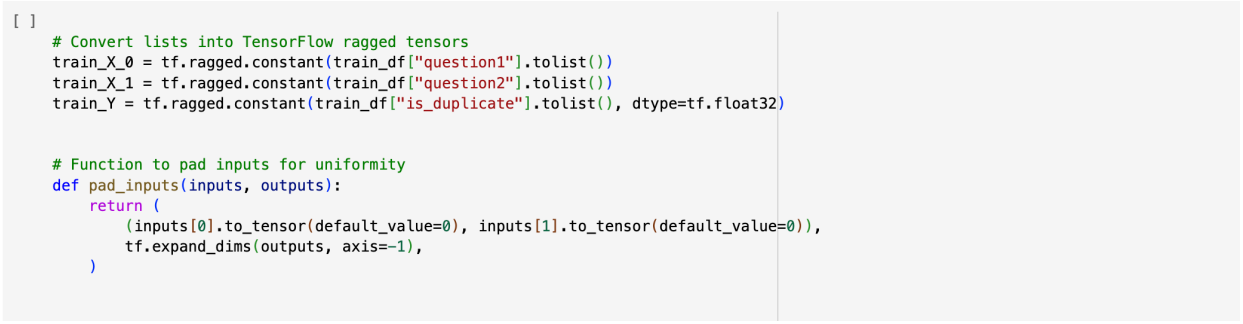


Figure C13

LSTM model Prepare dataset for training and preparing embedding layer function

```
# Prepare dataset for training and validation
train_dataset = tf.data.Dataset.from_tensor_slices(((train_X_0, train_X_1), train_Y))
train_dataset = train_dataset.shuffle(buffer_size=len(train_df)).batch(BATCH_SIZE).map(pad_inputs).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
[ ] def get_embedding_layer(embed_name="fasttext", embed_dim=100):
        embedding_matrix = np.load(os.path.join(CLEAN_DATA_FOLDER, f"{embed_name}.{embed_dim}d.embeddings.npy"))
        embedding_layer = keras.layers.Embedding(embedding_matrix.shape[0], embed_dim, name=f"{embed_name}")
        embedding_layer.build((1,))
        embedding_layer.trainable = False
        embedding_layer.set_weights([embedding_matrix])
        return embedding_layer
```

Figure C14

LSTM Model creation function

Model creation function

```
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense, Lambda, Subtract

# Model creation function
def create_lstm_model(embed_name="fasttext", embed_dim=100):
    question1 = Input(shape=(None,), dtype='int32', name="question1")
    question2 = Input(shape=(None,), dtype='int32', name="question2")

    embedding_layer = get_embedding_layer(embed_name, embed_dim)

    q1_embed = embedding_layer(question1)
    q2_embed = embedding_layer(question2)

    lstm_layer = keras.layers.LSTM(128, return_sequences=False)
    q1_encoded = lstm_layer(q1_embed)
    q2_encoded = lstm_layer(q2_embed)

    concatenated = keras.layers.concatenate([q1_encoded, q2_encoded])
    dense1 = keras.layers.Dense(64, activation='relu')(concatenated)
    output = keras.layers.Dense(1, activation='sigmoid')(dense1)

    model = keras.Model(inputs=[question1, question2], outputs=output)
    return model

model = create_lstm_model("glove", 100)
model.summary()
```

Figure C15

LSTM Model Plotting

Model Plotting

```
keras.utils.plot_model(
    model,
    to_file=os.path.join(MODEL_DATA_FOLDER, "lstm_model_glove.png"),
    show_shapes=True,
    show_layer_names=True,
)
```

Figure C16

Model Compilation and Model Training

```
# Model compilation
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=LEARNING_RATE),
    loss='binary_crossentropy',
    metrics=[keras.metrics.BinaryAccuracy(), keras.metrics.AUC()]
)

# Model training
history = model.fit(
    train_dataset,
    epochs=EPOCHS
)
```

Figure C17

LSTM Model saving

Model Saving

```
# to save model
model.save(os.path.join(MODEL_DATA_FOLDER, "lstm_glove.keras"))
```

Figure C18

LSTM Loading test data

LSTM Loading test data

```
# Load test data
test_df = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER, "test.tsv"),
    sep="\t",
    converters={
        "question1": lambda x: [int(val) for val in x.strip("[]").split(", ")],
        "question2": lambda x: [int(val) for val in x.strip("[]").split(", ")],
    }
)

# Convert lists into TensorFlow ragged tensors
test_X_0 = tf.ragged.constant(test_df["question1"].tolist())
test_X_1 = tf.ragged.constant(test_df["question2"].tolist())
test_Y = tf.ragged.constant(test_df["is_duplicate"].tolist(), dtype=tf.float32)

# Function to pad inputs for uniformity, reused from earlier
def pad_inputs(inputs, outputs):
    return (
        (inputs[0].to_tensor(default_value=0), inputs[1].to_tensor(default_value=0)),
        tf.expand_dims(outputs, axis=-1),
    )

# Prepare dataset for testing
test_dataset = tf.data.Dataset.from_tensor_slices(((test_X_0, test_X_1), test_Y))
test_dataset = test_dataset.batch(BATCH_SIZE).map(pad_inputs).prefetch(buffer_size=tf.data.AUTOTUNE)
```

Figure C19

LSTM Evaluation metrices

```
# Evaluate the model on the test set
evaluation_results = model.evaluate(test_dataset)
print("Test Loss:", evaluation_results[0])
print("Test Binary Accuracy:", evaluation_results[1])
print("Test AUC:", evaluation_results[2])
```

```
632/632 [==============================] - 4s 6ms/step - loss: 0.4872 - binary_accuracy: 0.8159 - auc: 0.8846
Test Loss: 0.4872010350227356
Test Binary Accuracy: 0.815891683101654
Test AUC: 0.8845945596694946
```

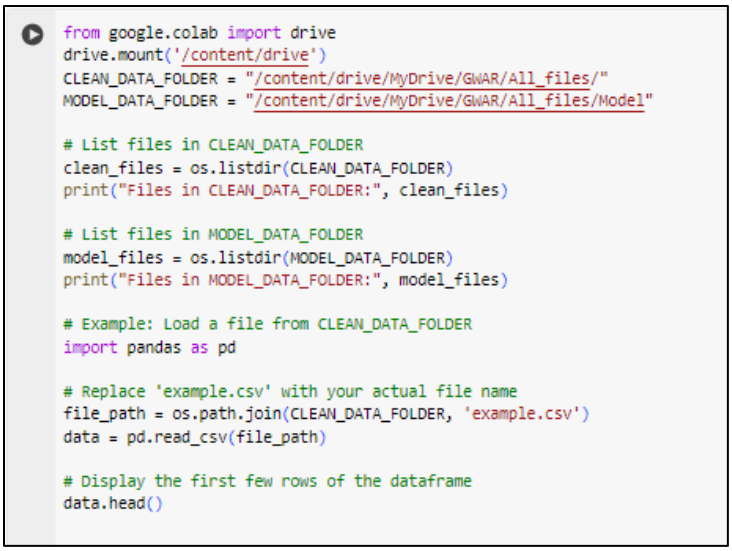
GRU Code

Figure C18

Importing necessary libraries required for GRU model development and evaluation

Figure C19

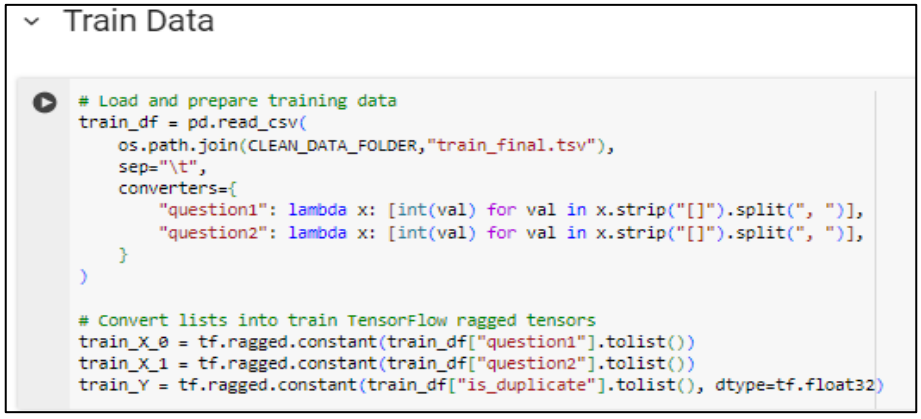
Importing Train and test data and setting up folder structure for Saving the model



```
from google.colab import drive
drive.mount('/content/drive')
CLEAN_DATA_FOLDER = "/content/drive/MyDrive/GWAR/All_files/"
MODEL_DATA_FOLDER = "/content/drive/MyDrive/GWAR/All_files/Model"

# List files in CLEAN_DATA_FOLDER
clean_files = os.listdir(CLEAN_DATA_FOLDER)
print("Files in CLEAN_DATA_FOLDER:", clean_files)

# List files in MODEL_DATA_FOLDER
model_files = os.listdir(MODEL_DATA_FOLDER)
print("Files in MODEL_DATA_FOLDER:", model_files)

# Example: Load a file from CLEAN_DATA_FOLDER
import pandas as pd

# Replace 'example.csv' with your actual file name
file_path = os.path.join(CLEAN_DATA_FOLDER, 'example.csv')
data = pd.read_csv(file_path)

# Display the first few rows of the dataframe
data.head()
```

Figure C20

Preparing training data and creating ragged tensors



Train Data

```
# Load and prepare training data
train_df = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER,"train_final.tsv"),
    sep="\t",
    converters={
        "question1": lambda x: [int(val) for val in x.strip("[]").split(", ")],
        "question2": lambda x: [int(val) for val in x.strip("[]").split(", ")],
    }
)

# Convert lists into train TensorFlow ragged tensors
train_X_0 = tf.ragged.constant(train_df["question1"].tolist())
train_X_1 = tf.ragged.constant(train_df["question2"].tolist())
train_Y = tf.ragged.constant(train_df["is_duplicate"].tolist(), dtype=tf.float32)
```

Figure C21

Preparing test data and creating ragged tensors

Test Data

```
test_df = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER, "test.tsv"),
    sep="\t",
    converters={
        "question1": lambda x: [int(val) for val in x.strip("[]").split(", ")],
        "question2": lambda x: [int(val) for val in x.strip("[]").split(", ")],
    }
)

# Convert lists into test TensorFlow ragged tensors
test_X_0 = tf.ragged.constant(test_df["question1"].tolist())
test_X_1 = tf.ragged.constant(test_df["question2"].tolist())
test_Y = tf.ragged.constant(test_df["is_duplicate"].tolist(), dtype=tf.float32)
```

Figure C22

Model Compilation and Model Training

Setting Up Embedding Layer and defining parameters

```
[ ] # Setting best or common parameters
    LEARNING_RATE = 1e-3
    BATCH_SIZE = 64
    EPOCHS = 50
```

```
# Function to pad inputs for uniformity
def pad_inputs(inputs, outputs):
    return (
        (inputs[0].to_tensor(default_value=0), inputs[1].to_tensor(default_value=0)),
        tf.expand_dims(outputs, axis=-1),
    )

def get_embedding_layer(embed_name="glove", embed_dim=100):
    embedding_matrix = np.load(os.path.join(CLEAN_DATA_FOLDER, f"{embed_name}.{embed_dim}d.embeddings.npy"))
    embedding_layer = keras.layers.Embedding(embedding_matrix.shape[0], embed_dim, name=f"{embed_name}")
    embedding_layer.build((1,))
    embedding_layer.trainable = False
    embedding_layer.set_weights([embedding_matrix])
    return embedding_layer
```

```
[ ] # Prepare dataset
    train_dataset = tf.data.Dataset.from_tensor_slices(((train_X_0, train_X_1), train_Y))
    train_dataset = train_dataset.shuffle(buffer_size=len(train_df)).batch(BATCH_SIZE).map(pad_inputs).prefetch(buffer_size=tf.data.AUTOTUNE)

    test_dataset = tf.data.Dataset.from_tensor_slices(((test_X_0, test_X_1), test_Y))
    test_dataset = test_dataset.batch(BATCH_SIZE).map(pad_inputs).prefetch(buffer_size=tf.data.AUTOTUNE)
```

Figure C23

GRU model development

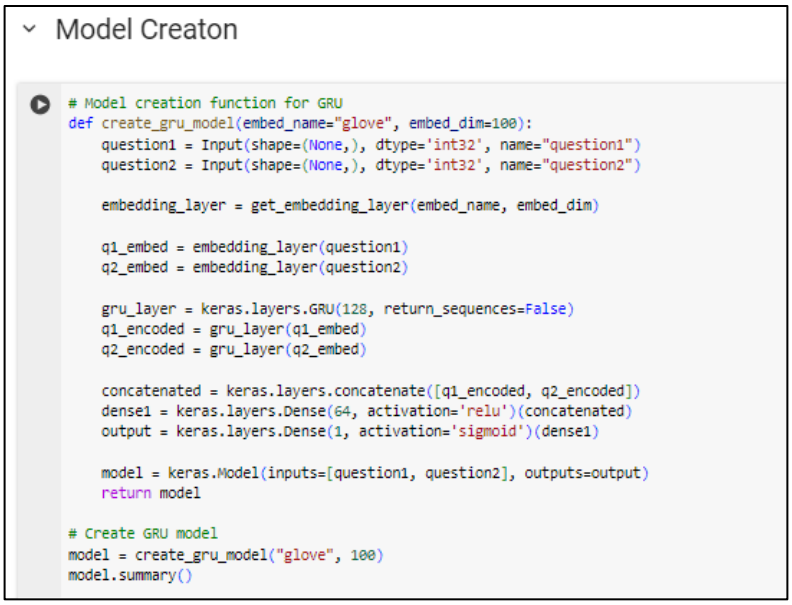


Figure C24

Compiling and saving the GRU model for future use

Figure C26

Defined a class to load the dataset to run the saved model

Figure C25

Creating data structure to store the evaluation metrics

Figure C26

Loading the Test dataset for final predictions

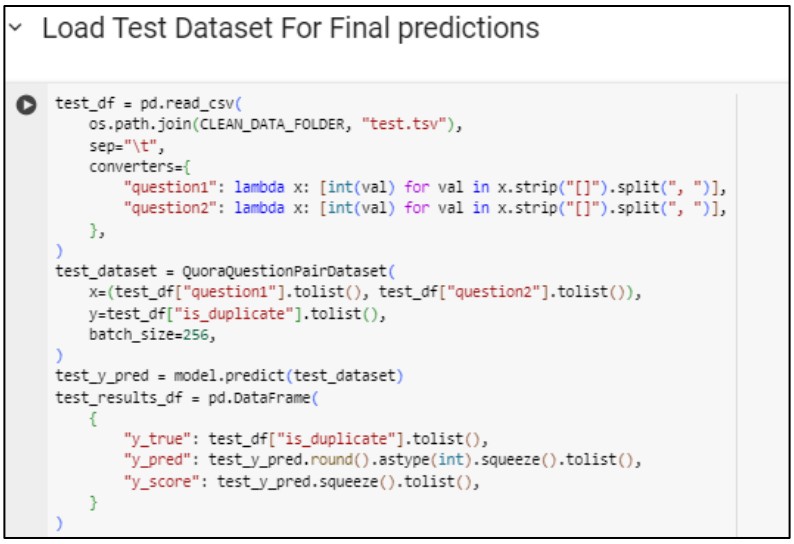
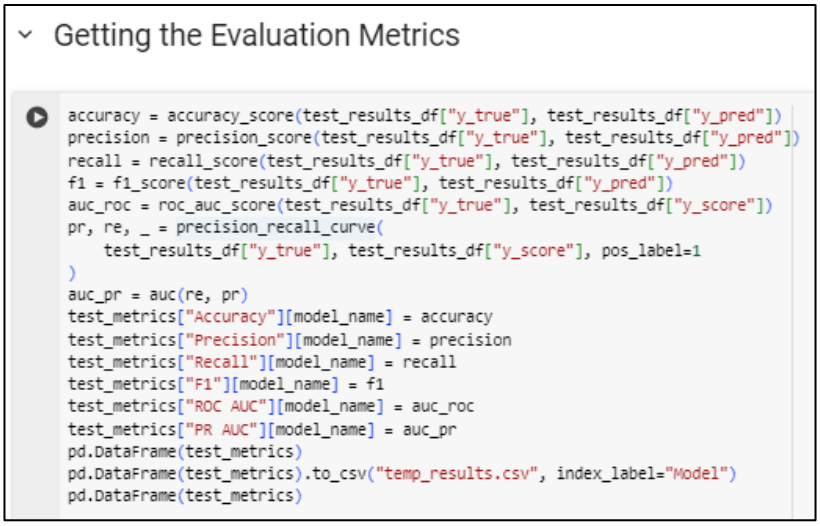


Figure C27

Run evaluation metrics



RNN Code

Figure C28

Importing Data

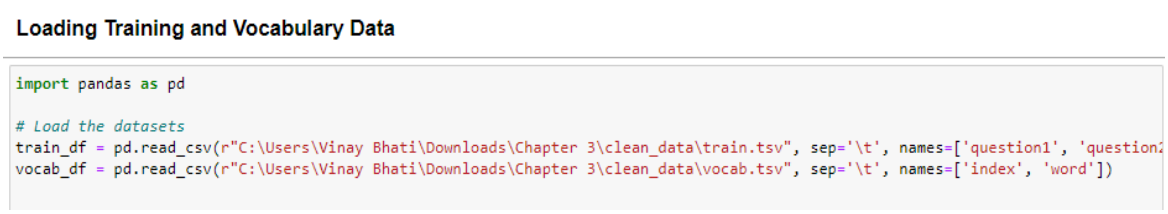


Figure C29

Calculating and Displaying Statistics on Question Lengths

Analyzing Sequence Lengths of Questions Pair

```
import pandas as pd
import ast

sequence_lengths_q1 = train_data['question1'].apply(len)
sequence_lengths_q2 = train_data['question2'].apply(len)

# You can check the maximum length or a specific percentile
max_length_q1 = sequence_lengths_q1.max()
max_length_q2 = sequence_lengths_q2.max()

percentile_length_q1 = sequence_lengths_q1.quantile(0.95)  # 95th percentile
percentile_length_q2 = sequence_lengths_q2.quantile(0.95)  # 95th percentile

print("Maximum sequence length in Question 1:", max_length_q1)
print("Maximum sequence length in Question 2:", max_length_q2)
print("95th percentile of sequence length in Question 1:", percentile_length_q1)
print("95th percentile of sequence length in Question 2:", percentile_length_q2)
```

Figure C30

Data Type Verification and Conversion for Question Columns

Preprocessing Data Types in Question Columns

```
# Check the data type of the first entry in each column to understand what you're working with
print(type(train_data['question1'].iloc[0]))
print(type(train_data['question2'].iloc[0]))

# If they are strings, convert them to lists using eval
if isinstance(train_data['question1'].iloc[0], str):
    X1 = train_data['question1'].apply(eval)
    X2 = train_data['question2'].apply(eval)
else:
    X1 = train_data['question1']
    X2 = train_data['question2']
```

```python
# Function to convert string representations of lists to actual lists
def string_to_list(column):
    return [ast.literal_eval(item) for item in column]

# Convert strings to lists
train_data['question1'] = string_to_list(train_data['question1'])
train_data['question2'] = string_to_list(train_data['question2'])
test_data['question1'] = string_to_list(test_data['question1'])
test_data['question2'] = string_to_list(test_data['question2'])
val_data['question1'] = string_to_list(val_data['question1'])
val_data['question2'] = string_to_list(val_data['question2'])

# Assuming the format of each file is: question1, question2, is_duplicate
# Function to concatenate question pairs and then pad them
def prepare_data(question1, question2):
    combined = [q1 + q2 for q1, q2 in zip(question1, question2)]
    return pad_sequences(combined, dtype='int32', padding='post')

# Prepare training data
X_train = prepare_data(train_data['question1'], train_data['question2'])
y_train = train_data['is_duplicate'].values

# Prepare test data
X_test = prepare_data(test_data['question1'], test_data['question2'])
y_test = test_data['is_duplicate'].values

# Prepare validation data
X_val = prepare_data(val_data['question1'], val_data['question2'])
y_val = val_data['is_duplicate'].values
```

Figure C31

SimpleRNN model for fasttext

Building SimpleRNN model for fasttext

```python
# Load the embedding matrix
embedding_matrix_100d = np.load(r"C:\Users\Vinay Bhati\Downloads\Chapter 3\clean_data\fasttext.100d.embeddings.npy")

# Define the model with the SimpleRNN architecture
model_simple_rnn = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=vocab_size, output_dim=100,
                              embeddings_initializer=tf.keras.initializers.Constant(embedding_matrix_100d),
                              input_length=20, trainable=False),
    tf.keras.layers.SimpleRNN(64, return_sequences=False),  # Using SimpleRNN
    tf.keras.layers.Dropout(0.5),  # Dropout rate adjusted
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Figure C32

Optimizing and Executing the SimpleRNN Training Process for fasttext

Configuring and Training the SimpleRNN Model with Callback

```
# Initialize a new optimizer
optimizer = tf.keras.optimizers.Adam()

# Define the learning rate scheduler
def scheduler(epoch, lr):
    if epoch < 10:
        return float(lr)
    else:
        return float(lr * tf.math.exp(-0.1))

lr_callback = tf.keras.callbacks.LearningRateScheduler(scheduler)

# Compile the model with the new optimizer
model_glove_rnn.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])

# Early stopping callback with increased patience
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Assuming X_train, y_train, X_val, y_val, X_test, and y_test are already prepared and loaded
# Train the model using the train dataset
history_glove_rnn = model_glove_rnn.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    batch_size=64,
    epochs=10,  # Increased epochs
    callbacks=[early_stopping, lr_callback]
)
```

Figure C33

Performance Metrics Calculation for the RNN Model (fasttext)

Evaluating Model Performance on Test Data(fasttext)

```
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, auc, precision_recall_curve
import numpy as np

# Assuming that y_test is your actual labels and model_glove_rnn is your trained model
# Get predictions for the test dataset
y_pred_probs = model_simple_rnn.predict(X_val)
y_pred = (y_pred_probs > 0.5).astype('int32')  # Convert probabilities to binary output

# Calculate Precision, Recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Calculate Area Under the ROC Curve (AUC-ROC)
roc_auc = roc_auc_score(y_test, y_pred_probs)

print("Area Under ROC Curve (AUC-ROC):", roc_auc)

# Calculate Precision-Recall Curve and Area Under the Precision-Recall Curve (AUC-PR)
precision_curve, recall_curve, thresholds = precision_recall_curve(y_test, y_pred_probs)
auc_pr = auc(recall_curve, precision_curve)

print("Area Under Precision-Recall Curve (AUC-PR):", auc_pr)
```

Figure C34

Exporting the Trained SimpleRNN(fasttext) Model

Saving the SimpleRNN Model to Disk ¶

```
# Save the SimpleRNN model
model_simple_rnn.save(r"C:\Users\Vinay Bhati\Downloads\Chapter 3\clean_data\simple_rnn_model.keras", save_format='tf')
```

Figure C35

SimpleRNN model for glove

Building SimpleRNN Model for Glove

```
# Load the GloVe embedding matrix
glove_embedding_matrix = np.load(r"C:\Users\Vinay Bhati\Downloads\Chapter 3\clean_data\glove.100d.embeddings.npy")

# Define the model with the SimpleRNN architecture using GloVe embeddings
model_glove_rnn = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=vocab_size, output_dim=100,
                              embeddings_initializer=tf.keras.initializers.Constant(glove_embedding_matrix),
                              input_length=20, trainable=False),
    tf.keras.layers.SimpleRNN(64, return_sequences=False),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Figure C36

Optimizing and Executing the SimpleRNN Training Process for glove

Configuring and Training the SimpleRNN Model with Callback for glove

```
# Initialize a new optimizer
optimizer = tf.keras.optimizers.Adam()

# Define the learning rate scheduler
def scheduler(epoch, lr):
    if epoch < 10:
        return float(lr)
    else:
        return float(lr * tf.math.exp(-0.1))

lr_callback = tf.keras.callbacks.LearningRateScheduler(scheduler)

# Compile the model with the new optimizer
model_glove_rnn.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])

# Early stopping callback with increased patience
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Assuming X_train, y_train, X_val, y_val, X_test, and y_test are already prepared and loaded
# Train the model using the train dataset
history_glove_rnn = model_glove_rnn.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    batch_size=64,
    epochs=10,  # Increased epochs
    callbacks=[early_stopping, lr_callback]
)
```

Figure C37

Performance Metrics Calculation for the RNN Model (glove)

Evaluating Model Performance on Test Data

```
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, auc, precision_recall_curve
import numpy as np

# Assuming that y_test is your actual labels and model_glove_rnn is your trained model
# Get predictions for the test dataset
y_pred_probs = model_glove_rnn.predict(X_test)
y_pred = (y_pred_probs > 0.5).astype('int32')  # Convert probabilities to binary output

# Calculate Precision, Recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Calculate Area Under the ROC Curve (AUC-ROC)
roc_auc = roc_auc_score(y_test, y_pred_probs)

print("Area Under ROC Curve (AUC-ROC):", roc_auc)

# Calculate Precision-Recall Curve and Area Under the Precision-Recall Curve (AUC-PR)
precision_curve, recall_curve, thresholds = precision_recall_curve(y_test, y_pred_probs)
auc_pr = auc(recall_curve, precision_curve)

print("Area Under Precision-Recall Curve (AUC-PR):", auc_pr)
```

Figure C38

Exporting the Trained SimpleRNN(glove) Model

Saving the SimpleRNN model

```
# Save the GloVe-based RNN model
model_glove_rnn.save(r"C:\Users\Vinay Bhati\Downloads\Chapter 3\clean_data\glove_rnn_model.keras", save_format='tf')
```

Appendix D

Inference code

Figure D1

Load the model and vocabulary for inferencing

Import packages

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import pandas as pd
import keras
import nltk
import os
```

```
CLEAN_DATA_FOLDER = "clean_data"
MODEL_DATA_FOLDER = "models"
nltk.download("stopwords")
nltk.download("punkt")
```

Load the model

```
LEARNING_RATE = 1e-3
EMBED_NAME = "glove"
EMBED_DIMS = 100
ATTN_HEADS = 4
ATTN_DIMS = 64
FF_DIMS = 32
MAX_LEN = 85
FF_DROPOUT = 0.1
from models.final_model import create_transformer_model
model = create_transformer_model(
    ATTN_HEADS, ATTN_DIMS, EMBED_NAME, EMBED_DIMS, MAX_LEN, FF_DIMS, FF_DROPOUT, LEARNING_RATE
)
model.load_weights(os.path.join(MODEL_DATA_FOLDER, f"final_model.keras"))
```

Load the vocabulary

```
vocab = pd.read_csv(
    os.path.join(CLEAN_DATA_FOLDER, "vocab.tsv"),
    sep="\t",
    keep_default_na=False,
)
vocab_mapping = {row["token"]: row["id"] for _, row in vocab.iterrows()}
```

Figure D2

Get the user input

User Input

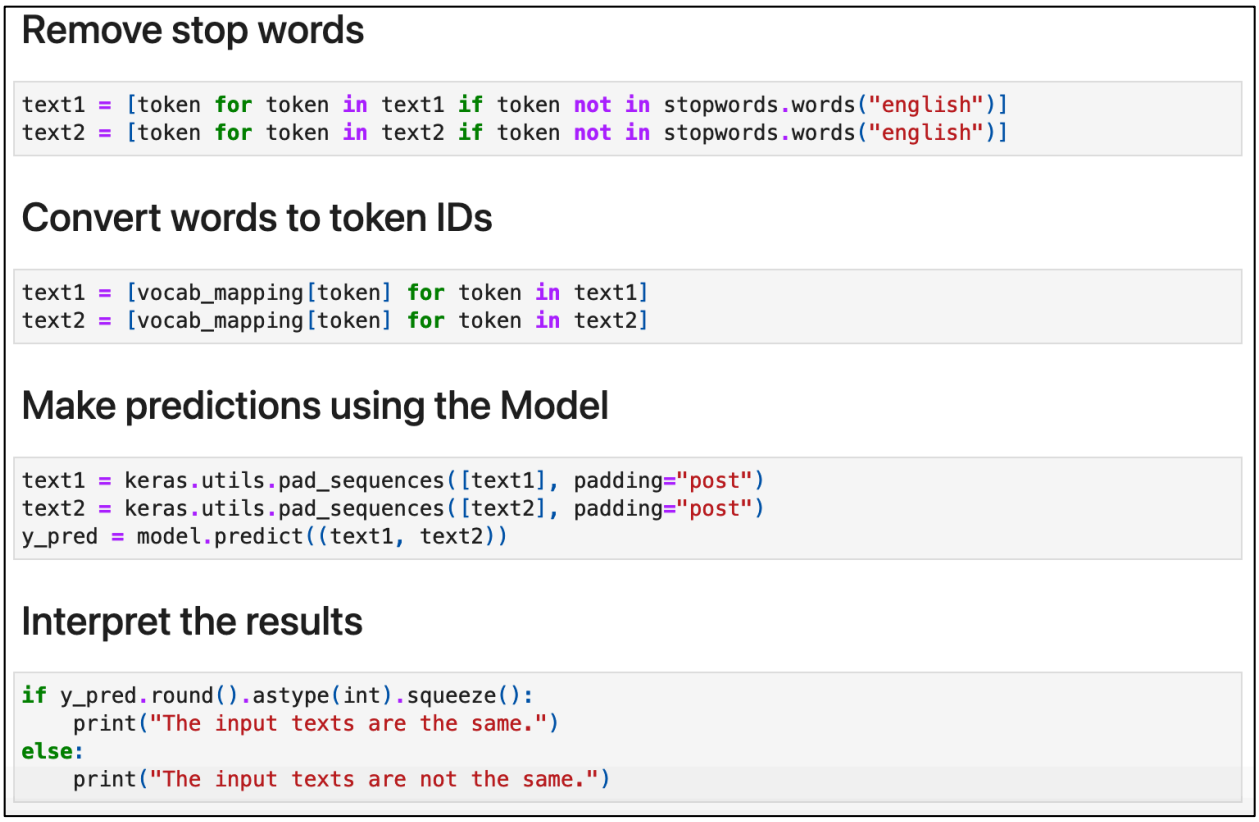
```
TEXT1 = "Are stocks and security the same?"
TEXT2 = "Are stocks considered security?"
```

Figure D3

Tokenize the input sentences

Figure D4

Remove stop words, convert to tokens and predict



Appendix E

Link to GitHub Repository

https://github.com/NehaBais/paraphrase-identification-using-quora-question-pairs