ISSN: 2096-3246
Volume 58, Issue 01, 2026

OPTIMIZING QUESTION ANSWERING SYSTEMS WITH A HIGH-PERFORMANCE PIPELINED ARCHITECTURE INTEGRATING BERT/ROBERTA AND MULTI-STAGE SCORE CLASSIFICATION MECHANISMS

[1]Rejimoan R
Department of Computer Science and Engineering, Annamalai University, Chidambaram | rejimoan@gmail.com

[2]Gnanapriya B
Department of Computer Science and Engineering, Annamalai University, Chidambaram | priyamvatha.joey@gmail.com

[3]Jayasudha J S
Department of Computer Science, Central University of Kerala
jayasudhajs@gmail.com

Abstract:
Question answering models are designed to automatically generate accurate and contextually relevant answers from a dataset. These models leverage advanced techniques in natural language processing, such as transformer architectures like BERT and RoBERTa, to understand and extract the precise information needed to formulate responses. This work presents a comparative evaluation of BERT and RoBERTa models, focusing on key performance metrices such as Exact Match (EM), BiLingual Evaluation Understudy (BLEU), and F1 Score. Start word score and end word score determine the precise boundaries of the answer within a passage. These scores reflect the ability of the model and its accuracy is crucial for high EM and F1 Scores. RoBERTa's advanced architecture and fine-tuning processes enable it to more accurately identify these positions, resulting in more precise and contextually relevant answers. This study highlights RoBERTa's superior performance, with an EM of 75%, a BLEU of 80%, and an F1 of 87%, outperforming BERT, which achieved 70%, 75%, and 82% in the respective metrics. The findings of this study establish RoBERTa as the preferred model for question answering tasks, particularly in applications requiring high precision and exact answer identification. This research emphasizes the importance of start and end word selection in driving model performance and suggests areas for further refinement in question answering systems.

Keywords: Question Answering, Start Word, End Word, Natural Language Processing, Transformer Model.

1. Introduction

Question-Answering (QA) based on Natural Language Processing (NLP) have emerged as a pivotal area of research, driven by the need for accuracy and efficiency. Traditional QA systems have predominantly focused on generating single-word or short-phrase answers, which, while useful in certain contexts, fall short in scenarios requiring more detailed and contextually rich responses. The development of large-scale datasets like SQuAD 2.0 has catalysed significant advancements in QA systems by providing a benchmark for the

development and evaluation of new schemes [1]. SQuAD 2.0 introduces more complex questions, often necessitating the identification of whether a question is answerable based on the provided text. This dataset has become a cornerstone for training QA models, yet it predominantly features single-word answers, limiting its applicability in real-world scenarios where nuanced, sentence-level responses are required. As such, there's a persistent need to advance beyond the current capabilities of QA systems to deliver more comprehensive answers. Extractive question answering involves selecting a specific span of text from a given passage that directly answers a query [2]. The model identifies and extracts the relevant portion of the text that most accurately responds to the question posed. The process flow of general question-answer model is illustrated in Figure 1.

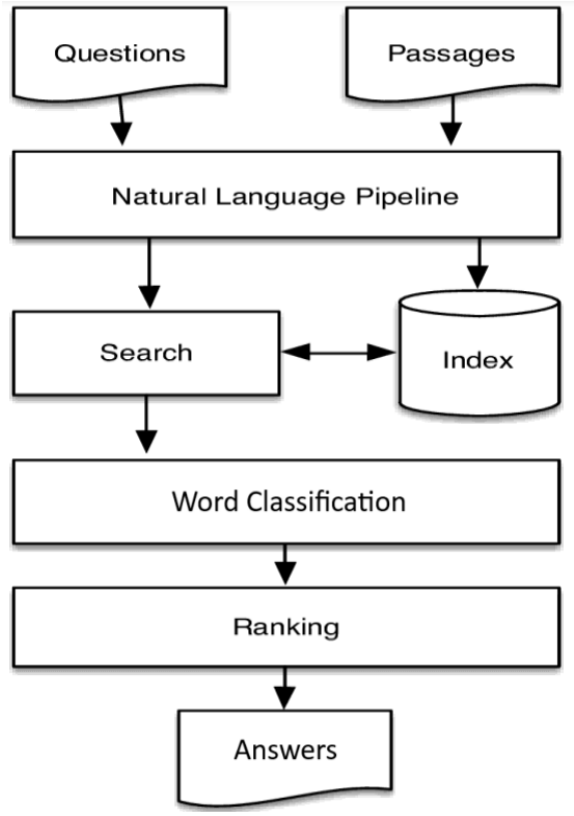

Figure 1: Extractive question-answer model

Bidirectional Encoder Representations from Transformers (BERT) has the ability to comprehend the context and it has revolutionized the way machines process and understand language. Despite these advancements, the inherent design of BERT and similar models still gravitates towards providing concise answers, often missing the mark in generating elaborate responses [3]. The limitations of single-word answers are starkly evident in practical applications. There is a gap in NLP research that underscores the necessity for QA systems that can produce longer, more contextually relevant answers, enhancing their utility across various domains. This work proposes a novel pipelined architecture that extends the capabilities of BERT by incorporating Deep Learning (DL) classifiers for start and end score classification. This approach aims to generate sentence-level answers by leveraging the strengths of BERT in understanding and processing natural language, while addressing its limitations in answer length and contextuality [4]. The development of this model involves several critical steps. Initially, we train the baseline BERT model using the SQuAD 2.0

dataset, which serves as the foundation for our QA system. The training process is designed to enhance BERT's ability to identify the span of text corresponding to the answer. Next, we introduce a DL classifier that employs a *softmax* activation function to identify the start and end words. This classifier enhances BERT's precision in pinpointing the relevant portion of the text, thereby facilitating the generation of longer answers.

The testing phase involves feeding customizable paragraphs and questions into the model, assessing its ability to adapt to various contexts and generate appropriate answers. By analysing the start and end word scores, we refine the model to improve its accuracy and reliability in producing sentence-level responses. By generating longer, contextually rich answers, our model can be deployed in diverse applications such as educational platforms, customer service, and virtual assistants [5]. These systems can provide detailed explanations, instructions, and solutions, enhancing user satisfaction and engagement. The integration of advanced DL techniques with transformer models like BERT/RoBERTa exemplifies the potential for hybrid approaches to overcome the constraints of individual methodologies.

This research presents a comprehensive exploration of a novel pipelined architecture designed to enhance QA systems. By leveraging the strengths of BERT and integrating DL classifiers for start and end score classification, we aim to generate sentence-level answers that are more aligned with real-world requirements. This approach not only addresses the limitations of single-word answers but also contributes to the advancements of intelligent and responsive models. The following sections will delve into the literature review, methodology, experimental results, and conclusions drawn from this research, providing a detailed account of our approach and its implications for the field of NLP.

2. Literature Review

Transformer models have revolutionized NLP by enabling more accurate and contextually aware understanding of text, leading to substantial improvements in QA tasks. Previous research has focused on enhancing model architectures, fine-tuning strategies, and evaluation metrics to optimize performance in various QA scenarios. This review explores the evolution of QA models by highlighting the key contributions, methodologies, and findings.

Seo *et al*. [6] developed a scalable document comprehension by utilizing phrase-indexed QA approach. This model ensures effective retrieval during the QA phase. The model enforced independency from encoders, which significantly enhanced the scalability and efficiency of the QA system. Tay *et al.* [7] presented a DL-QA model based on hyperbolic embeddings. The simplicity of HyperQA, requiring no feature engineering or complicated attention mechanisms, yet achieving competitive performance, is particularly noteworthy. Min *et al.* [8] incorporated essential sentences into the QA model, significantly reducing training and inference times. This approach not only enhances efficiency but also improves robustness to adversarial inputs, making the model more reliable in real-world applications.

Devlin *et al.* [9] introduced a transformative approach in NLP by utilizing bidirectional training of transformer models to achieve a deep understanding of language context. The ability of BERT to understand previous and following words in a sentence significantly enhances its comprehension capabilities. Park *et al.* [10] introduced S2-NET neural network for machine reading comprehension that uses self-matching attention mechanisms. This model processes the passage and question with attention and self-matching strategies to capture the intricate relationships between them, allowing for better comprehension and

answer extraction. Yu *et al.* [11] integrated Convolutional Neural Networks (CNNs) and self-attention mechanisms to improve reading comprehension. The model uses CNNs for local feature extraction and self-attention for capturing global dependencies, which improves the efficiency and accuracy of the QA system. This hybrid approach allows QANet to achieve faster training times compared to traditional NLP models.

Ashish [12] developed an attention-transformer with the ability to handle long-range dependencies and parallelize computations makes it particularly effective for QA tasks. DrQA is an open-domain QA system developed by Chen *et al.* [13], that reads Wikipedia articles to answer questions. The system consists of reader and retriever, which work together to find relevant articles and extract answers. DrQA's approach of leveraging large-scale knowledge sources like Wikipedia demonstrates the potential of combining information retrieval with DL for robust QA performance. Seo *et al.* [14] introduced an attention flow scheme that focuses on relevant areas of passages and the questions simultaneously. This approach enhanced the question-passage interaction, leading to more accurate answer extraction. BiDAF's innovative attention flow mechanism has influenced many subsequent models in the QA domain.

The SQuAD 2.0 dataset provides individual words as ground truths, which are then used by models trained on this dataset to generate predefined words as outputs. While these models excel within the confines of the dataset, real-world applications often require more than just a single word for an adequate response. In practice, answers need to be formulated as complete sentences to provide context and clarity, which is a significant extension beyond the dataset's original scope. This discrepancy highlights the need for models that can interpret and construct more complex and detailed responses suitable for practical use.

3. Methodology

To enhance the capability of answering systems to generate sentence-based responses, it's essential to develop a pipelined architecture that incorporates a transformer alongside multiple *softmax* classifiers. This design will facilitate the classification of both start and end scores, marking the borders of answers within a paragraph. By employing real-world text samples as training material, the system can learn to extract and formulate answers that are contextually accurate and complete [15]. The backbone of this architecture will be a pre-trained BERT model, which is leveraged for its powerful contextual embeddings and its proven effectiveness in understanding complex language patterns. This transformer-based architecture will enable more sophisticated interaction with text, where precise and comprehensive answers are required. The structure of the proposed question-answering model is depicted in Figure 2.

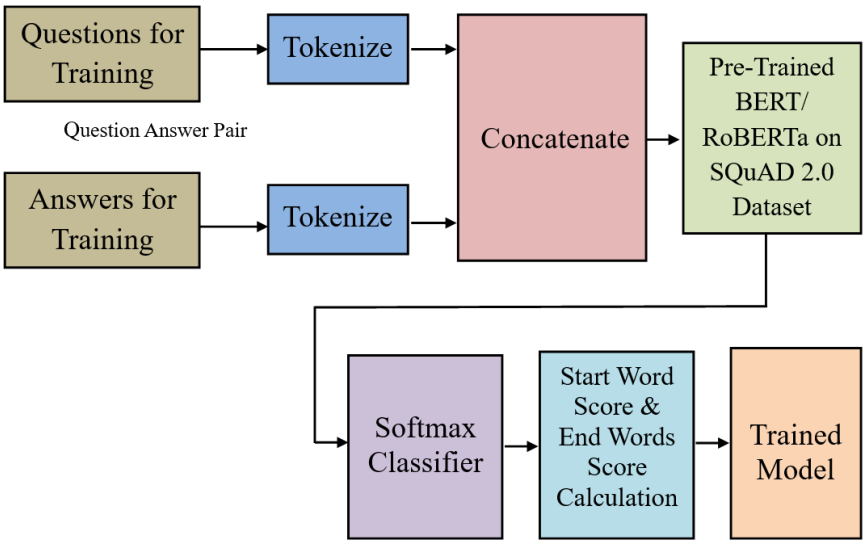

Figure 2: Architecture of proposed question answering model

3.1. *Questions and Answers for Training*

The foundational step in this architecture involves the preparation of train data (questions and corresponding answers). These pairs are utilized to train the proposed model in a supervised manner, allowing it to study the connection between questions and their respective answers within given text passages. The training data is typically sourced from large annotated datasets, such as the SQuAD 2.0 dataset [16]. This dataset contains passages from which questions are formulated, and corresponding answers are provided. Each entry in the dataset includes a Passage (P), a question (Q), and an answer (A). The answer (A) defined in Equation 3 is the text within P represented using Equation 1. The data is structured such that each Q represented by Equation 2 is linked to a specific A within P. The goal is to train the model such that given Q and P, it can accurately identify the span of A within P. Here, l is the token count in P, m is the token count of in Q. Here, I denote start index and j denotes end index of A within P.

$$P = \{p_1, p_2, ..., p_l\} \tag{1}$$

$$Q = \{q_1, q_2, ..., q_m\} \tag{2}$$

$$A = \{p_i, p_{i+1}, ..., p_j\} \tag{3}$$

3.2. Supervised Learning Objective

The objective of training is to reduce the error in predicting the correct start and end indices of A within P. For a given training example, the model should learn to predict i and j. The model uses these indices during the training phase to understand where A lies within P. This forms the basis for the model's ability to locate and extract the answer span accurately during inference [17]. By feeding a large number of such question-answer pairs into the model, it learns to generalize and accurately pinpoint answer spans for new, unseen questions posed against passages. This training process ensures that the model develops a nuanced

understanding of the context and can handle various forms of questions and answers, enabling it to perform well in real-world applications.

3.3. Tokenization

Tokenization involves breaking down of text into smaller units and mapping these units to numerical indices that the model can understand. The question Q and answer A are converted into a sequence of tokens. The passage P containing the context from which the answer is derived is also tokenized. Tokenization of the passage follows the same process as for the question and answer, converting the text to individual words [18]. Every token is assigned a distinct index corresponding to its position in the model's vocabulary. For more flexibility and to handle out-of-vocabulary words, sub-word tokenization methods known as Byte Pair Encoding (BPE) is used. For example, the word "machine" might be split into ["mach", "##ine"], where "##" indicates that "ine" is a sub-word. Special tokens are added to the tokenized sequence to help the model distinguish between different segments. [CLS] is a special token indicating the start of a sequence. [SEP] is a separator token used to distinguish different parts of the input. After tokenization, Q and A are combined into a single sequence as provided in Equation 4.

$$Input = [CLS; Q; SEP; P; SEP] \tag{4}$$

Let tokenize (T) be the tokenization function that maps a text T to a sequence of token indices. For given Q and P, the question token and passage tokens are represented using Equation 5 and Equation 6. The input sequence is formed by concatenating Q_{tokens} and P_{tokens} with special tokens as represented in Equation 7.

$$Q_{tokens} = \text{tokenize}(Q) \tag{5}$$

$$P_{tokens} = \text{tokenize}(P) \tag{6}$$

$$Input = [CLS] + Q_{tokens} + [SEP] + P_{tokens} + [SEP] \tag{7}$$

Tokenization ensures that the text is in a suitable format for the model to understand and process, which is essential for accurately identifying and extracting answers from the passage.

3.4. BERT Model

The pre-trained BERT model is a crucial component of the proposed architecture. Question - Answering experiments were performed using both BERT and RoBERTa models. BERT is a transformer-based models designed to understand the context of words in a sentence. It is pre-trained on SQuAD 2.0 dataset for QA tasks. The architecture of BERT model for question answering is depicted in Figure 3.

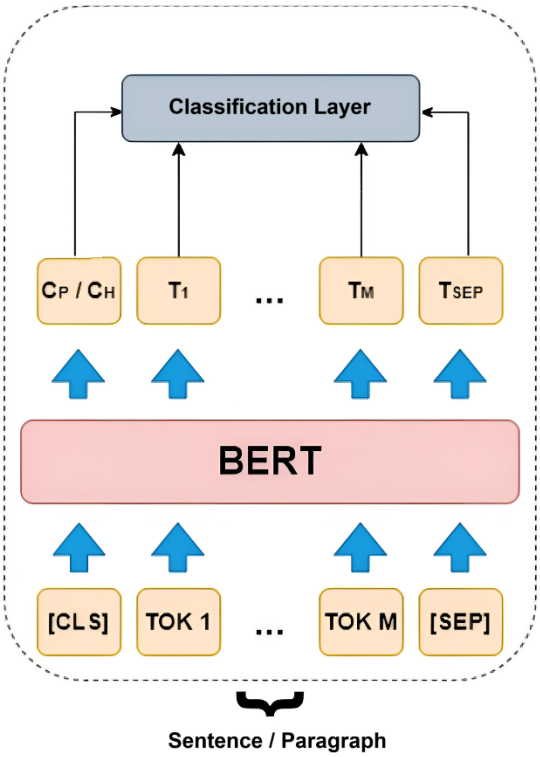

Figure 3: Architecture of BERT Model

The input to BERT is the tokenized sequence from the previous step defined by Equation 7. Each token in this sequence is converted into an embedding, which combines the token embeddings, segment embeddings, and position embeddings. Token embedding is the vector representations of each token. Segment embeddings distinguish whether a token is part of the question or the passage [19]. Position embeddings convey the position of a token within a sequence. The final input embeddings (E) represent the sum of these three embeddings. BERT is composed of several transformer layers with Feed-Forward Neural Networks (FFNNs) and multi-head self-attention. The embedding layer processes the input to produce the initial hidden representation, denoted as $H^{(0)}$, which is equal to E. The result from the l^{th} transformer layer ($H^{(l)}$) is defined by Equation (8).

$$H^{(l)} = Transformer\ Layer^{(l)}[\mathrm{H}^{(l-1)}] \tag{8}$$

Self-attention enables a model to assess the significance of each token in a sequence in relation to all other tokens. It calculates attention scores by analysing the connections between each token and every other token in the sequence. Self-attention captures dependencies between tokens, regardless of their positions, enabling the model to consider the context provided by the entire sequence when encoding information about any individual token. This capability is especially crucial in tasks where the meaning of a token can be influenced by tokens that are not adjacent to it, allowing the model to build richer and more context-aware representations. Attention scores are computed from Key (K), Query (Q) and value matrix (V) using Equation 9. These values are derived from the input embeddings. The dimension of K is represented as d_k.

$$Att(\mathrm{Q},\mathrm{K},\mathrm{V}) = V * \mathrm{softmax}\left(\frac{K^T Q}{\sqrt{d_k}}\right) \tag{9}$$

FFNN is applied independently to the output of self-attention. After self-attention computes the weighted context, the FFN processes this context individually for each token without considering other tokens. This allows the network to further transform the token representations based on learned patterns. Mathematically, the FFN operates as per Equation 10.

$$FFN(x) = \mathrm{Re}\,LU(xW_1 + b_1)W_2 + b_2 \tag{10}$$

Here, x represents the input to the FFN, which in this case is the output from the self-attention for a particular token. The weight matrices, W_1 and W_2, are learned during training. Here the bias vectors (b_1 and b_2) are also learned during training. The first linear transformation xW_1+b_1 adjusts the dimensions of the input and applies a bias. The ReLU activation function introduces non-linearity, which enables the network to capture complex relationships in the data [20]. The second linear transformation W_2 followed by the bias b_2 further refines the token's representation. The final output from this process is a transformed representation of the token that incorporates both its original context and the influences from other tokens in the sequence. The output H from BERT after processing through multiple layers of self-attention and FFNN is a set of contextual embeddings. These embeddings are rich in information, capturing both the local and global context for each token, making them highly effective for question answering.

When fine-tuning BERT on the SQuAD 2.0 dataset, additional classification layers are added to the model to enable it to predict the start and end words. To predict the start-position of the answer, a classification layer is added to BERT. This layer applies a linear transformation using a weight matrix W_s and a bias vector b_s to the hidden states H produced by BERT. The architecture of start token classifier is illustrated in Figure 4.

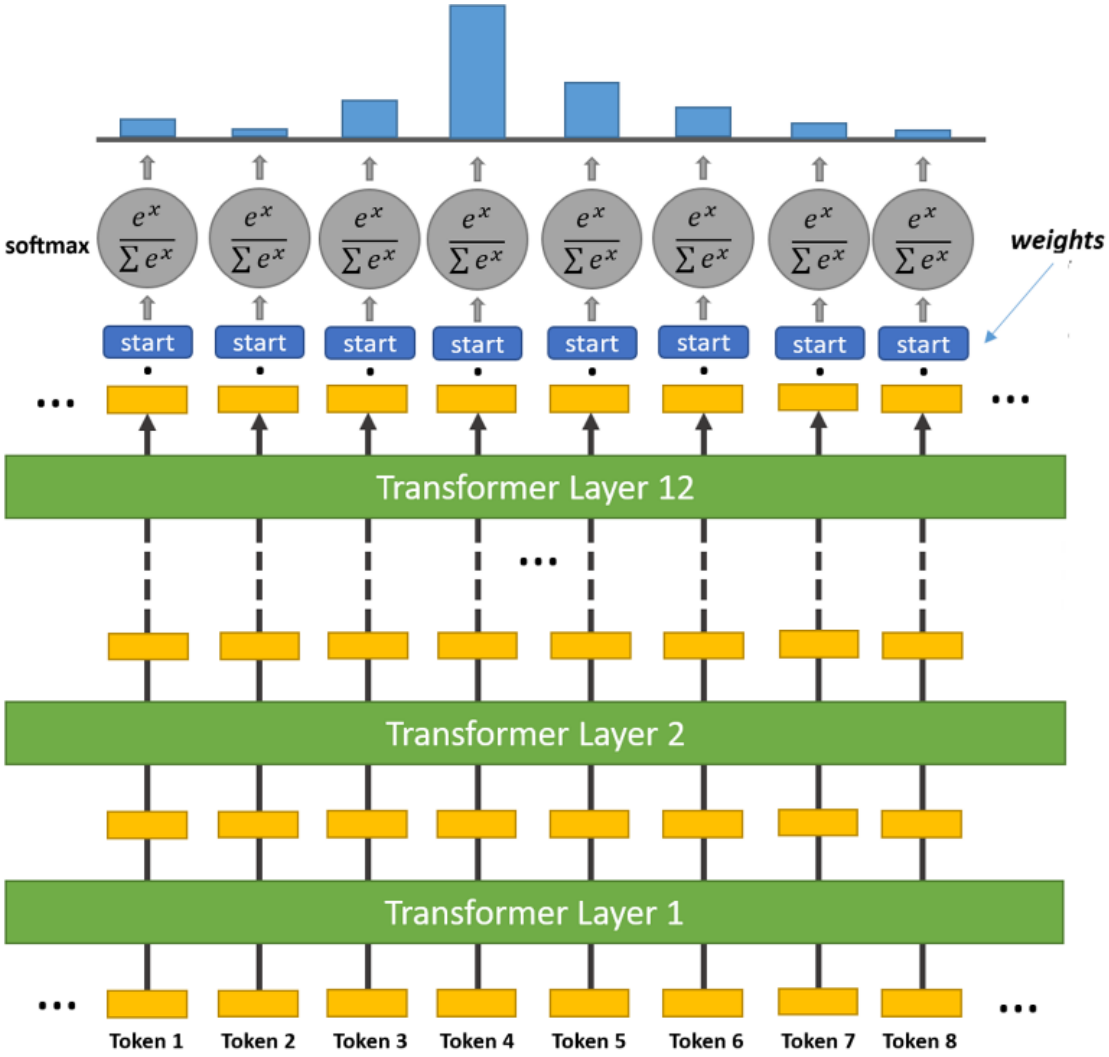

Figure 4: Start token classifier

The result of this linear transformation is applied to *softmax* function, to generate a probability value across all tokens in the context, representing the likelihood that each token marks the beginning of the answer. Mathematically, this operation is expressed using Equation 11. Here S is the vector of probabilities for the start position.

$$S = softmax(W_s H + b_s) \tag{11}$$

To predict the end-word, a classification layer is added to BERT. This layer applies a linear transformation using a weight matrix W_e and a bias vector b_e to the hidden states H produced by BERT. The architecture of end token classifier is illustrated in Figure 5.

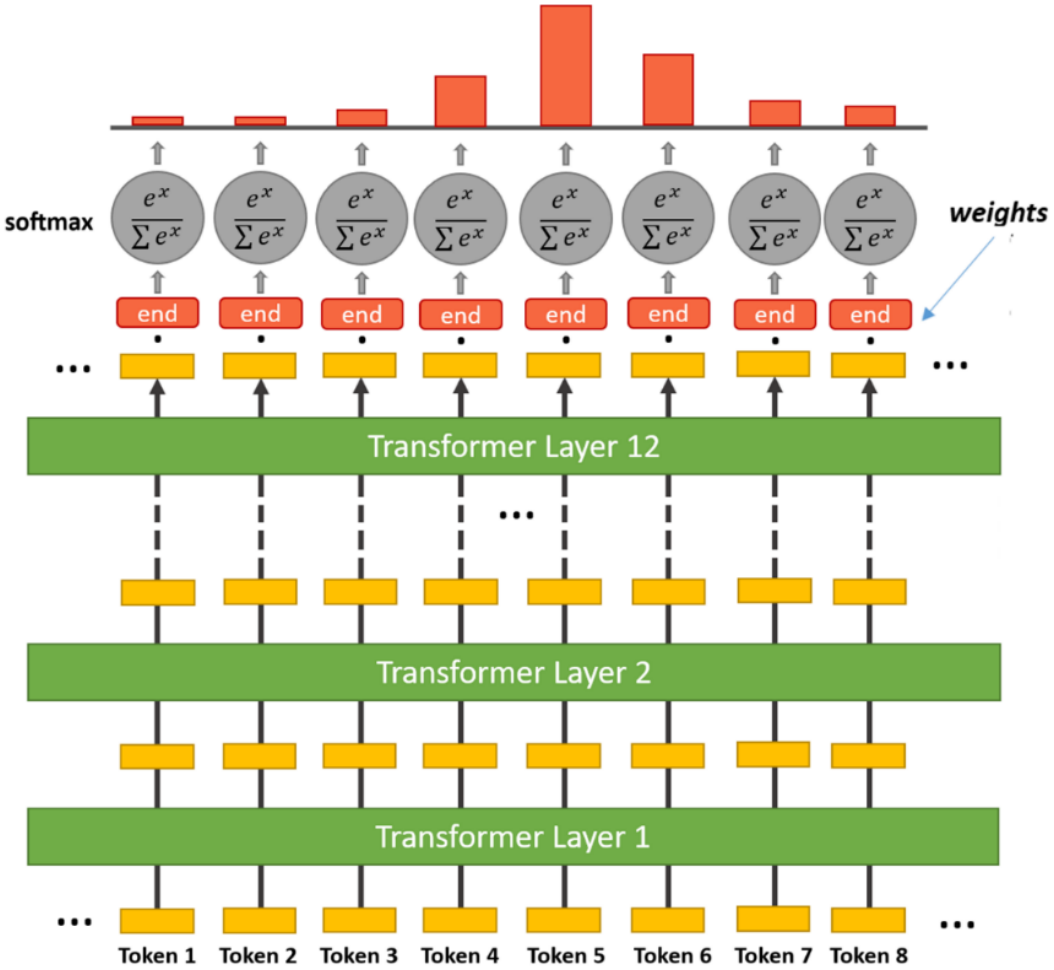

Figure 5: End token classifier

The output from this linear transformation is processed through a softmax function, creating a probability distribution over all tokens in the context, which reflects the likelihood of a token to become the end word. Mathematically, this operation is expressed using Equation 12. Here E is the vector of probabilities for the end position.

$$E = softmax(W_e H + b_e) \tag{12}$$

3.5. Training Phase

During training, the model's predictions for the start position are compared to the true start positions using cross-entropy loss. The loss for the start position is calculated using Equation 13. Here, y_i^{start} is a one-hot encoded vector representing the true start position, and s_i is the predicted probability for token i being the start.

$$L_{start} = -\sum_i y_i^{start} \log(s_i) \tag{13}$$

Similarly, the loss for the end position is calculated using Equation 14. Here, y_i^{end} is a one-hot encoded vector representing the true end position, and e_i is the predicted probability for token i being the end.

$$L_{end} = -\sum_i y_i^{end} \log(e_i) \tag{14}$$

The total loss used for training the model is the sum of the start and end position losses defined in Equation 15. This combined loss ensures that the model is simultaneously optimizing for both start and end words.

$$L = L_{start} + L_{end} \tag{15}$$

3.6. RoBERTa Model

Robustly optimized BERT (RoBERTa) approach) is an improved BERT that incorporates several advancements in its training methodology. These enhancements make RoBERTa more effective and robust for question-answering tasks. The architecture of RoBERTa model for question answering is depicted in Figure 6.

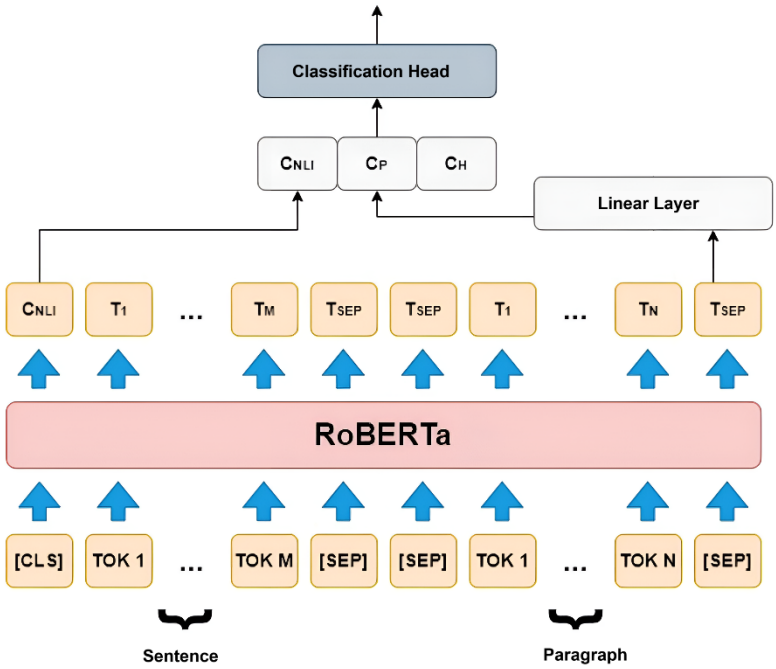

Figure 6: Architecture of RoBERTa Model

RoBERTa introduces dynamic masking, where the tokens to be masked are selected randomly during each epoch of training. This ensures that the model sees different masking patterns during training, enhancing its ability to generalize. For a given input sequence X = {x_1, x_2,……., x_n}, the masking process in RoBERTa can be represented using Equation (16).

$$M_t(X) = \{m_1, m_2, \ldots, m_n\} \tag{16}$$

Where, M_t is the masking function at training step t, and m_i is a binary mask indicator (1 if the token is masked, 0 otherwise). The masked input at step t can be represented using Equation 17.

$$\tilde{X}_t = \{(1-m_i)\cdot x_i + m_i \cdot [MASK]\} \tag{17}$$

RoBERTa focuses solely on the Masked Language Model (MLM) objective defined in Equation 18. This simplifies the training and allows the model to allocate more capacity to learning from the MLM task. Here $\tilde{X}_i$ is the masked input sequence with the i^{th} token masked, and x_i is the original token.

$$L_{MLM} = -\sum_{i=1}^{n} logP(x_i | \tilde{X}_i) \tag{18}$$

RoBERTa employs Byte-Pair Encoding (BPE), which allows for more efficient sub-word tokenization. The BPE algorithm operates by repeatedly merging the most frequent pairs of characters or sub-words in a sequence. Given a sequence of characters C = {c_1, c_2, ……., c_m}. BPE merges pairs (c_i, c_{i+1}) based on their frequency until the desired vocabulary size is reached. So, the merge operation can be represented using Equation 19. Here b_k are the resulting sub-words after applying BPE.

$$BPE(C) = \{b_1, b_2, \ldots, b_k\} \tag{19}$$

3.7. Testing Phase

During inference, the model leverages the weights learned during training to forecast the most likely start and end words within the given context. When a question and its corresponding context are input to the fine-tuned BERT model, it first generates a set of contextual embeddings for each token. These embeddings capture the nuanced relationships between the tokens in the context, influenced by the given question. The testing phase of the proposed question answering model is illustrated in Figure 7.

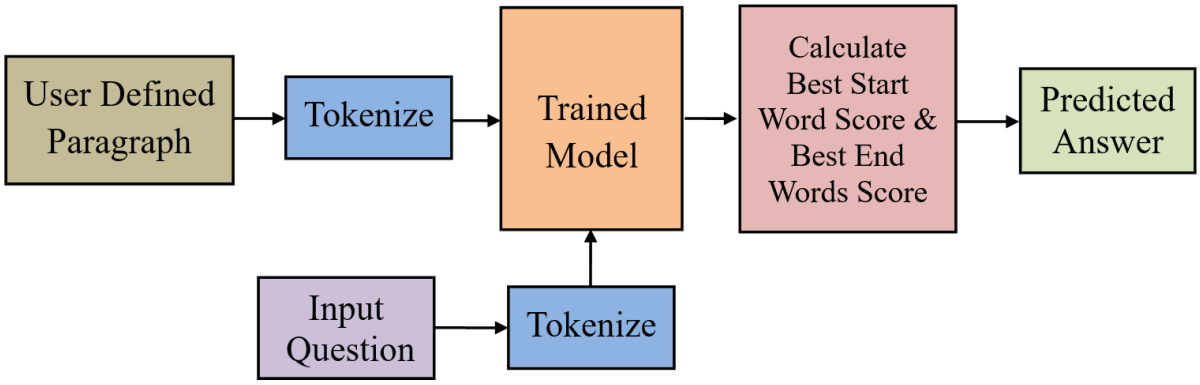

Figure 7: Training Phase

The contextual embeddings *H*, generated for each token in the sequence, are fed into the two classification layers: one for predicting the start word and another for predicting the end word. The model selects the token with the highest probability in the *S* distribution as the start of the answer. This is achieved by taking the index of the maximum value in the S vector as defined in Equation 20. Here, s_i is the probability assigned to token *i* being the start of the answer.

$$\hat{s} = \arg\max_i s_i \tag{20}$$

Similarly, the token with the highest probability in the *E* distribution is chosen as the predicted end of the answer. This is calculated using Equation 21. Here e_i is the probability assigned to token *i* being the end of the answer.

$$\hat{e} = \arg\max_i e_i \tag{21}$$

The model then combines the start and end positions ($\hat{s}$, $\hat{e}$) to extract the corresponding span of tokens from the context. This span is the final answer to the given question. Here $\hat{s}$ should ideally be less than or equal to $\hat{e}$, to form a valid span. After determining the most probable start and end positions, along with any necessary adjustments for no-answer scenarios, the model outputs the answer. If the start and end tokens correspond to meaningful text within the context, that text is returned as the answer. If a no-answer is determined, the model can return an indication.

4. Result and Discussion

The input paragraph provides the context from which the model generates its answers. Understanding this context is essential for interpreting the model's predictions and evaluation of performance. The input paragraph is a foundation for the model's classification process. By examining the paragraph, it is possible to assess whether the model is correctly identifying the relevant portions of the text to answer the question. The structure, complexity, and content of the paragraph all play significant roles in how well the model performs. Analysing the input paragraphs in cases where the model performed poorly could reveal specific challenges the model faces, such as dealing with multiple potential answers, distinguishing between relevant and irrelevant information, or interpreting indirect language. Recognizing how various paragraph structures affect model performance is crucial for improving the model's training process. For example, if the model struggles with certain types of paragraphs, introducing more similar examples into the training set could improve its ability to handle these cases. A sample input paragraph used for evaluation is illustrated in Figure 8.

```
print(wrapper.fill(test_paragraph))
```

```
Life insurance is a legally enforceable contract between two parties both of
whom are legally qualified to contract. It is therefore, necessary that the
terms and conditions of the agreement must be suitably documented in a manner
that would make it clear that both parties to the contract are of the same mind.
Ad-Idem means that both the parties understand the same thing in the same sense
or are of the same mind on the same subject. There must be consensus or Ad-Idem
between the parties to the contract. This is possible provided all the terms and
conditions, rights and duties - privileges and obligations are properly
documented in terms which can be clearly interpreted in a court of law.Between
two human beings sometime silence means an acceptance. But as the insurer is a
legal personality entitled to contract verbal discussion between parties to the
contract is not possible and hence there is a need for documentation. Insurance
is also a contract of utmost good faith and enforced only in the distant future.
It is therefore necessary that the declarations made by both the parties should
be put in black and white for future reference. Any suppression, willful and
material shall make the contract void. The insured, therefore, has a duty to
declare all that he knows about himself, his health, his financial status in
answering questions contained in the proposal form and other ancillary documents
which may be required by the insurer. Age is an important factor in deciding the
quantum of premium against a policy. Non-standard age proofs are those which are
comparatively less reliable and therefore the insurer accepts them with a pinch
of salt. In other words the insurer takes certain precautions before accepting
such age proofs as final. Proof of income is the document may become necessary
whenever the sum proposed is very high. Normally a sum proposed which is seven
to eight times of the declared income is acceptable for insurance. But proposals
do come to the insurer when the known source of income of the proposer is much
less compared to the amount of insurance desired. A service holder normally does
not face this problem as his sources of income are verifiable. In case of
business people, the assessed income is at times much less compared to what is a
desirable income for the amount of insurance desired. In such cases the insurer
at times calls for assessed income tax returns, or Chartered Accountant's
certificate etc.
```

Figure 8: Input paragraph for evaluation

The start word scores and end word scores represent the model's confidence in selecting particular words as the beginning or end of the correct answer. The start and end scores are plotted as a distribution over the entire paragraph. Peaks in these distributions indicate where the model believes the answer starts and ends. If the scores peak sharply around the correct start and end words, it suggests that the model has a strong understanding of the context and can precisely pinpoint the answer. The start word scores and end word scores generated by the model are depicted in Figure 9 and Figure 10 respectively.

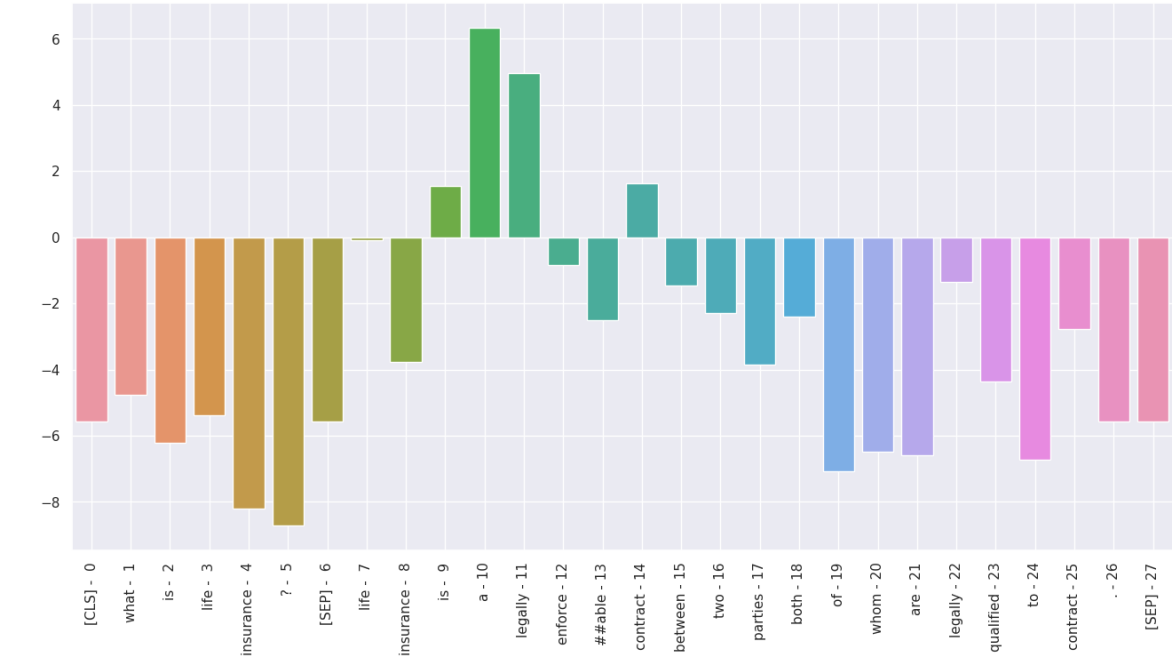

Figure 9: Start word scores

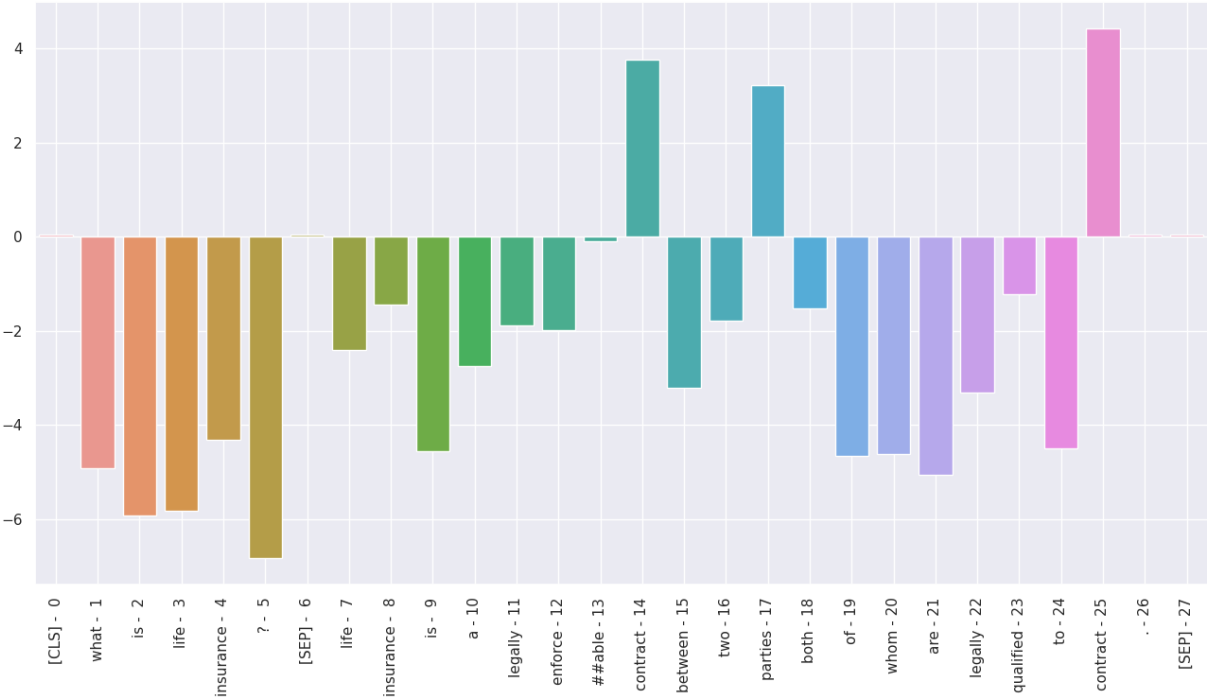

Figure 10: End word scores

High start and end word scores on the correct words reflect the model's capability to correctly interpret the question and locate the relevant information within the paragraph. This is a strong indicator of the model's effectiveness in question-answering tasks. In cases where the model assigns high scores to incorrect words, it's essential to analyse why this happens. Potential reasons could include ambiguous phrasing in the paragraph, synonyms or related terms that mislead the model, or complex sentence structures that challenge the model's ability to parse the context accurately.

The prediction visualization provides the answers generated by the model for particular input questions. Direct comparison between predicted answers and correct answers (ground truth) is essential for assessing the model's efficiency. The comparison can be quantitative, using metrics like Exact Match (EM), BLEU and F1 Score. EM is particularly stringent, requiring an exact word-for-word match with the ground truth, while F1 allows for partial credit based on overlapping words. If the predictions closely match the ground truth, both in terms of word choice and answer span, it indicates that the model is functioning well. Sample answers predicted by the proposed model are illustrated in Figure 11.

```
[ ] question = "what is meant by life insurance?"
    answer_question(question, test_paragraph)

    Query has 482 tokens.

    Answer: "a legally enforceable contract between two parties both of whom are legally qualified to contract"

    question = "what is meant by Ad-Idem?"
    answer_question(question, test_paragraph)

    Query has 484 tokens.

    Answer: "both the parties understand the same thing in the same sense or are of the same mind on the same subject"

[ ] question = "what is the significance of age?"
    answer_question(question, test_paragraph)

    Query has 482 tokens.

    Answer: "deciding the quantum of premium against a policy"
```

Figure 11: Answer predictions by the proposed model

EM is a strict metric that evaluates whether the predictions match the reference answer word-for-word. EM is particularly useful when the actual answer is a specific phrase or entity that the model needs to identify precisely. A higher EM score indicates that the model is accurate in producing the exact correct answer without any deviations. EM is computed using Equation 22.

$$EM = \frac{Correct\ Prediction\ Count}{Total\ Prediction\ Count} \tag{22}$$

The F1 score is particularly useful in QA when the answers can be partially correct, capturing overlay between predictions and references. F1 is calculated using Equation 23.

$$F1 = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{23}$$

The BLEU score assesses the quality of machine generated text by comparing it with the ground truth. BLEU is calculated up to 4-grams, meaning it considers sequences of up to four words, which helps capture the fluency and coherence of the generated text. A higher BLEU, closer to 1, specifies that the candidate text is very similar to the ground truth text, both in terms of content and structure. BLEU score is computed using Equation 24. Here, Brevity Penalty (BP) is applied to compensate for the smaller length of predicted answer compared to ground truth. The weight assigned to the n-gram precision (p_n) is denoted as w_n.

$$BLEU = BP \times exp\left(\sum_{n-1}^{N} w_n \log p_n\right) \tag{24}$$

The BERT model achieves an EM score of 70%, while RoBERTa model performs slightly better in terms of EM, achieving a score of 75%. A BLEU score of 75% suggests that BERT is quite good at generating answers that are close to the reference text, though there may be occasional differences in phrasing or word choice that reduce the score slightly. RoBERTa's BLEU score of 80% indicates that its generated answers closely match the reference text in terms of phrasing and word choice. The F1 score of 82% for BERT indicates that the model has a strong balance between precision and recall, effectively capturing relevant information while minimizing errors. The F1 score of 87% for RoBERTa is robust, showing that the model is even more effective at balancing precision and recall. The computation of performance metrices for input questions is illustrated in Figure 12.

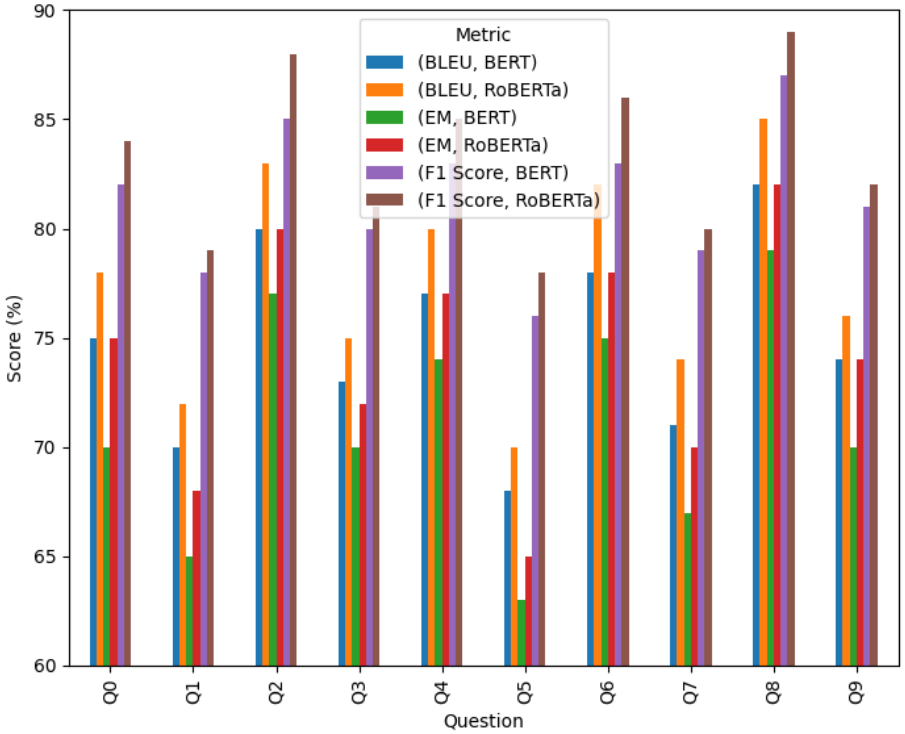

Figure 12: Performance matrices computed for the proposed models

RoBERTa's higher EM score (75% vs. BERT's 70%) suggests that RoBERTa is generally better at producing exact matches with the reference answers. This improvement can be attributed to RoBERTa's more extensive training and better handling of contextual nuances. The BLEU score for RoBERTa (80%) is slightly higher than that for BERT (75%), indicating that RoBERTa tends to generate answers that are more closely aligned with the reference text in terms of n-gram overlap. This suggests that RoBERTa is better at capturing the correct phrasing and structure of the answers. The F1 score for RoBERTa (87%) is also higher than BERT's (82%), reflecting RoBERTa's superior ability to balance precision and recall. The performance comparison of proposed question answering models with existing models is described in Table 1.

Table 1: Performance Comparison

Metric	BERT	RoBERTa	GPT-2	XLNet	ALBERT
Exact Match (EM) (%)	70	75	65	72	73

BLEU (%)	75	80	70	78	77
F1 Score (%)	82	87	80	85	86

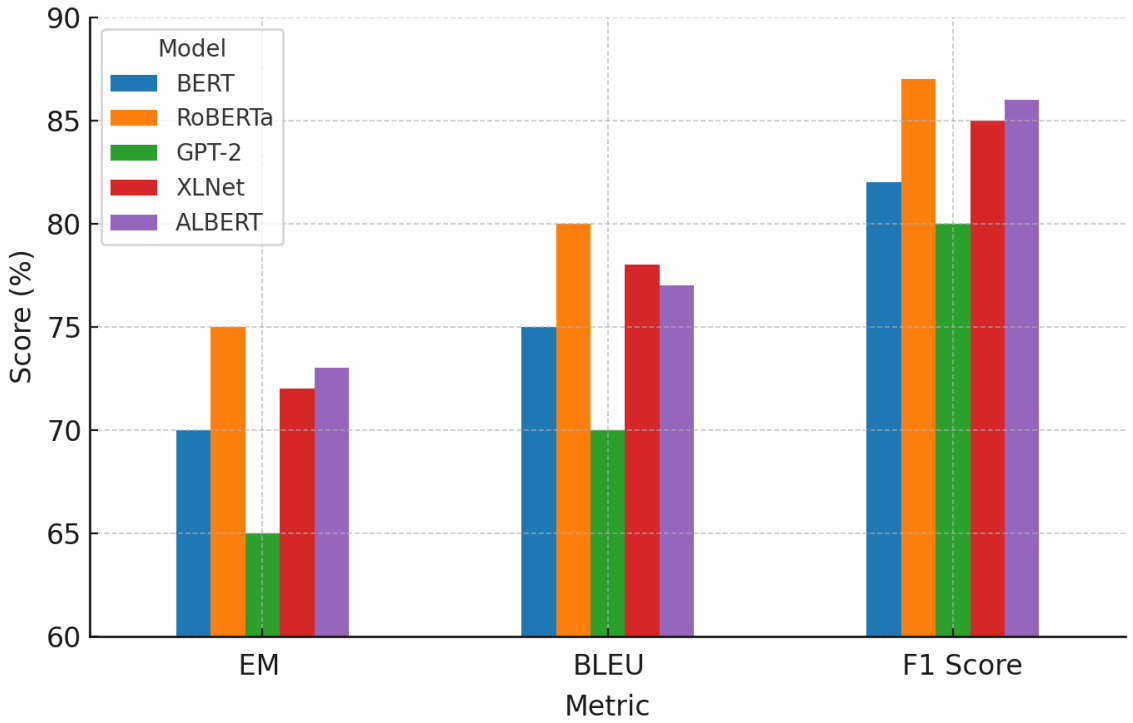

Figure 13: Model comparison based on metric scores

The performance comparison of proposed question answering models with existing models is illustrated in Figure 13. In this comparative study, the performance of five prominent natural language processing models—BERT, RoBERTa, GPT-2 [21], XLNet [22], and ALBERT [23]—was evaluated across three key metrics: EM, BLEU, and F1 Score. RoBERTa consistently outperformed the other models, achieving the highest scores across all metrics, with an EM score of 75%, a BLEU score of 80%, and an F1 Score of 87%. This indicates RoBERTa's superior ability to produce accurate, fluent, and contextually relevant text. ALBERT and XLNet also demonstrated strong performance, with both models closely trailing RoBERTa, making them reliable alternatives for tasks requiring precision and recall. BERT, while robust, showed slightly lower scores, reflecting its older architecture compared to its successors. GPT-2, although effective in generating coherent text, lagged behind in precision, with lower EM and BLEU scores, indicating that it may prioritize fluency over exact matches. Overall, this analysis underscores the advancements in newer models like RoBERTa, ALBERT, and XLNet, which offer significant improvements in performance, making them better suited for tasks requiring high accuracy and n-gram fidelity. The model comparison based on metric trends is illustrated in Figure 14.

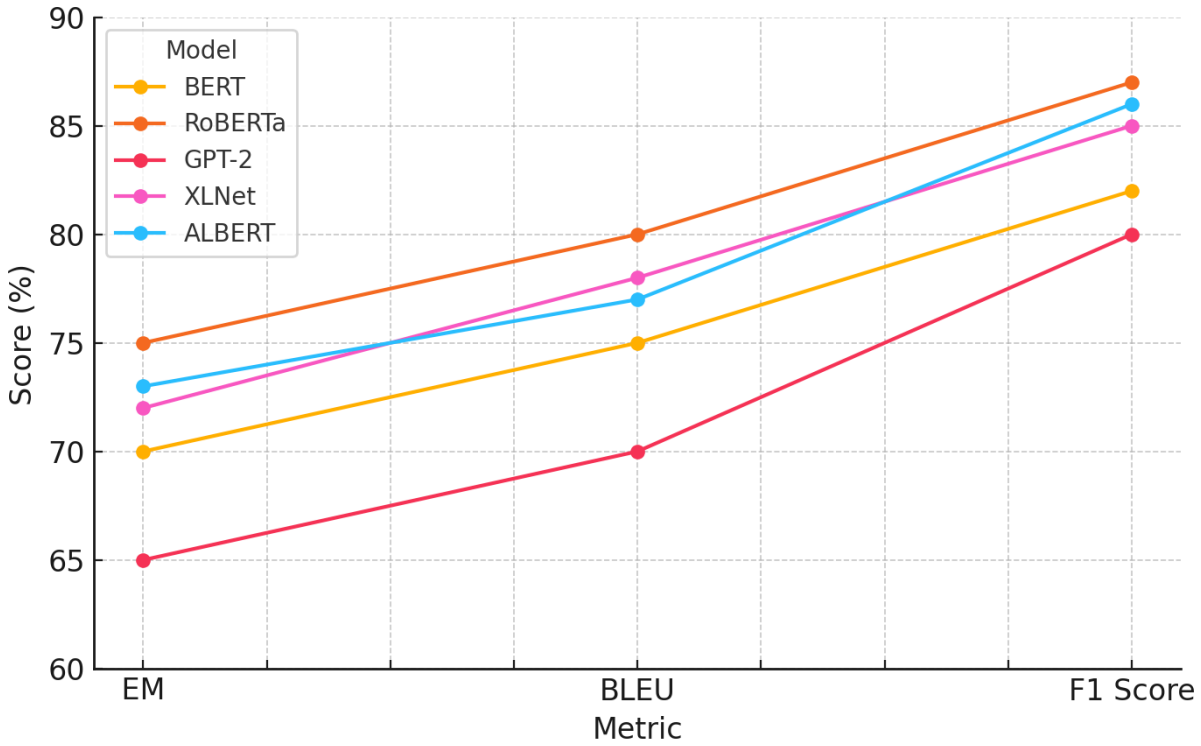

Figure 14: Model comparison based on metric trends

5. Conclusion

This study proposed and assessed BERT and RoBERTa based models for the task of question answering, focusing on key performance metrics such as EM, BLEU, and F1-score. A critical aspect of this performance is the accuracy of the start word score and end word score, which determine the precise boundaries of the answer within the passage. RoBERTa demonstrated superior performance, with an EM score of 75%, a BLEU score of 80%, and an F1 Score of 87%, significantly outperforming BERT, which achieved an EM score of 70%, a BLEU score of 75%, and an F1 Score of 82%. These scores, representing the confidence of model in predicting the start and end positions of the answer, are crucial for achieving high EM and F1 Scores. RoBERTa's optimized architecture and fine-tuning allow it to more accurately identify these positions, leading to more precise and contextually appropriate answers. This study underscores RoBERTa's effectiveness in question answering tasks, making it the preferred model for achieving higher accuracy and overall performance, particularly in tasks where the exact identification of answer spans is critical. Future work may focus on further refining these models to enhance their ability to handle even more complex questions.

References

1. Ashraf, N., Siddiqui, M. H. F., Sidorov, G., & Gelbukh, A. (2021). Transformer-based extractive social media question answering on TweetQA. Computacion y Sistemas, 25(1), 23-32.

2. Pandey, A. K., & Roy, S. S. (2024). Extractive Question Answering over Ancient scriptures Texts using Generative AI and Natural Language Processing Techniques. IEEE Access, 12, 53420-53431.

3. Yoon, W., Jackson, R., Lagerberg, A., & Kang, J. (2022). Sequence tagging for biomedical extractive question answering. Bioinformatics, 38(15), 3794-3801.
4. Krishnan, A., Sriram, S. R., Ganesan, B. V. R., & Sridhar, S. (2023). An Extractive Question Answering System for the Tamil Language. Advances in Science and Technology, 124, 312-319.
5. Keymanesh, Moniba, Micha Elsner, and Srinivasan Parthasarathy. "Privacy policy question answering assistant: A query-guided extractive summarization approach." arXiv preprint arXiv:2109.14638 (2021).
6. Seo, M., Kwiatkowski, T., Parikh, A. P., Farhadi, A., & Hajishirzi, H. (2018). Phrase-indexed question answering: A new challenge for scalable document comprehension. arXiv preprint arXiv:1804.07726.
7. Tay, Y., Tuan, L. A., & Hui, S. C. (2017). Hyperqa: Hyperbolic embeddings for fast and efficient ranking of question answer pairs. *arXiv preprint arXiv:1707.07847*.
8. Min, S., Zhong, V., Socher, R., & Xiong, C. (2018). Efficient and robust question answering from minimal context over documents. *arXiv preprint arXiv:1805.08092*.
9. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
10. Park, C., Lee, C., Hong, L., Hwang, Y., Yoo, T., Jang, J., and Kim, H. K. (2019). S2-Net: Machine reading comprehension with SRU-based self-matching networks. Etri Journal, 41(3), 371-382.
11. Yu, A. W., Dohan, D., Luong, M. T., Zhao, R., Chen, K., Norouzi, M., & Le, Q. V. (2018). QANet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.
12. Ashish, V. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*, I.
13. Chen, D., Fisch, A., Weston, J., & Bordes, A. (2017). Reading Wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.
14. Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.
15. Seonwoo, Y., Kim, J. H., Ha, J. W., & Oh, A. (2020). Context-aware answer extraction in question answering. *arXiv preprint arXiv:2011.02687*.
16. Zhang, Y., Xu, G., Wang, Y., Lin, D., Li, F., Wu, C., and Huang, T. (2020). A question answering-based framework for one-step event argument extraction. *IEEE Access*, *8*, 65420-65431.
17. Sovrano, F., Palmirani, M., & Vitali, F. (2020). Legal knowledge extraction for knowledge graph-based question-answering, *Legal knowledge and information systems* (pp. 143-153). IOS Press.
18. Dhandapani, A., & Vadivel, V. (2021). Question answering system over semantic web. *IEEE Access*, *9*, 46900-46910.
19. Sarrouti, M., & El Alaoui, S. O. (2020). SemBioNLQA: A semantic biomedical question answering system for retrieving exact and ideal answers to natural language questions. *Artificial intelligence in medicine*, *102*, 101767.
20. Nassiri, K., & Akhloufi, M. (2023). Transformer models used for text-based question answering systems. *Applied Intelligence*, *53*(9), 10602-10635.

21. Tsai, D. C., Chang, W., & Yang, S. (2021, November). Short answer questions generation by Fine-Tuning BERT and GPT-2. In *Proceedings of the 29th International Conference on Computers in Education Conference, ICCE* (Vol. 64).
22. Raza, S., Schwartz, B., & Rosella, L. C. (2022). CoQUAD: a COVID-19 question answering dataset system, facilitating research, benchmarking, and practice. *BMC bioinformatics*, *23*(1), 210.
23. Alzubi, J. A., Jain, R., Singh, A., Parwekar, P., & Gupta, M. (2023). COBERT: COVID-19 question answering system using BERT. *Arabian journal for science and engineering*, *48*(8), 11003-11013.