Duplicate Question Detection

Team Semantics

***Abstract*—This project explores multiple deep learning models for sequence-to-sequence tasks and duplicate question detection. The implemented architectures incorporate mechanisms such as attention, memory networks, and Siamese frameworks. The models were evaluated using the Quora Question Pairs dataset, which reflects real-world linguistic variations and challenges in detecting semantic similarity. The architectures include Baseline Neural Networks, Siamese models, encoder-decoder systems, transformers, and advanced mechanisms such as Gated Residual Networks and multi-head attention.**

I. Introduction

Duplicate question pair detection determines whether two questions are semantically equivalent despite variations in phrasing. For example, "How can I lose weight fast?" and "What are quick ways to reduce weight?" ask the same thing in different ways. This task is vital in applications such as:

- Customer support automation: Reducing redundancy in FAQs and customer queries.
- Search engine optimization: Clustering similar user questions for better search results.
- Community platforms: Improving user experience by merging related questions (e.g., Quora, Stack Overflow).
- Chatbot development: Mapping queries to predefined similar questions for better responses.

II. Dataset Overview and Preprocessing

A. Dataset Overview

The Quora dataset consists of question pairs with binary labels:

- Label 1: The questions are semantically identical.
- Label 0: The questions are semantically different.

This dataset provides an excellent benchmark for detecting duplicate questions due to its linguistic diversity and real-world relevance.

B. Preprocessing and Tensor Creation

Each question is tokenized into words, converted to lowercase, and cleaned by removing punctuation and special characters. Pre-trained GloVe embeddings (e.g., 300d) are used to represent words as dense vectors in high-dimensional space. Questions are then converted into tensors using PyTorch, padded to equal lengths, and paired with their binary labels.

III. Implemented Architectures

This section describes the deep learning models implemented for duplicate question detection.

A. Baseline Neural Network (BaselineNN)

The simplest model averages GloVe embeddings for each question, concatenates them, and passes them through fully connected layers.

- Provides a baseline for performance comparison.
- Cannot capture sequential dependencies or local patterns.

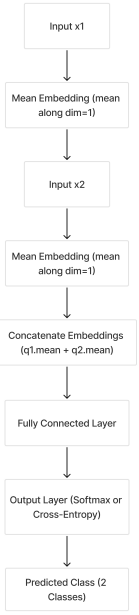

Fig. 1. Baseline Neural Network Architecture.

B. Siamese Convolutional Neural Network (SiameseCNN)

Uses CNN layers to extract local features and max pooling to capture n-gram patterns.

- Effective for local feature extraction.
- Cannot model long-term dependencies.

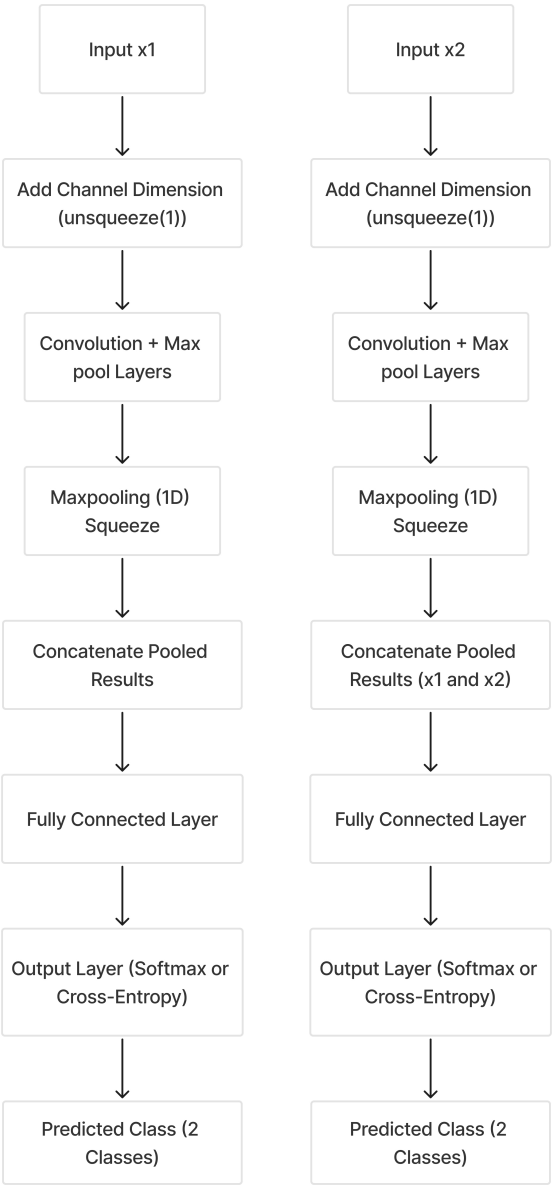

Fig. 2. Siamese CNN Architecture.

C. Siamese Long Short-Term Memory (SiameseLSTM)

Processes input questions with bidirectional LSTMs to capture sequential and contextual dependencies.

- Suitable for long-term dependencies.
- Computationally intensive compared to CNNs.

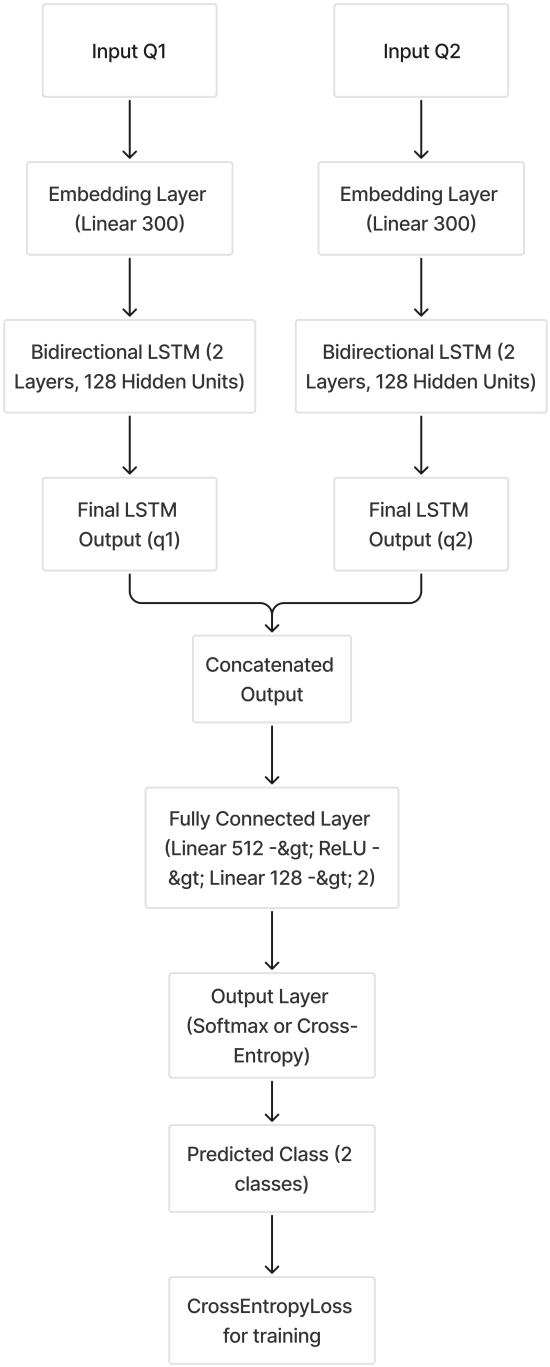

Fig. 3. Siamese LSTM Architecture.

D. Siamese LSTM with CNN (SiameseLSTM-CNN)

Combines LSTMs for sequential processing with CNNs for local feature extraction.

- Leverages both sequential and local features.
- Increased complexity and training time.

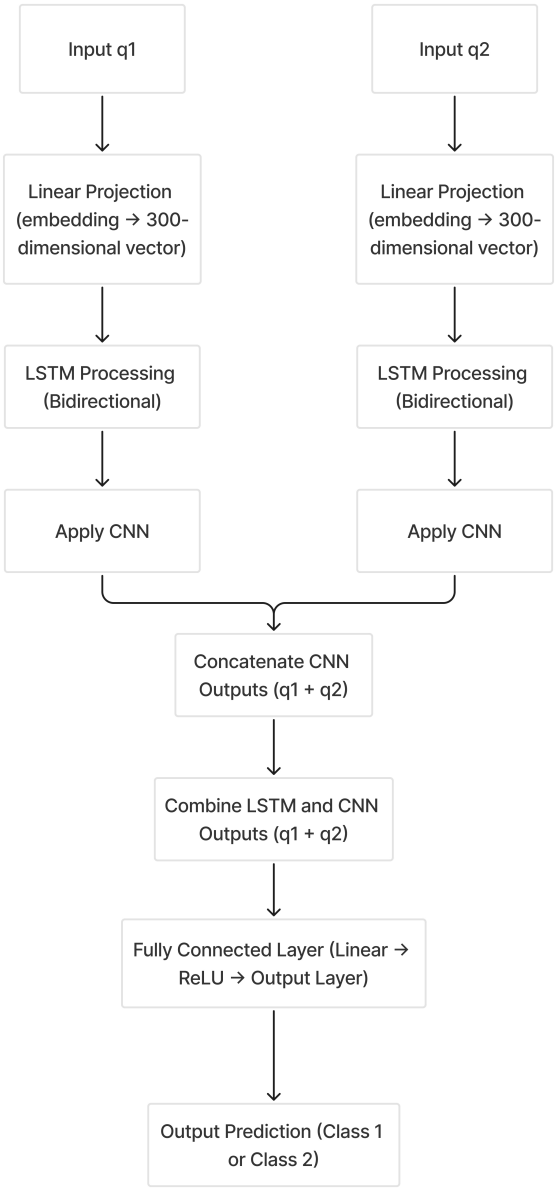

Fig. 4. Siamese LSTM-CNN Architecture.

E. Siamese LSTM with Attention

Enhances the SiameseLSTM with an attention mechanism to focus on the most critical parts of the sequence.

- Improves understanding of semantic relationships.
- Handles long sequences effectively.

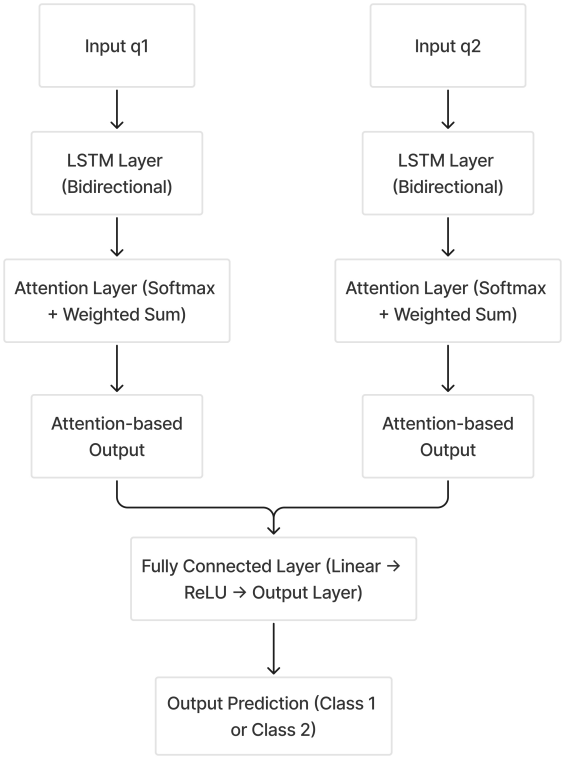

Fig. 5. Siamese LSTM with Attention Architecture.

F. Simple Encoder-Decoder Model

This model uses an encoder-decoder architecture with LSTM layers. The encoder processes the input sequence and passes its output to the decoder, which generates a sequence representation. The last hidden state of the decoder is used for classification.

- LSTM-based sequence processing.
- Relies only on the final hidden state of the decoder.

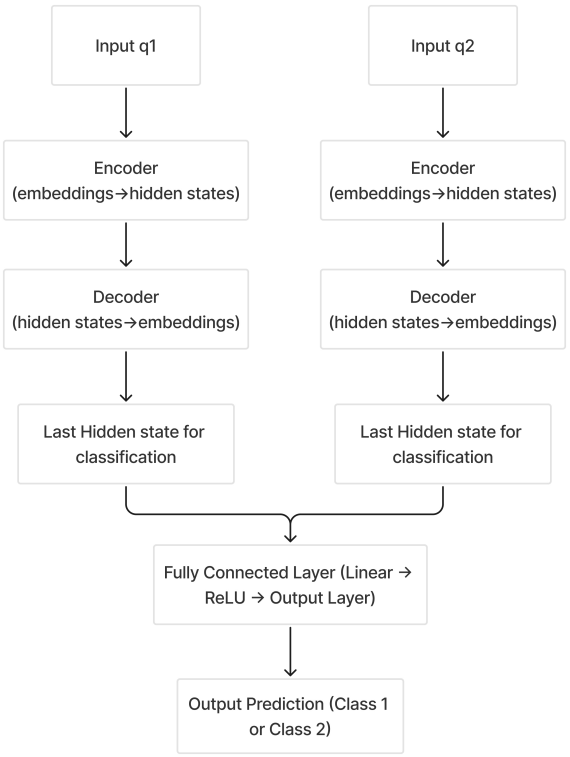

Fig. 6. Simple Encoder-Decoder Architecture.

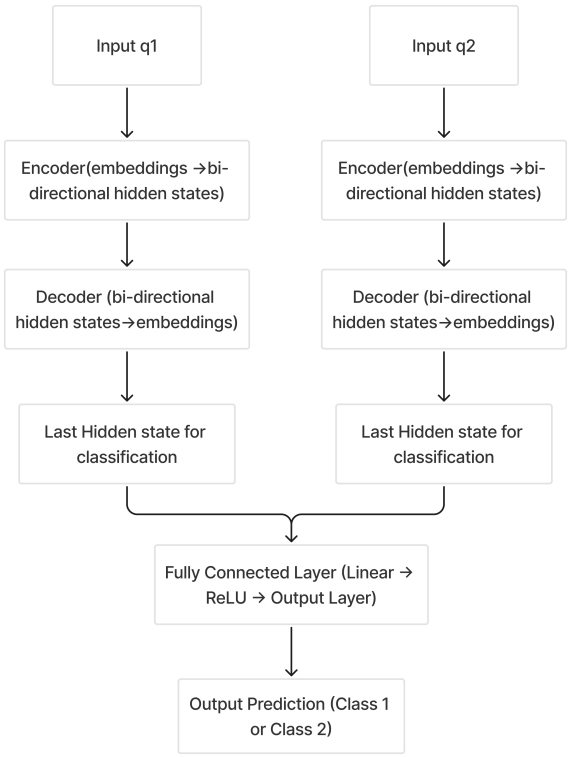

Fig. 7. Bidirectional LSTM Encoder-Decoder Architecture.

G. Bidirectional LSTM Encoder-Decoder (BiLSTMEncoderDecoder)

This model introduces bidirectional LSTM layers in both the encoder and decoder. Bidirectional LSTMs capture both past and future contexts.

- Stronger contextual modeling.
- Captures dependencies from both ends of the sequence.

H. Attention-based Encoder-Decoder (AttentionEncoderDecoder)

This model uses an attention mechanism in the decoder to focus on relevant parts of the input sequence.

- Focuses on the most relevant tokens in the input.
- Better handling of long sequences.

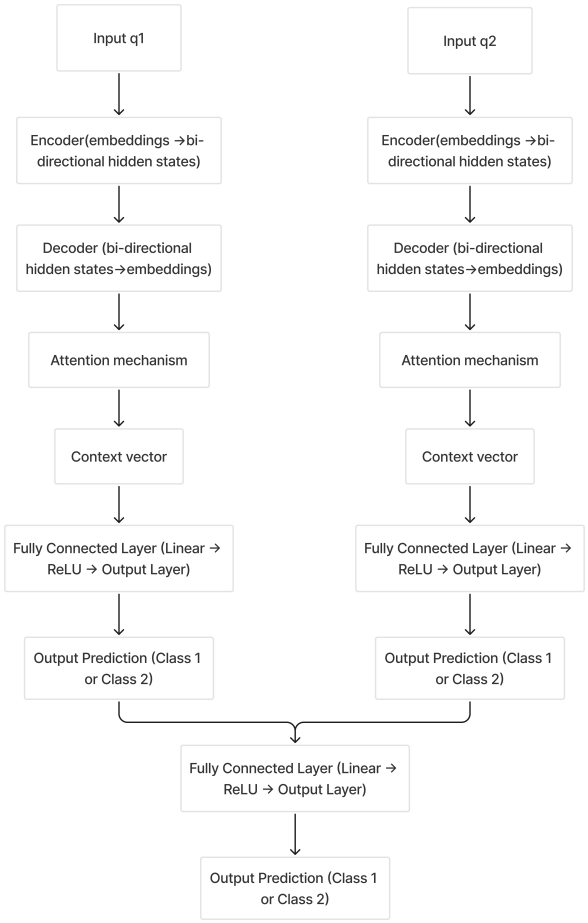

Fig. 8. Attention Encoder-Decoder Architecture.

I. Transformer Encoder-Decoder (TransformerEncoderDecoder)

The Transformer uses only attention mechanisms for scalability and efficiency, with no recurrent layers.

- Scalable and efficient for long sequences.
- Completely parallelizable due to self-attention mechanisms.

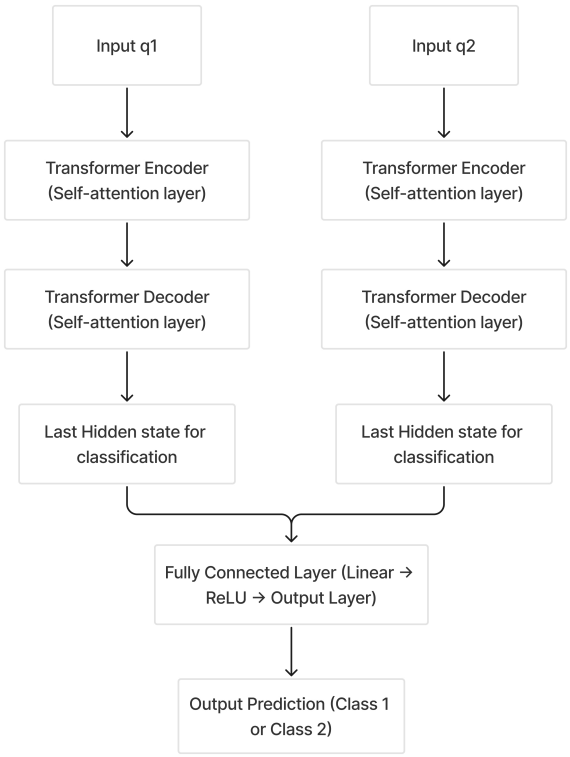

Fig. 9. Transformer Encoder-Decoder Architecture.

J. Ultimate Encoder-Decoder with Gated Residual Networks and Memory (UltimateEncoderDecoder)

This model combines Gated Residual Networks (GRN), Memory Networks, and Multi-Head Attention to create a highly expressive sequence-to-sequence architecture.

- Handles long-range dependencies effectively.
- High computational complexity due to advanced techniques.

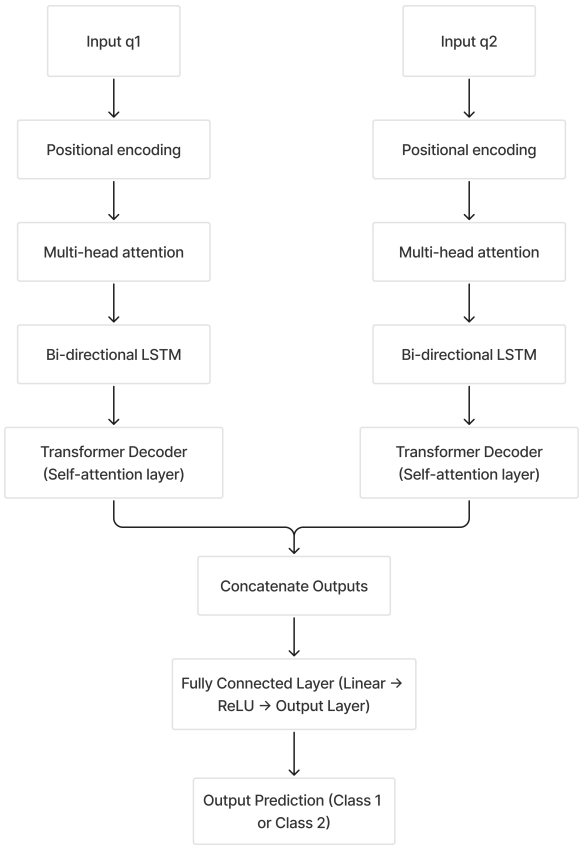

Fig. 10. Ultimate Encoder-Decoder Architecture.

IV. Results and Conclusion

The study evaluates diverse architectures for duplicate question detection. Models like Transformer-based and attention-enhanced systems excel at scalability and semantic understanding. Future work can explore optimized architectures for computational efficiency and real-world deployment.

All the code for this project can be found on GitHub. Click the link below to access the repository:

GitHub Repository

TABLE I
MODEL PERFORMANCE COMPARISON

Model	Accuracy (%)	Strengths
BaselineNN	72.21	Simple, fast implementation
SiameseCNN	81.53	Efficient local feature extraction
SiameseLSTM	74.65	Captures sequential dependencies
SiameseLSTM-CNN	78.88	Combines sequential and local features
SiameseLSTMWithAttention	80.70	Focuses on critical tokens
Simple Encoder-Decoder	75.45	Effective for moderate-length sequences
Bidirectional Encoder-Decoder	76.89	Rich contextual modeling
Attention Encoder-Decoder	79.09	Better long-sequence handling
TransformerEncoderDecoder	81.62	Scalable, efficient architecture
UltimateEncoderDecoder	73.28	Rich feature representations