Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

CQ-Transformer: A Query Decomposition Framework with ALBERT-Based Semantic Matching for Complex Extractive Question Answering

Hasnain abdullah

hassnaincomsats@gmail.com

Alhamd Islamic University

Abd Ur Rahman

COMSATS University Islamabad

Riaz Ahmed

Alhamd Islamic University

Muhammad Awais

Alhamd Islamic University

Ali Hamza

National University of Sciences and Technology

Research Article

Keywords: Extractive Question Answering, Query Decomposition, Semantic Similarity Matching, RoBERTa, XLNet, SQuAD 2.0, Reading Comprehension, Natural Language Processing

Posted Date: June 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10088904/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

CQ-Transformer: A Query Decomposition Framework with ALBERT-Based Semantic Matching for Complex Extractive Question Answering

Hasnain Abdullah[1], Abd Ur Rahman[2], Riaz Ahmed[1], Muhammad Awais[1], Ali Hamza[3]

1Department of Computer Science, Alhamd Islamic University, Quetta, Pakistan,

2Department of Computer Science, COMSATS University, Islamabad, Pakistan

3Department of Computer Science National University of Sciences and Technology (NUST), Islamabad, Pakistan

Corresponding Author:

Hasnain Abdullah: hassnaincomsats@gmail.com

Abstract

Question Answering (QA) systems play an indispensable role in Natural Language Processing (NLP) by enabling automated extraction of specific answers from large-scale text corpora. Nonetheless, existing transformer-based approaches, including BERT and its derivatives, encounter significant limitations when processing complex multi-category queries and long-range contextual dependencies, frequently resulting in ambiguous or incomplete answer extraction. This paper proposes a hybrid QA framework that incorporates RoBERTa and XLNet with a query decomposition mechanism and ALBERT-based semantic similarity matching. A delimiter-based split function decomposes multi-category queries into discrete sub-queries, which are independently processed through pre-trained encoders after sentence-level relevance filtering via ALBERT; the extracted answer spans are subsequently aggregated into a unified response. Evaluated on SQuAD 2.0, the proposed RoBERTa-CQ variant achieves an F1 score of 87.95% and Exact Match (EM) of 84.96%, significantly outperforming baseline BERT (F1: 66.36%, EM: 57.85%), standard RoBERTa (F1: 82.58%, EM: 76.58%), and XLNet (F1: 68.14%, EM: 65.45%). The XLNet-CQ variant attains an F1 of 77.49% and EM of 74.79%. Qualitative analysis confirms effective resolution of multi-category queries and uncertainties where baselines fail entirely. The proposed framework demonstrates a scalable, high-performance solution for real-world extractive QA tasks.

Key Words

Extractive Question Answering, Query Decomposition, Semantic Similarity Matching, RoBERTa, XLNet, SQuAD 2.0, Reading Comprehension, Natural Language Processing.

1.Introduction

QA systems represent a fundamental task within the area of NLP. In the current digital era, the exponential growth of information has created a pressing need for automated systems capable of accurately retrieving relevant answers from large-scale text corpora. Although web search engines assist users in navigating this information, their utility is limited to returning document locations rather than precise answers. This limitation motivates the development of automated QA systems capable of directly extracting relevant answers from text [1].

According to [2], QA systems are classified based on the domain into two types: open-oriented QA systems and restricted-oriented QA systems. In an open-oriented QA system, there are no restrictions on domain topics, and it can accept questions on any topic, but in a restricted-oriented QA system, it can only accept questions within a specific domain.

The QA system is categorized based on the types of questions it is trying to answer. There are some popular applications in the open-oriented QA system domain, including Apple's Siri and IBM's Watson [3]. In recent years, search engines such as Google and virtual assistant such as Siri, Alexa, and Cortana have significantly improved in providing direct answers to user queries [4]. Generally, we have more applications in areas of QA systems such as medical diagnosis and health care[5], support [6], e-learning [7], and language translation [8].

In the QA system, several challenges and issues arise, including ambiguities that can lead to the retrieval of irrelevant answers [9], inconsistent responses [10], and common sense-based reasoning. Due to the limited availability of well-structured data, machines are facing difficulties in acquiring common sense knowledge [11].

In paper [1], there are different types of questions, including factoid questions (what, when, where, and who), simple reasoning (one-hop why and who), complex reasoning (multi-hop why and how), fusion list (compiled from different sources), interactive contexts (follow-up questions), and speculative questions (questions requiring a speculative answer). A study revealed that the QA system process is divided into three phases: The first phase involves question analysis, which the system looks for to understand the question. In the second phase, find contextual information to answer the question. Finally, the third phase, answer extraction, focuses on finding the most appropriate answer from the given context. The general process of answer span extraction is in Figure 1.

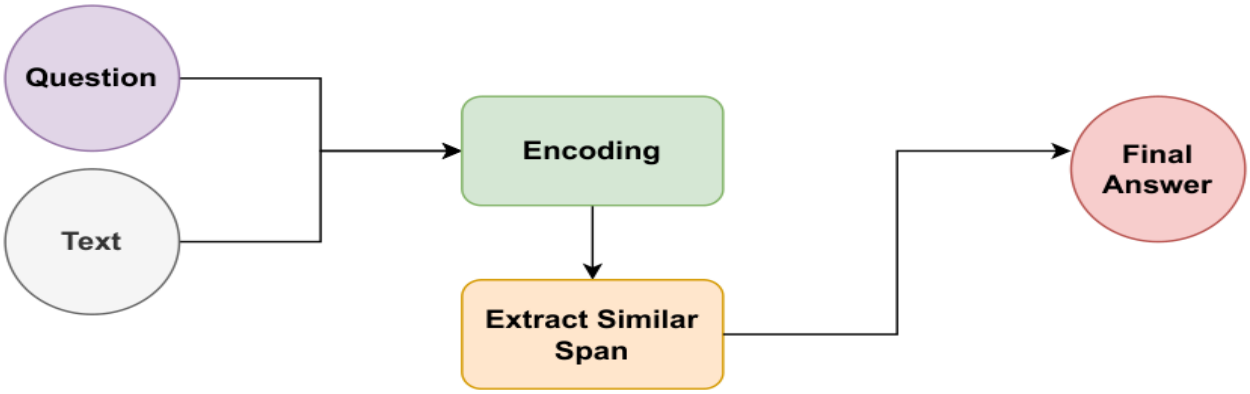


Figure 1: Answer Extraction Process

1.1 Machine Reading Comprehension

Machine Reading Comprehension (MRC) is essential in NLP. In this typical scenario, the procedure is to retrieve the answer to a question from a document [12]. MRC is a task that enables machines to understand natural language and answer questions based on a given context [13]. It facilitates meaningful interaction between humans and machines, as demonstrated in Figure 2, where a search engine extracts relevant answers from web pages using MRC techniques.

The first MRC model was developed in 1977 and its name is the QUALM system [14]. It is a small-scale and restricted domain so it is not used widely. The research was stopped from 1980 to 1990 [15]. After the MRC dataset was released, it contained stories and five "wh" (where, why, how, when, who) questions research interest in the MRC task increased significantly. At that time MRC used rule-based or machine learning-based to solve its task [13].

In the paper [15], researchers proposed a bag-of-words technique that passes question and context as sets of words and then selects words for answers that are the same in both question and context. [16], designed a rule-based MRC system to use different heuristic rules with morphological analysis such as semantic class tagging, entity recognition, and part-of-speech tagging for "wh" questions. [17], used machine learning methods and self-supervised learning for machine reading. However, those methods had some limitations such as not handling long dependencies, needing human effort, not extracting contextual information, and performance issues due to large-scale datasets [13].



Figure 2: Answer Extraction Example

The MRC task has changed since 2015. MRC is based on deep learning, also called neural machine reading comprehension. It handles contextual problems and different large-scale datasets such as MS MARCO [18], CNN & Daily Mail [19], and Stanford Question-Answering Dataset [20]. It is used with deep neural architecture to evaluate the MRC performance. MRC tasks are categorized into four different groups based on the type of answer they require [18]. Which are cloze style [19], multiple choice [21], span prediction [22], and free-form answer [23].

1.2 Cloze Tests

Cloze tests also referred to as gap-filling test, are commonly used in language proficiency assessments. This test is used to evaluate how well a machine understands natural language. In this test, some words are masked from the passage. To answer this question first ask to fill in the masked words and then extract the answer it is optional.

1.3 Multiple Choice

Multiple choice is another MRC task format, inspired by language proficiency examinations. In this task, the model is provided with a question, a context passage, and multiple answer options, from which the most relevant answer is selected. The multiple-choice answer is more flexible as compared to the cloze test.

1.4 Span Extraction

Cloze test and multiple choice can measure the ability of the machine to understand natural language, but there are some issues in those tasks. Words are not sufficient to answer the question. It needs a complete sentence for the answer. That issue is handled by the answer extraction MRC task. Given a question and context as input, the model is required to extract the relevant answer by identifying the start and end positions of the answer span within the context.

1.5 Free Answering

As compared cloze test and multiple-choice MRC task, the span extraction is more flexible but it is still not enough yet because it is unrealistic. Free-form answering requires collecting and synthesizing information from multiple contexts to generate a comprehensive response. Among four MRC tasks free answering is more complicated but it is suitable for real application scenarios.

1.6 Research Contributions

1. This work presents a pre-trained model-based QA framework incorporating RoBERTa and XLNet to address ambiguity arising from complex queries and long-range dependencies.
2. A Split function is introduced to decompose complex multi-category queries into single-category sub-queries, whereas prior studies processed queries without such segmentation.
3. An ALBERT-based semantic similarity module is introduced to identify the most relevant sentence for each query category, replacing conventional approaches such as cosine similarity and LSTM-based matching.

The remainder of this paper consists of 6 sections, namely Section 2"Literature review", Section 3 " System Model and Proposed Methodology ", Section 4 "Experimental setting ", Section 5 "Results evaluation and Performance analysis", Section 6 "Conclusion and Future Work".

2. Literature Review

The existing literature on reading comprehension QA is broadly categorized into two categories: single span extraction QA and multi-span extraction QA.

2.1 Single-based Span Extraction QA

Single span extraction is a QA approach that identifies a single contiguous answer span by predicting its start and end positions within the context.

The study [24], thoroughly examines the appropriateness of unsupervised representation learning techniques (SciBERT, Bio-SentVec, and BioBERT) for biomedical QA systems. They propose a novel pre-training task to enhance unsupervised representations by reasoning about biomedical entities in context through the introduction of context corruption and restoration. This de-noising task improves model performance by learning from vast amounts of unlabeled biomedical text. The study employs distinct models for different question types, using BioBERT or SciBERT as base models and fine-tuning them with specific datasets. Furthermore, the authors leverage the BioSentVec model to compute similarity scores for question context and predicted answers, which are integrated into BioBERT's predicted scores for final predictions. For yes/no questions, a three-layer neural network was used for classification. However, this approach is limited to biomedical de-noising tasks and achieves accuracy below 0.5 on list-type questions requiring complex reasoning.

The study [25], addresses the challenges of understanding Chinese sentences and the scarcity of suitable QA datasets. It employs convolutional neural networks, long and short-term memory networks, contention mechanisms, and attention mechanisms to analyze sentence information. The model is evaluated on the CCKS2018 dataset and a self-created Chinese QA dataset for an inverted pendulum experiment. The results demonstrate the model's analytical capabilities on Chinese text. The concept of question-pair matching and the creation of the inverted pendulum dataset lays the groundwork for future intelligent QA research. However, the model has limitations in domain expansion, dataset scope, and practical real-world applications that are yet to be realized.

The study [26], concentrated on answer span extraction within the QA task. It offers a comprehensive comparison of frequently employed span extraction techniques, emphasizing the specific advantages and limitations of each. Drawing inspiration from Faster R-CNN, the researchers introduce a novel span extraction method. Experimental results demonstrate that their proposed approach outperforms existing span extraction techniques in both English and Chinese MRC tasks. It needs a custom decoder that is larger in scale and shows increased complexity, especially in the case of emerging model architectures like transformer-based encoders supplemented with an output layer.

The study [27], combined deep learning techniques with knowledge graphs to produce two types of responses: straightforward natural language answers and lists of natural language responses. While a sequence-to-sequence model was employed to process simple answers, knowledge graphs played a pivotal role in deriving related responses, departing from the conventional approach of providing a single answer. The fusion of deep learning and knowledge graphs significantly bolstered the quality of responses. Additionally, the study implemented a QA system, utilizing a manually curated dataset of Vietnamese tourism, to confirm its effectiveness by using LSTM. However, certain limitations were encountered, such as the use of a relatively small dataset and the manual processes involved in extracting binary templates and constructing knowledge graphs. Future research should prioritize the automation of these tasks. Furthermore, there is a need for enhancements in deep neural networks for question classification and entity identification. Lastly, the deployment of the QA system on distributed computing infrastructure and its evaluation of extensive datasets are avenues for further investigation.

The study [28], highlighted the difficult challenges of QA. The paper addresses and resolves ambiguities within conversations. A task that proves to be difficult even for human participants. The research investigates the limitations of existing end-to-end neural models. It further presents a dataset suited for investigating deeper into the management of ambiguities in conversational QA. The outcomes indicate that relying solely on pre-trained large models such as BERT and BART falls short in recognizing ambiguities or generating the necessary clarification questions for resolution. Future research stemming from Abg-CoQA models may consider incorporating ambiguity as an indicator of ambiguity and utilize a multi-step approach to generate clarification questions. This process involves identifying the type of ambiguity, creating context-specific phrases, and formulating a comprehensive question.

The study [29], investigated the present status of QA systems and their effective utilization of machine learning techniques such as conditional random fields and long short-term memory networks. It suggests the adoption of deep learning approaches to overcome the limitations of these systems. The research integrates the advantages of conditional random fields and long short-term memory networks to introduce a novel model called Bi-LSTM-CRF, which enhances system performance and precision without requiring complex feature rules. Furthermore, it underscores the importance of improving learning methods for feature extraction from the existing training data, given its insufficiency. Subsequent research will emphasize this area of improvement.

The study [30], aims to address the closed-domain reading comprehension QA task on the Quranic Reading Comprehension Dataset by utilizing BERT as the baseline model used to extract an answer span from the given paragraph. They implemented regularization techniques such as weight decay and data augmentation to enhance the model's performance. Through various strategies, they achieved partial reciprocal ranking (PRR) scores of 59\% and 31\% on the development and testing data splits, respectively. The lack of observable improvements in test data performance during the PRR execution may have led to us potentially overlooking the best model state at some point during the training epochs. Another contributing factor could be the relatively limited size of the augmented dataset.

The study [31], presented RGX which is a cooperative self-training framework designed to improve model performance by automatically generating non-trivial question-answer pairs. RGX employs a masked answer extraction task with a communicating learning environment, including a question generator (QG), an answer extractor (AE), and an answer entity recognizer (AER). The framework permits the training of question generation (QG) and answering models using unannotated text corpora, and it utilizes a self-training technique to enhance both models' performance. Experimental results demonstrate RGX's authority over state-of-the-art pre-trained language models and transfer learning (TL) methods in standard QA benchmarks, achieving new state-of-the-art performance with the given model size and transfer learning settings. This paper introduces a novel multi-hop QA model that combines a coarse-grained decomposition approach for breaking down multifaceted queries into simpler single-hop queries and a fine-grained interaction strategy to improve the representation of individual words in the document. These improvements enable the model to extract the complete and accurate sentences required to answer the questions effectively. Through investigations, they prove that the models are consistently and significantly better than the baseline model. This model outperforms the baseline model, but it can not handle complex queries, has different correlation words in different documents, and also needs to predict supporting sentences. This model employs a pipeline-based approach, which is at risk of error propagation.

The study [12], focused on extractive QA and introduced a new end-to-end model for multi-granularity matching, which includes sentence selection, answer extraction, and paragraph recognition. The model utilizes two Siamese BERT (Bidirectional Encoder Representations from Transformers) networks to encode questions and paragraphs and optimizes matching scores for various granularities. Experimental results on four large-scale datasets demonstrate the effectiveness and forcefulness of the proposed MGRC (Multi Granularity Reading Comprehension) model, confirming its potential for improving extractive QA tasks. It does not effectively handle ambiguity, complex queries, and long dependencies.

The study [32], introduced an interactive robot system for healthcare QA aimed at improving healthcare knowledge acquisition. It utilizes multiple attention-based models to enhance the quality of knowledge retrieval, providing a dependable and adaptable framework. Furthermore, a recommendation module enhances user engagement by suggesting pertinent questions for deeper exploration. The system takes into account semantic meaning and medical entities in user queries, surpassing comparable approaches in retrieval experiments by a substantial margin. They use the BERT model, which is not able to cover all dependencies in expectation.

2.2 Multi-based Span Extraction QA

Multi-span extraction is a QA approach that identifies multiple answer spans by predicting the start and end positions of each span within the context.

The study[33], presented an innovative method for addressing questions in machine reading comprehension (MRC) tasks. The answers cannot be straightforwardly retrieved from the provided passages. Traditional

techniques using extractive or generative decoders have their limitations. Extractive methods struggle when dealing with paraphrased or discontinuous answers in the text. Whereas generative methods encounter challenges with syntax and semantics in lengthy sentences. To address these issues, the paper suggests an answer generation approach that combines multiple extracted spans, which are confident n-gram candidates from the passage. This strategy results in answers composed of non-continuous spans. This leads to enhanced accuracy when generating longer answers. Experimental findings using MS MARCO data demonstrate that this method surpasses the performance of conventional one-span and Seq2Seq baseline decoders in producing accurate answers. However, this approach extracts multiple spans without accounting for multi-category queries.

The study[34], focused on addressing issues within extractive QA models. These models have shown promise in accurately predicting answers from passages but sometimes offer answers that lack context. This problem becomes more noticeable when the answer text is found multiple times in the passage. To resolve this, the authors present BLANC (BLock Attention for Context prediction), which consists of two main components: using context prediction as an auxiliary task in multi-task learning and implementing a block attention mechanism for learning the context prediction task. Empirical findings in reading comprehension reveal that BLANC outperforms existing state-of-the-art QA models, and this superiority becomes more pronounced as the frequency of answer text occurrences increases. Furthermore, in an experiment where models are trained on SQuAD and tested on HotpotQA to predict supporting facts, BLANC surpasses all baseline models. However, this approach does not address multi-category queries within a single question.

The study [35], introduced an inventive approach to QA, wherein responses consist of multiple sentence-level segments extracted from lengthy documents. They introduce a fresh QA dataset, known as MASH-QA, which is centered on the consumer health domain and is comprised of questions that necessitate information from several document segments to answer. Furthermore, they put forth a new QA architecture named MultiCo, which utilizes a query-driven, context-aware sentence selection method to locate multi-span answers in extensive documents. Their model, which employs a sentence-selection-oriented approach, achieves substantial performance improvements over existing state-of-the-art QA models, thus serving as a catalyst for further exploration in the domain of multi-span QA. However, this approach extracts multiple spans without accounting for multi-category queries.

The study [36], addressed multiple-span extraction. Previous work advanced on single-answer extraction but questions with multiple answers are unexplored. To handle this gap, the author introduces a new dataset containing questions with multiple answers. They evaluate this dataset with Bert-based models. Roberta outperforms others. The study highlights the flexibility of these models to questions that demand multiple answers. And providing valuable insights for the future development of QA systems and improvements in real-world applications. However, the model requires further improvement in performance and generalization to real-world applications.

The study [37], introduced MUSST (Multi-Span Style Extraction Framework), a novel solution for generative machine reading comprehension (MRC). Their approach effectively addresses issues encountered by single-span extraction models, such as generating incomplete answers or including redundant words. To evaluate MUSST, they test it on the challenging generative Machine Reading Comprehension (MRC) dataset (MS MARCO v2.1) and demonstrate its superior performance compared to the single-span extraction model. This research opens up a new way for generative machine reading comprehension (MRC), complementing the current techniques of single-span extraction and seq-2-seq generation. Even with the support of only an independent ranking classifier, their proposed technique achieves close-to-state-of-the-art overall performance, indicating its significant potential. Sometimes it

needs a clear boundary, there may not be clear boundaries between answer spans, leading to ambiguity in extraction and interpretation.

The study [38], introduced a novel multi-hop QA model that combines a coarse-grained decomposition approach for breaking down multifaceted queries into simpler single-hop queries and a fine-grained interaction strategy to improve the representation of individual words in the document. These improvements enable the model to extract the complete and accurate sentences required to answer the questions effectively. Through investigations, they prove that the models are consistently and significantly better than the baseline model. This model outperforms the baseline, but it still has issues with complex queries, has different correlation words in different documents, and also needs to predict supporting sentences.

These limitations are particularly critical in the context of intelligent information systems, where users frequently pose complex, multi-category queries to retrieve precise information from large-scale knowledge repositories.

3- System Model and Proposed Methodology

This chapter introduces the proposed model for answer extraction. The proposed model is shown in Figure 3. All the components of the proposed model are explained in detail with respect to their work. Supporting algorithms are also available in this chapter.

3.1 System Model

The structure of the proposed model is shown in Figure 3. In this proposed model the algorithm 6, shows the pre-trained encoder consists of both questions and paragraphs. For handling complex queries, there are different methods. In the First step question will be passed to the split function throw delimiters such as ',', 'and', '?', if the query is complex in categorized form, then it will be an individual category. Then it passes the text to the sentence splitter toolkit [39] to divide all text into sentences. Further, it passes all sentences with single-category questions to the similarity technique (ALBERT) [40]. It checks the question with all sentences, the most similar sentence, and the single-category question delivered to the pre-trained model. The pre-trained model takes both question and text as input. It first tokenizes them, then embeds them, and finally gives embedding tokens as an output. This output contains multiple similar spans and then passes all of these similar spans to the argmax () function for the most similar span that has end and start boundaries of the most similar text. The proposed model extracts all boundaries as an output from the text selects the most similar and inserts them into a list. After all possible categories of question list will considered as a final answer.

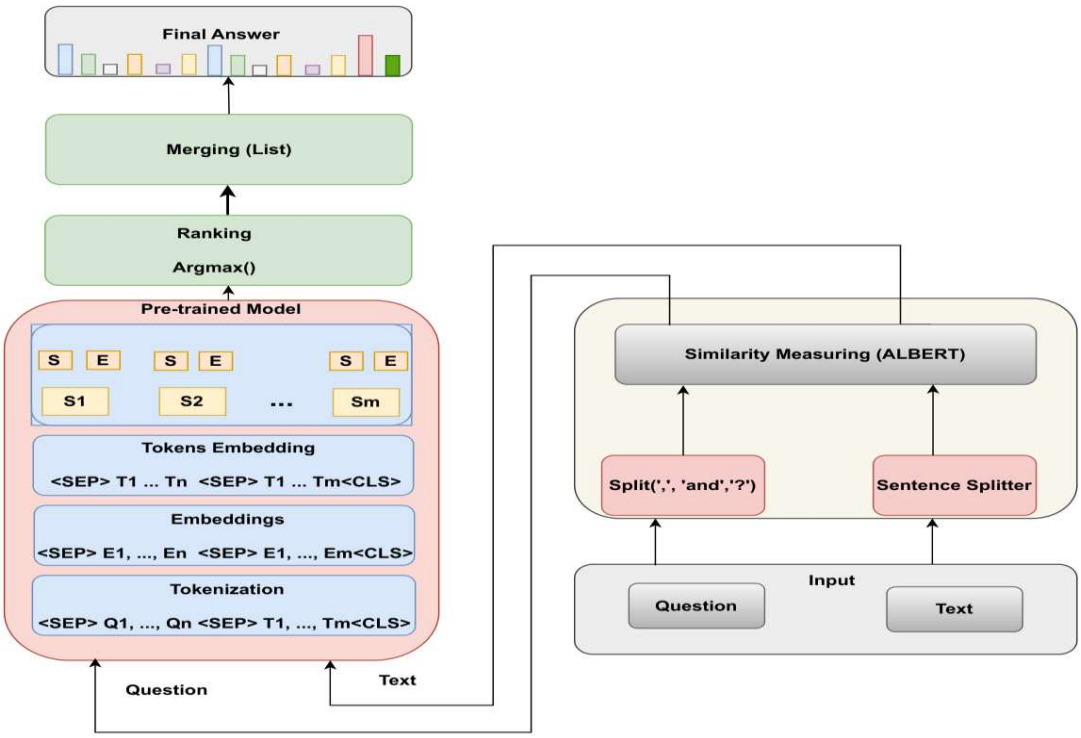


Figure 3: Proposed system model

3.1.1 Pre-trained Models

For QA tasks in NLP, there are already trained models that are quite popular. Some are well-known by their names here:

3.1.1.1 XLNet QA Model

According to [41], XLNet (eXtreme Language understanding interdependence Transformer) is an advanced autoregressive pre-trained model. It learns bidirectional context understanding by optimizing the expected probability across all possible permutations of the factorization order. It successfully addresses BERT's limitations. Additionally, XLNet combines concepts from Transformer-XL, an advanced autoregressive model, during the pre-training phase. Comparing XLNet with BERT that is perform partial probability. XLNet outperforms BERT according to dependencies. To understand its dependencies with an example "New, York, is, a, city". Both BERT and XLNet select "New York" as the prediction target to gain the maximum probability for for $logp\ (Y\ or\ k\ New \mid is\ a\ city)$.

$$For\ BE\ RT = logP\ (New|is\ a\ city) + logP\ (Y\ or\ k\ |is\ a\ city), \quad (3.1)$$

$$For\ X\ LNet = logP\ (New|is\ a\ city) + logP\ (Y\ or\ k\ |New, is\ a\ city), \quad (3.2)$$

Notice that XLNet can understand the connection between the words "New" and "York," something that BERT misses. While BERT might pick up on relationships like "New" with "city" and "York" with "city" in this example, it's clear that XLNet consistently learns more connections when dealing with the same target, and it gathers stronger training signals. The following algorithm is about working of the activities of this algorithm are elaborated in Figure 4.

Algorithm 1: XLNet-Based Question Answering

Extractive QA Pipeline using Pre-trained XLNet

Algorithm 1 XLNet Question Answering

Input: Questions, Text Output: Predicted Final Answer
1 Load the Pre-trained XLNet Model and Tokenizer
2 Input ← {Questions, Text}
3 Tokenization ← {Questions, Text}
4 Encoding embedding ← {Q <SEP> Text <SEP> <CLS>}
5 XLNet ← {tokens embeddings}
6 Generate → multiple spans ["S & E", "S & E" …]
7 Argmax() ← { ["S & E"], [["S & E"] … }
8 Extract the Answer ← Tokens
9 Decode Answer Span
10 Output the Predicted Final Answer

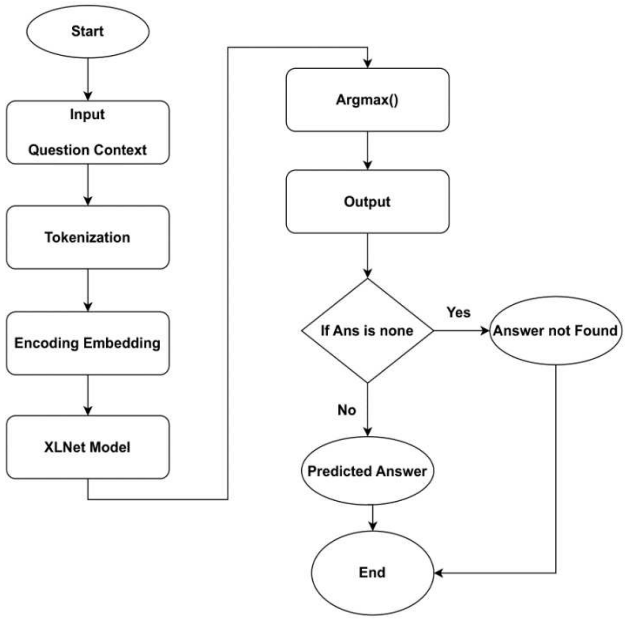


Figure 4: XLNet pre-trained model flowchart

3.1.1.2 Roberta

According to [42], the researchers suggest changes to how BERT is trained to make it work better for specific tasks. They've put together these improvements and tested how well they work when used together. They named this improved setup "Roberta," which stands for Robustly Optimized BERT Approach. In simple terms, the way we use Roberta in a QA system involves three main steps: first, we get the pre-trained model; then, we adjust it to better fit our specific QA task, and finally, we check how well it performs to make sure it gives an accurate answer. The following algorithm is about the workings of Roberta's model, as elaborated in Figure 5.

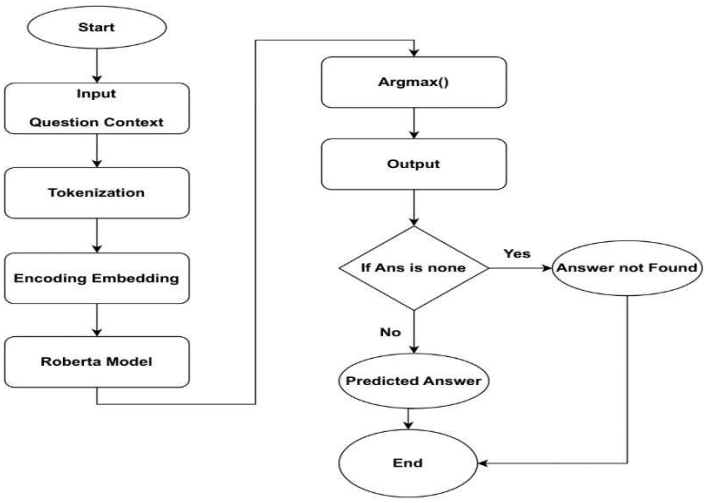


Figure 5: Roberta pre-trained model flowchart

Algorithm 2 RoBERTa Answer Span Extraction

Input: Questions, Text Output: Predicted Final Answer

1 Load the Pre-trained RoBERTa Model and Tokenizer
2 Input ← {Questions, Text}
3 Tokenization ← {Questions, Text}
4 Encoding embedding ← [CLS] {Q [SEP] Text }
5 RoBERTa ← {tokens embeddings}
6 Generate → multiple spans ["S & E", "S & E" …]
7 Argmax() ← { ["S & E"], [["S & E"] … }
8 Extract the Answer ← Tokens
9 Decode Answer Span
10 Output the Predicted Final Answer

3.1.2 Split Method

In Python, we use Split () as a special slicer that can cut text into smaller pieces based on certain rules given in regular expressions. This is helpful when we want to break down the text into specific parts, such as finding words or phrases in a sentence, and can control the rules for how it gets sliced. It was helpful to split queries according to their categories. After this, each single category is passed on for further processing. The splitting of queries is elaborated in the algorithm 3 and Figure 6.

Algorithm 3 Split Function

Input: S1, S2, S3, ... Sn Output: Appended List

1 Input ← { S1, S2, S3, ... Sn }
2 Delimiter ← { "&", ",", "/", "and", etc. }
3 Initialize empty list ← List = []
4 Split() ← { Delimiter, Input }
5 Append to List ← Output

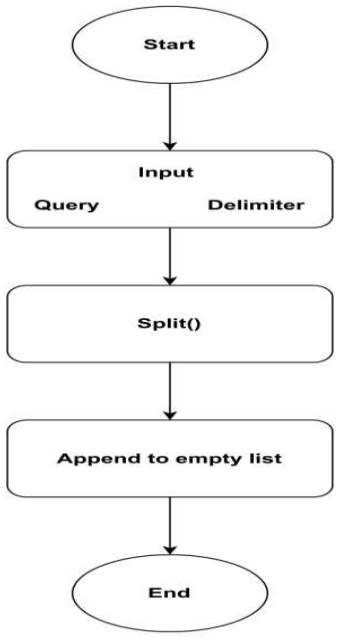


Figure 6: Flowchart of the split function

3.1.3 Similarity Measuring

In a QA system, similarity measuring helps in determining the most relevant sentence from a set of sentences. A question involves evaluating the similarity between the question and each sentence. The system compares the question to each sentence in the set and identifies the one with the highest similarity. This process relies on advanced NLP techniques to understand the context and semantics of both the question and the sentences. It prioritizes the sentence that is most similar to the question and helps the system provide an accurate and contextually relevant answer. Thus, the overall effectiveness of the QA mechanism.

3.1.3.1 Cosine Similarity

According to [43], cosine similarity, which is a math tool, measures how similar two pieces of text are when they are turned into numbers. It's a common way to measure how much text pieces look like each other by examining the angles between their numerical representations in a multi-dimensional space. Suppose there are two vectors A and B as input, each representing a piece of text. Where A = {a1, a2, a3 …} and B = {b1, b2, b3 ...}. Cosine similarity aims to indicate how similar they are.

$$Similarity = \cos(\Theta) = A.B|| A||||B|| = \Sigma n\ i{=}1\ Ai\ Bi\sqrt{\Sigma ni{=}1\ A2\ i}\sqrt{\Sigma n\ i{=}1\ B2} \quad (3.3)$$

A high cosine similarity score indicates a significant level of similarity. The following algorithm is about the cosine similarity technique. It is further elaborated in Figure 7.

Algorithm 4 Cosine Similarity

Input: A = {a1,a2,...an}, B = {b1,b2,...bn} Output: Maximum Similarity Score

1 Input ← A = {a1, a2, a3, ... an}, B = {b1, b2, b3, ... bn}

2 TF-IDF ← [A, B]

3 Calculate Similarity = $\cos(\Theta) = A \cdot B / (\|A\| \|B\|)$

$= \Sigma_i^{-1n} A_i B_i$

$\sqrt{(\Sigma_i^{-1n} A_i^2)} \times \sqrt{(\Sigma_i^{-1n} B_i^2)}$

4 Append to empty List ← Output

5 Select the maximum similarity score

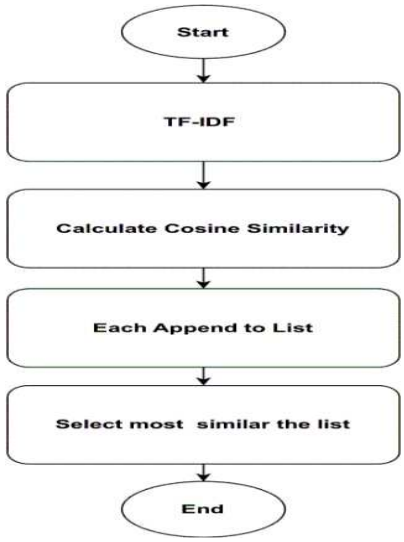


Figure 7: Flowchart of cosine similarity

3.1.3.2 ALBERT Similarity

According to [40], making the model larger during the initial training for understanding language usually makes it better at later tasks. However, making it even bigger becomes challenging because of limits in computer memory and longer training times. To solve these issues, researchers introduced this method to reduce the model's complexity, use less memory, and speed up the training of BERT [44].

We can use ALBERT for the similarity of a question with a set of sentences. This is a type of transformer model for tasks like answering questions or matching sentences. ALBERT is like a more efficient version of BERT, another popular model. It tries to give good results with less computations. The following algorithm is about ALBERT similarity. And it is further elaborated in Figure 8.

Algorithm 5 ALBERT Similarity

Input: Questions, Set of Sentences Output: Most Similar Sentence

1 Load the Pre-trained ALBERT Model and Tokenizer

2 Input ← {Questions, Set of Sentences}

3 Tokenization ← {Question, Set of Sentences}

4 Encoding embedding ← { [CLS] Q [SEP] S }

5 ALBERT { tokens embeddings }

6 Generate → multiple span boundaries ["S1, S2, S3 … Sn", "E1, E2, E3 … En"]

7 Calculate Similarity_Score = Max(Start) + Max(End)

8 Select the most similar sentence and return

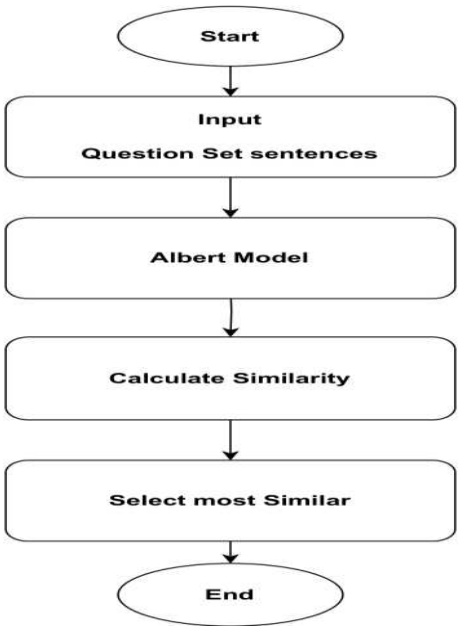


Figure 8: Flowchart of Albert similarity

Algorithm 6 Proposed Model

Input: Q_i (Question), T (Text) Output: Predicted Final Answer

Load the Pre-trained model & ALBERT model
Input ← {Q_i, T}
S_i ← NLTK(T)
If Q → complex:
← q ← Split(Q)
ALBERTSimilarity(q, s_i)
Else
ALBERTSimilarity(Q, s_i)
EndIf
Function ALBERTSimilarity(q, s_i)
BestSentence ← none
MaxScore ← 0
For s in S_i:
input ← Tokenization ← {q, s}
output ← Model ← input
score ← Max(S) + Max(E)
If score > MaxScore:
MaxScore ← Cal_score
BestSentence ← s
EndIf
EndFor
return pre-trainedModel(q, BestSentence)
EndFunction
Function pre-trainedModel(q, s)

```
17      Input ← {q, s}
18      Tokenization ← {q, s}
19      Encoding embedding ← [CLS] { Q [SEP] Text }
20      Output ← pre-trainedModel ← {tokens embeddings}
21      Output → multiple spans [ "S & E", "S & E" … ]
22      Argmax() ← { ["S & E"], [["S & E"] … }
23      Decode  Answer Span
24      Output  the Predicted Final Answer
        EndFunction
```

3.1.4 Evaluation Metrics

In the proposed research, the F1 score and EM have been used as evaluation metrics.

3.1.4.1 F1 Score

The F1 score is a widely used metric in the fields of machine learning, deep learning, and statistics. It serves the purpose of evaluating the effectiveness of classification models, especially when dealing with situations where there is an imbalance in the distribution of classes. This metric acts as a unified measure, striking a balance between precision and recall, it offers a more complete evaluation of a model's performance. The F1 score is employed to evaluate how effectively a system answers, the questions with precision. It is a metric applicable to both extractive and generative QA systems. If we want to find the F1 score, we must find recall and precision.

$$\text{F1 Score} = 2 \times ((\text{precision} \times \text{recall})/(\text{precision} + \text{recall})) \quad (3,4)$$

3.1.4.2 Precision

Precision is a crucial evaluation metric used to measure the correctness of the system's responses. To be more precise, in the context of extractive QA, precision quantifies the extent to which the words or tokens in the extracted answer align with the relevant content in the reference answer or ground truth. For precision and recall we must calculate true positive (TP), False Positive (FP), and False Negative (FN).

$$\text{Precision} = TP/((TP + FP)) \quad (3.5)$$

3.1.4.3 Recall

Recall is a crucial evaluation metric. It evaluates the ability of the system to correctly identify and extract all relevant information from the source text. It measures how well the system captures all the words or tokens from the reference answer in the extracted answer.

$$\text{Recall} = TP/((TP + FN)) \quad (3.6)$$

3.1.4.4 Exact Match

EM (Exact Match) is an evaluation metric. It is evaluated as a binary measurement. Where 1 indicates an exact match, and 0 indicates no match. It is used to evaluate whether the system's extracted answer is matching to the reference answer in terms of the exact words or tokens used.

$$E\,M = exact\ match.\ com\ pute(Reference\ Answer, Predict\ Answer) \quad (3.7)$$

3.1.5 Dataset

This study makes use of a QA dataset, namely the SQUAD 2.0 dataset. For more explanation check out the following dataset.

3.1.5.1 SQUAD 2.0 Dataset

According to [45], SQUAD 2.0 (Stanford QA Dataset), is a well-known and widely used dataset in the area of QA system training and evaluation. It is the extension of the SQUAD dataset. The Squad 2.0 dataset contains a collection of passages from different sources, and questions about them. It contains answerable and unanswerable questions. it contains 150000 question answer pairs. Each pair has a passage, question, reference answer, and start index of answer span. It has two categories of data presentation one for training and other for validation as shown in Figure 9 and Table 1.

```
[ ] dataset

    DatasetDict({
        train: Dataset({
            features: ['id', 'title', 'context', 'question', 'answers'],
            num_rows: 130319
        })
        validation: Dataset({
            features: ['id', 'title', 'context', 'question', 'answers'],
            num_rows: 11873
        })
    })
```

Figure 9: SQUAD 2.0 dataset data representation

Table 1: Squad 2.0 dataset

Dataset	Train	Validation	Domain
SQUAD 2.0	130319	11873	Wikipedia

Chapter 4: Experimental Settings

4.1 Implementation Details

This section presents complete details regarding the implementation, offering an overview of the programming language employed and the libraries utilized for experimental purposes. All experiments were conducted utilizing Python 3.9 within the Google Colab Pro environment, with comprehensive documentation provided in this section. This section investigates specifics such as the quantity and background of datasets, the number of instances, instance labeling, and the partitioning of data into training and validation sets. Furthermore, the section provides insight into the tuning hyperparameters applied during the training phase of the proposed model.

We built our proposed model using just a base-case version of pre-trained models such as XLNet-base and Roberta. We should handle complex queries (multi-category queries) during data pre-processing. Initially, each query is decomposed into single-category sub-queries. Simultaneously, the input text is passed to the NLTK toolkit for sentence segmentation.

Single category query and all text sentences pass to the ALBERT model for question and sentence matching. The ALBERT model finds the most similar sentence of a single-category query.

Query and given text pass both to the pre-trained model to tokenize the single category query and its similar sentence. Subsequently, the pre-trained model outputs the start and end boundaries of spans as embedding tokens.

After that use the argmax function to extract the most similar span from the given set of spans. This process is repeated for all categories; at last it is merged for the final answer.

We set the input sequence length size as 512, the learning rate as 3e-5, epochs as 3, and the batch size as 72.

4.2 Tools Used

All experiments were conducted on a standard workstation with 8GB RAM. GPU-accelerated training was performed using Google Colab Pro, which provides a high-RAM runtime of 51GB. We used Google Colab with Python 3.9 for the actual work. To train our proposed model, we relied on the Python TensorFlow library, and during the process, we used the NumPy library too. We also used the Matplotlib library to create visual representations of our experiments in Python. Details about the tools used are provided in Table 2.

Table 2: Tools Information

Tools	Description
Python 3.9	Python is the predominant programming language in deep learning research, offering extensive libraries and frameworks that facilitate model development and experimentation.
Google Colab	Google Collaboratory (Colab) is a cloud-based Python execution environment supporting GPU acceleration. The Pro edition was utilized in this study for its extended high-RAM runtime of 51GB.
TensorFlow	TensorFlow is an open-source machine learning library developed by the Google Brain team. It is extensively employed for the construction and deployment of machine learning models, with a particular emphasis on deep learning models.
Matplotlib	Matplotlib is a complete library for creating animated, interactive, and static visualizations in Python. It is widely used for plotting and data visualization in scientific research, engineering, data analysis, machine learning, and deep learning.
Transformers	The Transformers library was developed by Hugging Face. It is a popular open-source library for NLP tasks. It provides easy-to-use interfaces to pre-trained models for different NLP applications such as question-answering, language translation, text classification, and summarization.
Datasets	Dataset's library in the Hugging Face Transformers library, specific to natural language processing tasks.
NLTK	NLTK stands for the Natural Language Toolkit. It is a powerful Python library for working with human language data. It is used to split text into sentences.

4.3 Evaluation Metrics

The following metrics (shown in Table 3) have been used to evaluate the proposed model.

Table 3: Evaluation Metrics and Equations

No	**Evaluation Metric**	**Equation**

1	F1 Score	F1 = 2 × (Precision × Recall) / (Precision + Recall)
2	Exact Match (EM)	EM = 1 if predicted answer exactly matches reference answer, else 0

4.4 Hyperparameters Used

Hyperparameters are like the adjustable parts of the model that get fine-tuned while the model is learning. These parts help the model understand and remember patterns, connections, and meanings in the text. Table 4 shows all appropriate Hyperparameters.

Table 4: Hyperparameters and Their Values

Pre-Trained Model	Hyperparameter	Value
XLNet	learning_rate	3e-5
XLNet	train_batch_size	96
XLNet	eval_batch_size	192
XLNet	seed	42
XLNet	epsilon	1e-8
XLNet	lr_scheduler_type	Linear
XLNet	max_seq_len	512
XLNet	num_epochs	3
RoBERTa	batch_size	96
RoBERTa	n_epochs	3
RoBERTa	max_seq_len	386
RoBERTa	learning_rate	3e-5
RoBERTa	lr_schedule	LinearWarmup

Pre-Trained Model	Hyperparameter	Value
RoBERTa	warmup_proportion	0.2
RoBERTa	doc_stride	128
RoBERTa	max_query_length	64

4.5 Dataset

4.5.1 SQuAD 2.0 Dataset

SQuAD 2.0 is a large-scale MRC dataset comprising both answerable and human-generated unanswerable questions, each paired with a Wikipedia context paragraph. The training set comprises a total of 130k questions. For evaluation purposes, we utilize the development set, consisting of 11.9k questions, with 50% of them being answerable.

5. RESULTS EVALUATION AND PERFORMANCE ANALYSIS

5.1 Baseline

To measure the efficiency of our approach. We compare it against its baseline method. such as Bert. The particulars of the baseline model is explain below

5.1.1 BERT

It is a one-step approach. It uses BERT as the encoder to create a question-aware representation of the evidence text. The resulting output is then input into a linear layer to forecast the starting and ending positions of the answer span. For fine-tuning, we utilize Huggingface's PyTorch BERT implementation, adjusting parameters like the number of epochs and the initial learning rate.

5.2 Overall Performance

5.2.1 SQuAD 2.0 Dataset

Table 5 presents the comprehensive performance of various methods on both answerable and unanswerable questions within the SQuAD 2.0 train and validate datasets. To ensure a fair comparison with the baseline approach.

First of all, we train the proposed model on the SQUAD 2.0 dataset. Then evaluate the performance with baseline models. Furthermore, the SQUAD 2.0 dataset is utilized to train and evaluate the chosen baseline methods and compare their results to our proposed model. In the phase of training, we have the pre-processing step. In this step, we load the SQUAD 2.0 dataset, which has two types of data: "training" and "validation". Training data is used to train the model, but validation data is used for evaluation. Before we pass the data to the model, we should find the end position of the span start position that is already given in the dataset. It takes a minimum of 9 hours to train the model. First, we train the baseline model "BERT" and then others. The Figure 10 shows training and validation loss during training of the Bert base model.

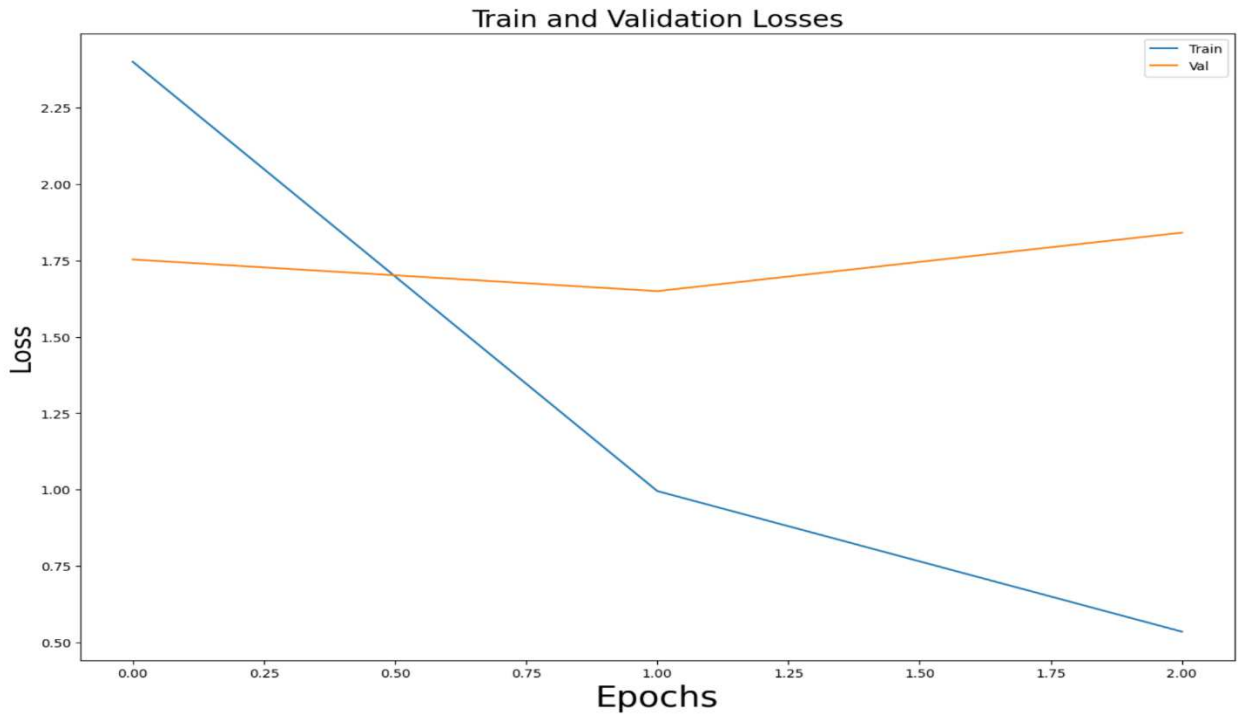


Figure 10: Training and validation loss (BERT)

The Training loss decreases at each epoch's iteration, but validation loss increases at each epoch of iteration. In simpler terms, when we are adjusting a BERT-based QA system to work better for a specific task, we regularly check how much it disagrees with the correct answers during training. We want to make this disagreement as small as possible, indicating that the system is getting better at answering questions correctly. We use a process called backpropagation to update the system's internal settings, making it more accurate over time.

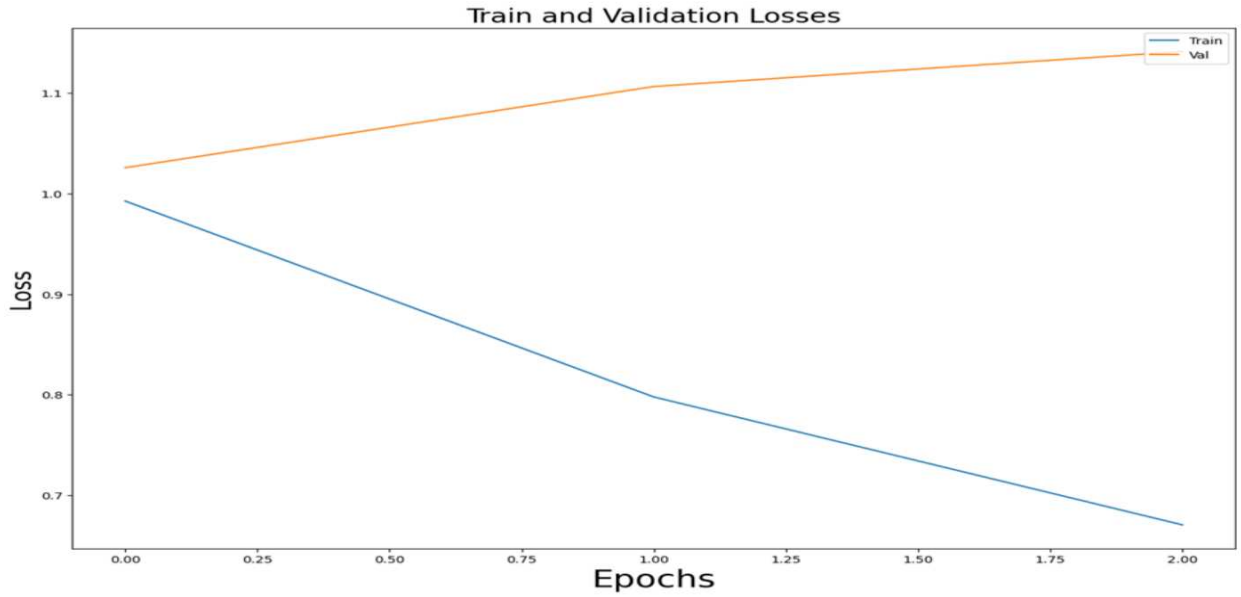


Figure 11- Training and validation loss (Roberta)

Figure 11 shows training and validation for Roberta that is less than the BERT model.Roberta training and validation loss is less than the BERT model in the QA system. Because Roberta validation loss starts at 1.1, the BERT model validation loss starts at 1.75. Training loss decreased to 0.5, but BERT model training loss decreased to 0.6. It means it is better than the BERT model.

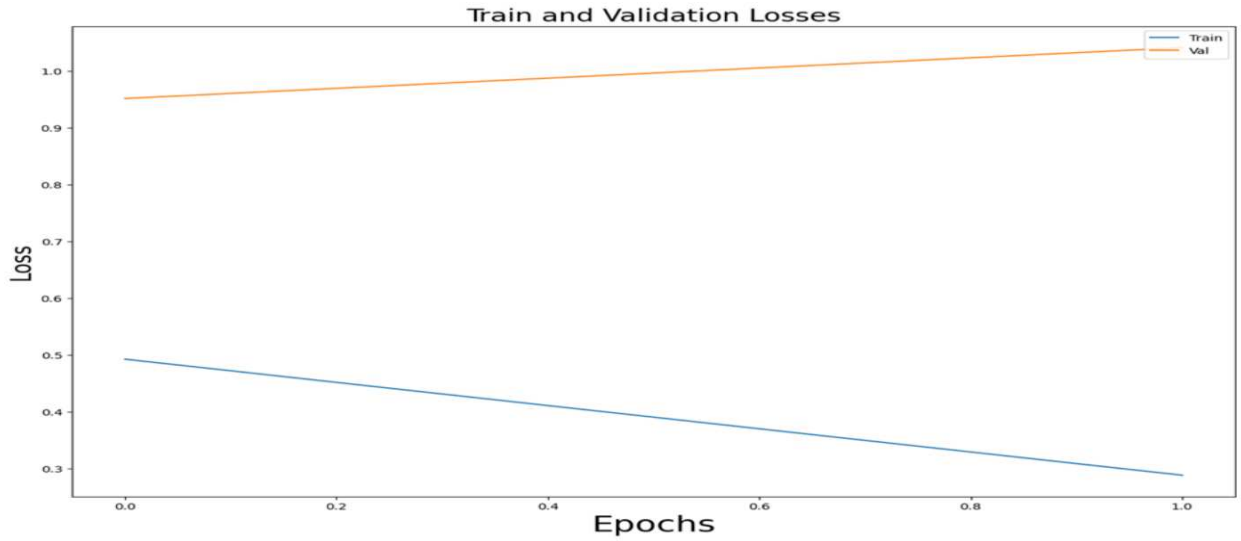


Figure 12 - Training and validation loss (XLNet)

Figure 12 shows training and validation loss for XLNet that is less than the BERT model.

During the proposed model, training validation loss starts at 0.95 and training loss decreases to 0.3. If we look at another baseline model, the proposed pre-trained model outperforms during training.

Our proposed pre-trained model is better than BERT because it uses a smart design. It understands words in both directions, pays attention to where words are placed, and has a special way of remembering important information.

5.2.2 Evaluation Performance Comparison of the Proposed Model and the Roberta, XLNet, and BERT Models on the SQUAD 2.0 Dataset

The results show that our proposed model is better than the baseline model as shown in Table 5. The F1 scores, which check how well the model balances accuracy and completeness, indicate that our proposed model is better.

Our proposed model shows an F1 score of 0.8795, which is better than Roberta (0.8258) and BERT (0.5785). This means our model is better at giving accurate and complete answers to questions. It is better at finding the right balance between being precise and not missing important information compared to the baseline model.

Moreover, when we look at the Exact Match (EM) scores, it is clear that our proposed model is better. It shows an EM score of 0.8496, which is higher than Roberta (0.7658) and much better than BERT (0.5785) in giving exact answers to the questions.

As shown in Table 5, both F1 and EM metrics show that our proposed model works better for answering questions. This suggests our proposed model could be even better than the usual Roberta and BERT models in real-world question-answering situations.

Table 5: Overall Performance of the Proposed Model on the SQUAD 2.0 Dataset (Roberta)

Performance of SQUAD 2.0 Dataset		
Method	**EM**	**F1**
BERT	0.5785	0.6636
Roberta	0.7658	0.8258
Roberta-CQ	0.8496	0.8795

The higher F1scores and EM scores tell us that the proposed model (shown in Figure 13) is strong and accurate, making it a promising choice for improving QA tasks compared to the commonly used Roberta and BERT models.

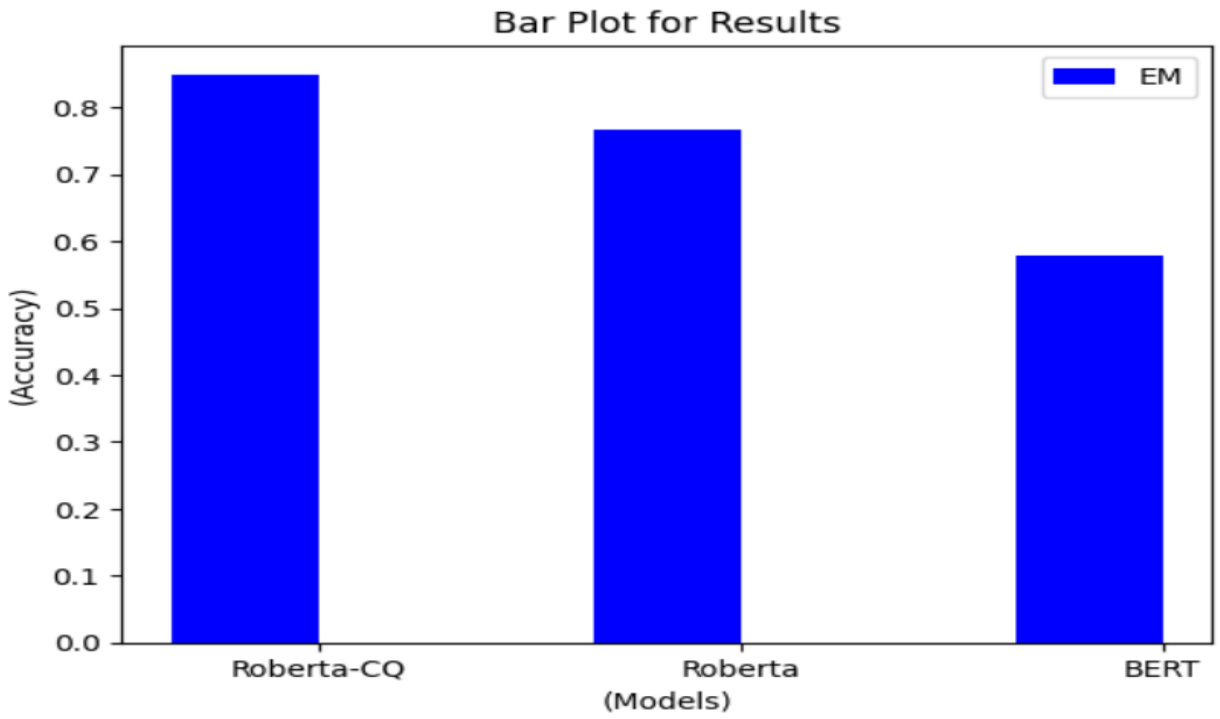


Figure 13 - Performance evaluation of the proposed model (bar chart)

Table 6 shows the proposed model result. Achieved a high F1 score of 0.7749 and an EM score of 0.7479. These scores indicate the model's effectiveness in accurately answering questions. Comparatively, the baseline XLNet model performed fairly well, achieving an F1 score of 0.6545 and an EM score of 0.6814. In contrast, the BERT model had an F1 score of 0.6636 and an EM score of 0.5785.

Table 6: Overall Performance of the Proposed Model on the SQUAD 2.0 Dataset (XLNet)

Performance of SQUAD 2.0 Dataset			
Method	**EM**	**F1**	
BERT	0.5785	0.6636	
XLNet	0.6545	0.6814	
XLNet-CQ	**0.7479**	**0.7749**	

Figure 14 shows the accuracy comparison with a bar plot which clearly shows that the proposed model performs better than both baseline XLNet and BERT. Indicating its excellence in F1 score and EM and highlighting its better performance in natural language processing tasks.

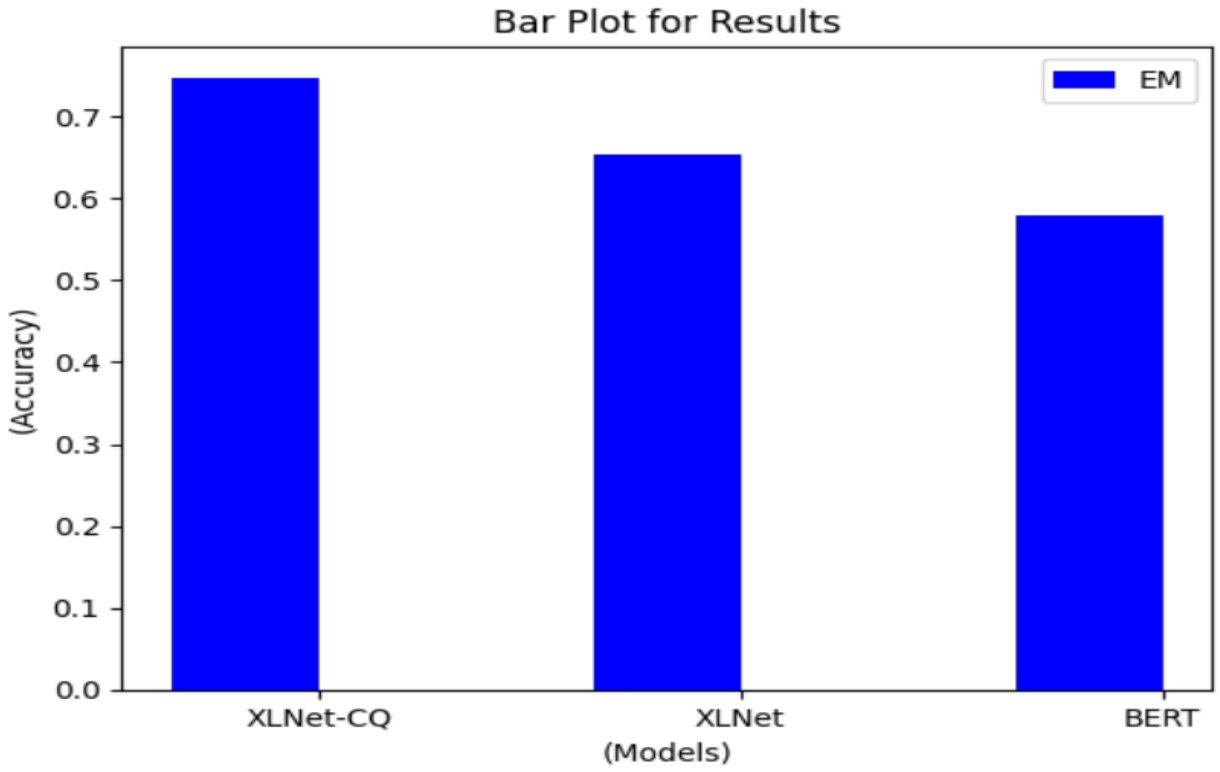


Figure 14 - Performance evaluation of XLNet and BERT (bar chart)

Table 7 compares how well our proposed and baseline models perform in terms of precision, recall, and F1. The results clearly show that our proposed model is better than the baseline.

Table 7: Comparison of Proposed Model and Baseline Models (Precision, Recall, F1)

Method	Precision	Recall	F1
BERT	0.6650	0.6825	0.6636
XLNet	0.6830	0.6894	0.6814
Roberta	0.8206	0.8272	0.8258
XLNet-CQ	0.7781	0.7814	0.7749
Roberta-CQ	0.8826	0.8894	0.8795

The results of our research are presented in Figure 15, displaying the precision, recall, and F1 scores for different models. The Roberta-CQ model achieved a recall of 0.8894, a precision of 0.8826, and an F1 score of 0.8795. In comparison, the Roberta model demonstrated a recall of 0.8272, a precision of 0.8206, and an F1 score of 0.8258. XLNet-CQ exhibited a recall of 0.7814, a precision of 0.7781, and an F1 score of 0.7749. Meanwhile, the XLNet model showed a recall of 0.6894, a precision of 0.6830, and an F1 score of 0.6814. Finally, BERT achieved a recall of 0.6825, a precision of 0.6650, and an F1 score of 0.6636.

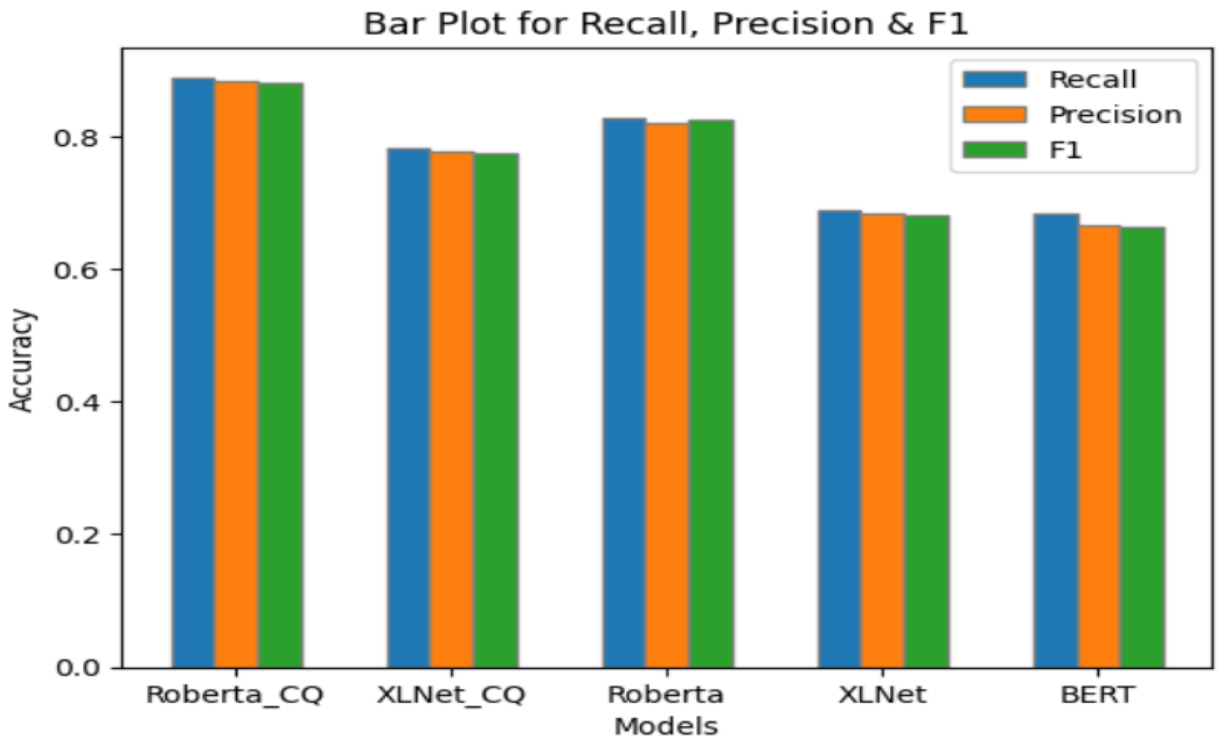


Figure 15 - Performance evaluation based on precision, recall, and F1

5.3 Comparative Evaluation of Proposed Model and Roberta, XLNet, and BERT Models in the Context of Complex Query Handling

In this section, we investigate how well different models handle the difficulty of complex questions that have multiple categories. We conducted experiments using different models, such as BERT, XLNet, Roberta, and our own proposed model.

5.3.1 BERT Model Result for Complex Query

When we used the BERT model shown in Figure 16 on complex queries, we noticed that the model could not give any answers. Ambiguity arises when the BERT model fails to provide an answer.

5.3.2 Comparison of XLNet, Roberta, and BERT Model Results for Long Dependencies

While applying the BERT model shown in Figure 16 for answer extraction in the context of long dependencies, it failed to extract an answer. In contrast, the XLNet and Roberta models provided a single-category answer. This suggests that XLNet and Roberta are more adept at handling long dependencies compared to BERT in the context of answer extraction.

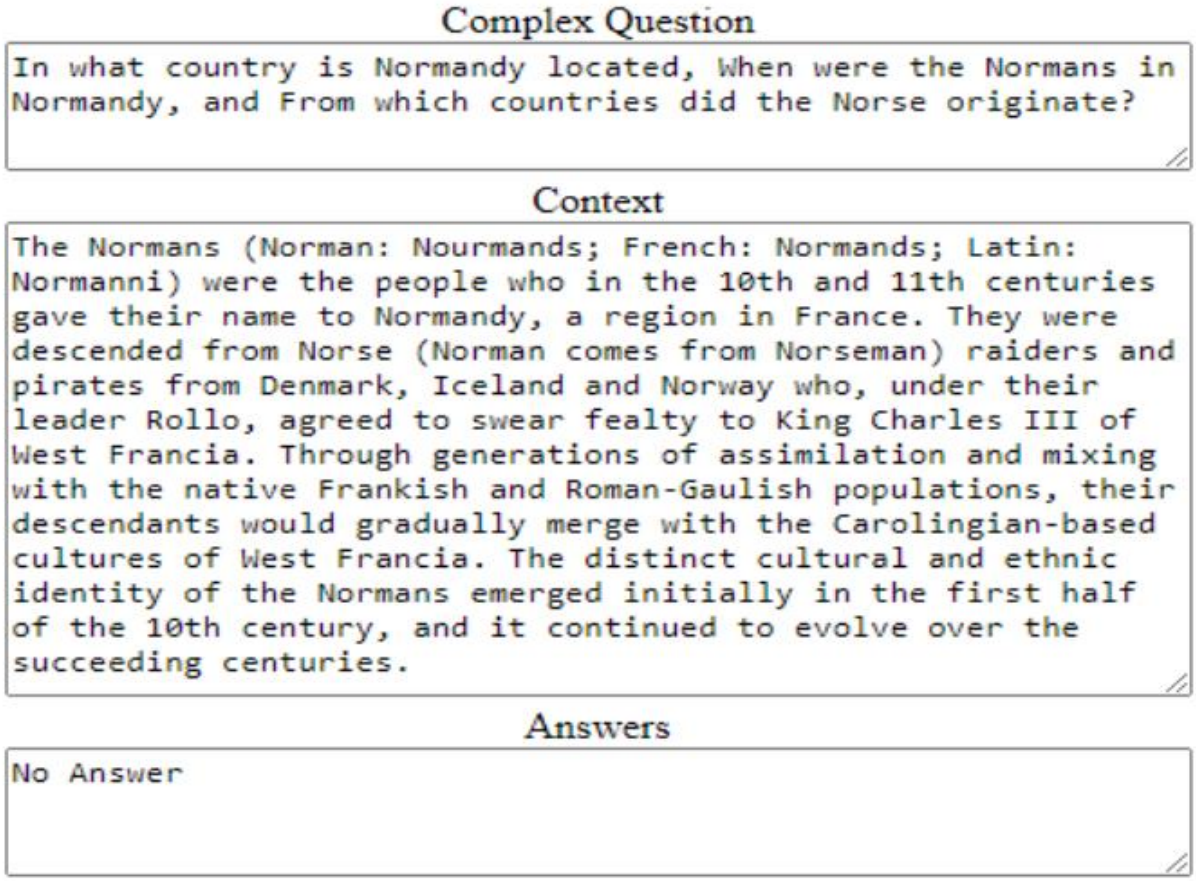


Figure 16 - BERT result for complex query

5.3.3 XLNet and Roberta Models Results for Complex Query

We applied the XLNet and Roberta models shown in Figure 17 to the same complex query. Although these models produced answers for a single category, not all categories were addressed. There is still ambiguity because each model extracts just one response for multi-categories. The result of Roberta model is shown in Figure 17.

Both models are the same in structure, but the difference is just extracting different single-category results. The result of XLNet model is shown in Figure 18.

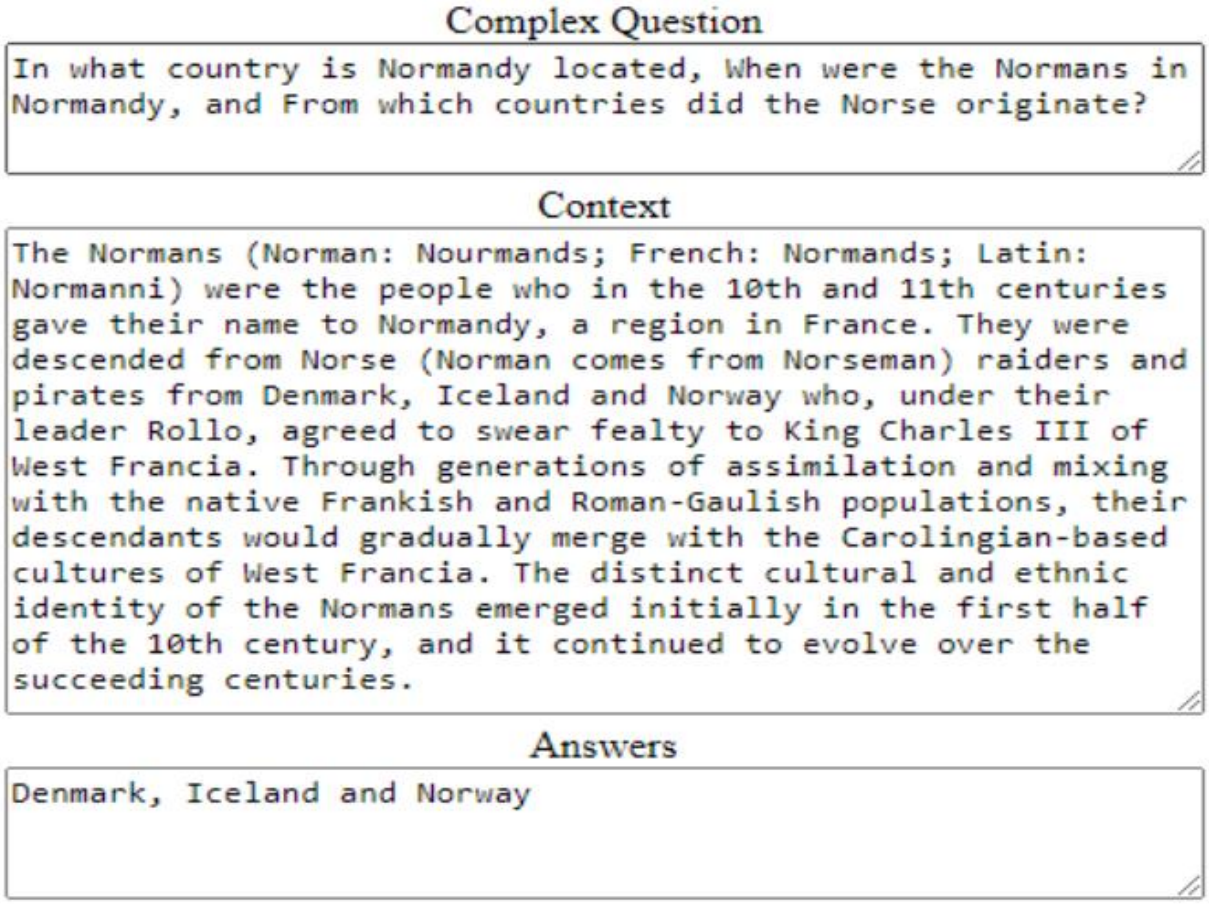


Figure 17 - Roberta result for complex query

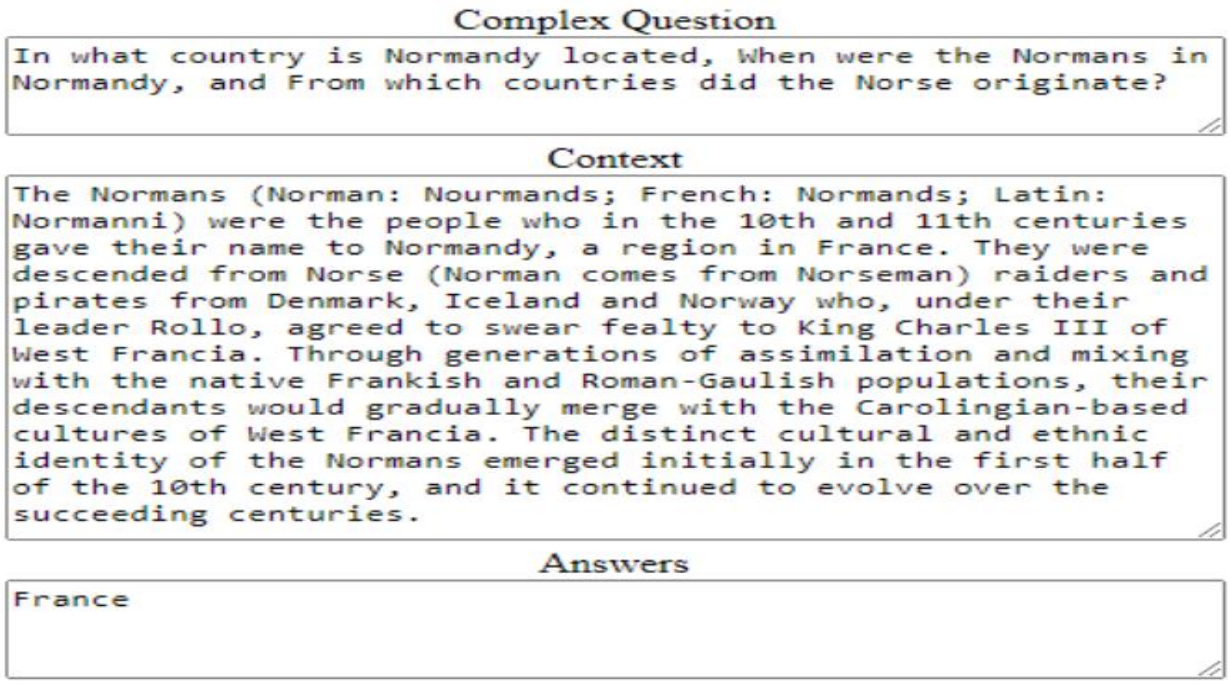


Figure 18 - XLNet result for complex query

5.3.4 Our Proposed Model Results for Complex Query

Our proposed model proved significant advancements in handling complex queries with multiple categories. The model has shown better results by providing precise answers for all categories encompassed within the complex query. The result in Figure 19 shows our proposed model extract answers for all categories.

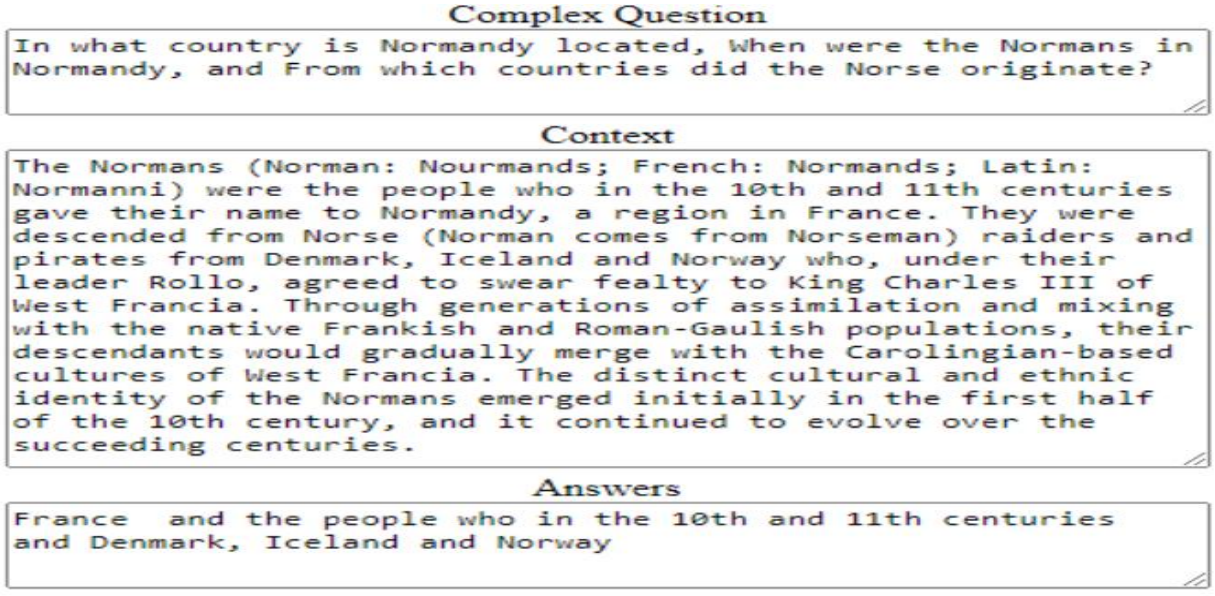


Figure 19 - Our proposed model result for complex query

6. Conclusion and Future Work

This research presented a model that combines pre-training, a split function, and the ALBERT model to better handle complex queries and long dependencies in question answering tasks. The split function helps break down difficult inputs, while ALBERT strengthens semantic understanding across extended text sequences. Together, these components produced measurable improvements in accuracy and contextual reasoning compared to existing approaches.

Experimental evaluation on SQuAD 2.0 demonstrates that the proposed RoBERTa-CQ variant achieves an F1 score of 87.95% and Exact Match of 84.96%, substantially outperforming baseline BERT (F1: 66.36%), standard RoBERTa (F1: 82.58%), and XLNet (F1: 68.14%). The XLNet-CQ variant further validates the effectiveness of the query decomposition mechanism, attaining an F1 of 77.49%. These results confirm that integrating a delimiter-based split function with ALBERT-based semantic matching significantly enhances extractive QA performance on complex, multi-category queries.

The proposed framework holds direct implications for intelligent information systems, including enterprise search engines, knowledge base querying, and automated customer support platforms, where accurate resolution of complex user queries is essential.

Future work will focus on three directions: first, evaluating the framework on additional benchmark datasets such as TriviaQA and Natural Questions to assess generalizability; second, investigating the integration of generative models to handle free-form answer generation beyond span extraction; and third, exploring multilingual extensions of the proposed framework to support cross-lingual QA in intelligent information systems.

Declaration Statement:

Conflict of Interest: The authors declare no conflict of interest.

Ethical Approval: This study utilizes publicly available datasets (SQuAD 2.0) with no direct involvement of human participants.

Funding: No external funding was received for this research.

Data Availability: The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

[1] B. Zope, S. Mishra, K. Shaw, D. R. Vora, K. Kotecha, and R. V. Bidwe, “QuestionAnswer System: A State-of-Art Representation of Quantitative and QualitativeAnalysis,” Big Data Cogn. Comput., vol. 6, no. 4, 2022, doi: 10.3390/bdcc6040109.

[2] R. D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, and V. R. Goodrum,“The Structure and Performance of an Open-Domain Question Answering System,”Comput. Linguist., pp. 563–570, 2000.

[3] U. G. Mada, U. G. Mada, and U. G. Mada, “A Review on Question Analysis,Document Retrieval and Answer Extraction Method in Question Answering System,”pp. 0–4, 2020, doi: 10.1109/ICoSTA48221.2020.1570614175.

[4] S. G. Aithal, A. B. Rao, and S. Singh, “Automatic question-answer pairs generationand question similarity mechanism in question answering system,” Appl. Intell., vol.51, no. 11, pp. 8484–8497, 2021, doi 10.1007/s10489-021-02348-9.

[5] Y. He, Z. Zhu, Y. Zhang, Q. Chen, and J. Caverlee, “Infusing disease knowledge intoBERT for health question answering, medical inference and disease namerecognition,” EMNLP 2020 - 2020 Conf. Empir. Methods Nat. Lang. Process. Proc.Conf., pp. 4604–4614, 2020, doi: 10.18653/v1/2020.emnlp-main.372.

[6] N. Kanodia and K. Ahmed, “Question Answering Model Based ConversationalChatbot using BERT Model and Google Dialogflow,” 2021 31st Int. Telecommun.mNetworks Appl. Conf., pp. 19–22, 2021, doi: 10.1109/ITNAC53136.2021.9652153.

[7]M. Almotairi and F. Fkih, “Developing a Semantic Question Answering System for E-learning Environments using Linguistic Resources,” J. Educ. e-Learning Res., vol. 9,no. 4, pp. 224–232, 2022, doi: 10.20448/jeelr.v9i4.4201.

[8]E. Loginova, S. Varanasi, and G. Neumann, “Towards End-to-End MultilingualQuestion Answering,” Inf. Syst. Front., vol. 23, no. 1, pp. 227–241, 2021, doi: 10.1007/s10796-020-09996-1.

[9] S. Jusoh, “A study on nlp applications and ambiguity problems,” J. Theor. Appl. Inf.Technol., vol. 96, no. 6, pp. 1486–1499, 2018.

[10] M. Zaib, Q. Z. Sheng, and W. Emma Zhang, “A Short Survey of Pre-trained LanguageModels for Conversational AI-A New Age in NLP,” ACM Int. Conf. Proceeding Ser.,2020, doi: 10.1145/3373017.3373028.

[11] Y. Yang and S. Kang, “Common Sense-Based Reasoning using External Knowledgefor Question Answering,” IEEE Access, pp. 0–7, 2020, doi: 10.1109/ACCESS.2020.3045762.

[12] Q. Liu, X. Geng, H. Huang, T. Qin, J. Lu, and D. Jiang, “MGRC: An End-to-End Multigranularity Reading Comprehension Model for Question Answering,” IEEE Trans. Neural Networks Learn. Syst., vol. 34, no. 5, pp. 2594–2605, 2023, doi: 10.1109/TNNLS.2021.3107029.

[13]S. Liu, X. Zhang, S. Zhang, H. Wang, and W. Zhang, “applied sciences Neural Machine Reading Comprehension: Methods and Trends,” 2019, doi: 10.3390/app9183698.

[14] Kumar, F. Natalia, S. Sudirman and D. Al-Jumeily, "Technical Document Query System using Transformer Model-based Machine Reading Comprehension," 2024 17th International Conference on Development in eSystem Engineering (DeSE), Khorfakkan, United Arab Emirates, 2024, pp. 497-502, doi: 10.1109/DeSE63988.2024.10911893.

[15] L. Hirschman, M. Light, E. Breck, and J. D. Burger, “Deep Read: A Reading Comprehension System,” pp. 325–332.

[16] E. Riloff and M. Thelen, “A Rule-based Question Answering System for Reading Comprehension Tests”.

[17] Salıcı and Ü. E. Ölçer, "Impact of Transformer-Based Models in NLP: An In-Depth Study on BERT and GPT," 2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya,

Turkiye, 2024, pp. 1-6, doi: 10.1109/IDAP64064.2024.10710796. [18] T. Nguyen et al., “Ms Marco: a H Uman G Enerated Ma Chine,” no. 2004, pp. 1–10, 2017.

[19] K. Moritz et al., “Teaching Machines to Read and Comprehend,” Adv. Neural Inf. Process. Syst., vol. 28, pp. 1693–1701, 2015, [Online]. Available: http://www.github.com/deepmind/rc-data/\\

[20] Nawaz, A., Ahmed, M., Khan, H.U. et al. Resolving passage ambiguity in machine reading comprehension using lightweight transformer architectures. Sci Rep 15, 42406 (2025). https://doi.org/10.1038/s41598-025-26671-0.

[21]D. Chen, Neural reading comprehension and beyond. Stanford University, 2018.\\

[22] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, “RACE: Large-scale ReAding comprehension dataset from examinations,” EMNLP 2017 - Conf. Empir. Methods Nat. Lang. Process. Proc., pp. 785–794, 2017, doi: 10.18653/v1/d17-1082.

[23] T. Kwiatkowski et al., “Natural Questions: A Benchmark for Question Answering Research,” Trans. Assoc. Comput. Linguist., vol. 7, pp. 453–466, 2019, doi: 10.1162/tacl-a-00276.\\

[24] V. Kommaraju et al., “Unsupervised Pre-training for Biomedical Question Answering,” CEUR Workshop Proc., vol. 2696, pp. 22–25, 2020. \\

[25] L. Q. Cai, M. Wei, S. T. Zhou, and X. Yan, “Intelligent Question Answering in Restricted Domains Using Deep Learning and Question Pair Matching,” IEEE Access, vol. 8, pp. 32922–32934, 2020, doi: 10.1109/ACCESS.2020.2973728.

[26] T. Zhou and P. Gong, “An Object Detection Framework for Span Extraction in Question Answering,” Proc. 2021 7th IEEE Int. Conf. Netw. Intell. Digit. Content, IC-NIDC 2021, pp. 36–40, 2021, doi: 10.1109/IC-NIDC54101.2021.9660483.

[27] P. Do, T. H. V. Phan, and B. B. Gupta, “Developing a Vietnamese Tourism Question Answering System Using Knowledge Graph and Deep Learning,” ACM Trans. Asian Low-Resource Lang. Inf. Process., vol. 20, no. 5, pp. 1–18, 2021, doi: 10.1145/3453651.

[28] M. Guo, “Abg-CoQA: Clarifying Ambiguity in Conversational Question Answering,” pp. 1–17, 2021.

[29] L. Zhang, Y. Luo, P. Gao, D. Zhang, and F. Wan, “Answer Extraction Method Based on BiLSTM and CRF in Q-A System,” 2021 IEEE Int. Conf. Artif. Intell. Comput. Appl. ICAICA 2021, pp. 734–738, 2021, doi: 10.1109/ICAICA52286.2021.9498018.

[30] E. Aftab and M. K. Malik, “eRock at Qur’an QA 2022: Contemporary Deep Neural Networks for Qur’an based Reading Comprehension Question Answers,” 5th Work. Open-Source Arab. Corpora Process. Tools with Shar. Tasks Qur’an QA Fine-Grained Hate Speech Detect. OSACT 2022 - Proc. Lang. Resour. Eval. Conf. Lr. 2022, no. June, pp. 96–103, 2022.

[31] H. Luo, S. W. Li, M. Gao, S. Yu, and J. Glass, “Cooperative Self-training of Machine Reading Comprehension,” NAACL 2022 - 2022 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. Proc. Conf., pp. 244–257, 2022, doi 10.18653/v1/2022.naacl-main.18.

[32] A. Q. Retrieval, “Interactive Healthcare Robot using,” IEEE J. Biomed. Heal. Informatics, vol. PP, pp. 1–12, 2023, doi: 10.1109/JBHI.2023.3320939.

[33] Z. Zhang, Y. Zhang, H. Zhao, X. Zhou, and X. Zhou, “Composing Answer from Multi-spans for Reading Comprehension,” no. January 2021, 2020, [Online]. Available: http://arxiv.org/abs/2009.06141

[34] Y. Seonwoo, J. H. Kim, J. W. Ha, and A. Oh, “Context-aware answer extraction in question answering,” EMNLP 2020 - 2020 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf., pp. 2418–2428, 2020, doi: 10.18653/v1/2020.emnlp-main.189.

[35] M. Zhu, A. Ahuja, D.-C. Juan, W. Wei, and C. K. Reddy, “Question Answering with Long Multiple-Span Answers Ming,” pp. 3840–3849, 2020, [Online]. Available: https://github.com/mingzhu0527/ \\

[36] Q. U. Nswering, “IMPORTANCE OF THE SINGLE-S PAN TASK FORMULATION TO EXTRACTIVE,” pp. 107–116, 2020, doi: 10.5121/csit.2020.101809.

[37] J. Yang, Z. Zhang, and H. Zhao, “Multi-span style extraction for generative reading comprehension,” CEUR Workshop Proc., vol. 2831, 2021.

[38] X. Cao and Y. Liu, “Coarse-grained decomposition and fine-grained interaction for multi-hop question answering,” J. Intell. Inf. Syst., vol. 58, no. 1, pp. 21–41, 2022, doi: 10.1007/s10844-021-00645-w.

[39] S. Bird, “NLTK: The natural language toolkit,” COLING/ACL 2006 - 21st Int. Conf. Comput. Linguist. 44th Annu. Meet. Assoc. Comput. Linguist. Proc. Interact. Present. Sess., pp. 69–72, 2006.

[40] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: a Lite Bert for Self-Supervised Learning of Language Representations,” 8th Int. Conf. Learn. Represent. ICLR 2020, pp. 1–17, 2020.

[41]Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized autoregressive pretraining for language understanding,” Adv. Neural Inf. Process. Syst., vol. 32, no. NeurIPS, pp. 1–18, 2019.

[42]Y. Liu et al., “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” no. 1, 2019, [Online]. Available: http://arxiv.org/abs/1907.11692

[43]Vand F. L. Peipei Xia, Li Zhang, “Learning similarity with cosine similarity ensemble.pdf.” Information Sciences, Jiangsu, China, p. 14, 2015.

[44] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf., vol. 1, no. Mlm, pp. 4171–4186, 2019.

[45] R. Jia, “Know What You Don’t Know: Unanswerable Questions for SQuAD,” 2017.