COMPARATIVE ANALYSIS OF LANGUAGE MODELS ON AUGMENTED LOW-RESOURCE DATASETS FOR APPLICATION IN QUESTION & ANSWERING SYSTEMS

SEYEDEHSAMANEH RANJBARGOL

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ARTS

GRADUATE PROGRAM IN MASTER OF INFORMATION SYSTEMS AND TECHNOLOGY

YORK UNIVERSITY

TORONTO, ONTARIO

MAY 2024

ABSTRACT

This thesis aims to advance natural language processing (NLP) in question-answering (QA) systems for low-resource domains. The research presents a comparative analysis of several pre-trained language models, highlighting their performance enhancements when fine-tuned with augmented data to address several critical questions, such as the effectiveness of synthetic data and the efficiency of data augmentation techniques for improving QA systems in specialized contexts. The study focuses on developing a hybrid QA framework that can be integrated with a cloud-based information system. This approach refines the functionality and applicability of QA systems, boosting their performance in low-resource settings by using targeted fine-tuning and advanced transformer models. The successful application of this method demonstrates the significant potential for specialized, AI-driven QA systems to adapt and thrive in specific environments.

ACKNOWLEDGMENT

I extend my heartfelt gratitude to my supervisor, Professor Marina G. Erechtchoukova, for her invaluable guidance and support throughout my studies. Her expertise has been key in shaping this research. I also thank all my professors from York University for their profound academic insights.

Additionally, I wish to express my deepest appreciation to my husband for his unwavering encouragement and support. His belief in my abilities has been immensely motivating. I am deeply grateful to him for his contributions to my journey.

TABLE OF CONTENTS

LIST OF TABLES

List of Figures

1. Introduction

Question Answering (QA) systems have become essential in the field of information technology for providing fast relevant information. By utilizing advancements in Natural Language Processing (NLP), these systems deliver prompt and relevant responses to user requests. QA systems effectively bridge the vast realm of available knowledge with the specific users' questions due to their ability to process and interpret human language (Wanjawa & Muchemi, 2020).

QA systems can be applied in various sectors, including support in research and education, to enhance customer service and user experience on digital platforms. The focus of this thesis is on the exploration and assessment of QA systems, particularly addressing the challenges associated with low-resource domains. A comparative examination of diverse language models and investigation of data augmentation strategies is undertaken in this study to boost the reliability of QA systems.

Furthering the practical application of these advancements, integration of QA systems into Software-as-a-Service (SaaS) platforms emerges as a crucial step. These systems, empowered by cloud-based NLP, offer an intuitive, conversational interface over data. This feature is particularly beneficial for SaaS applications that demand dynamic and user-friendly interfaces. The inclusion of QA systems can significantly enhance the utility of SaaS software, enabling it to efficiently process and respond to free-format user requests, delivering relevant information. In SaaS environments, QA systems become helpful in developing various client interfaces such as chatbots, social media tools, and speech-enabled utilities, thereby broadening the scope and accessibility of technological innovations for end users (Jboback, 2023).

1.1 Motivation

NLP has enhanced QA systems by making interactions with humans more natural; however, it struggles in data-scarce fields. Modern NLP models depend on extensive datasets for training and fine-tuning. This study addresses the need to enhance QA systems in low-resource specialized domains by analyzing and fine-tuning pre-trained language models, thereby making NLP more inclusive and widely applicable. It aims to improve QA performance in resource-limited settings, increasing technology accessibility and expanding its application scope. The study aims to develop a framework for NLP-based QA system development in a low-resource domain. In addition, this study conducts a thorough comparative analysis of pre-trained large language models using augmented datasets, focusing on their implementation in specialized QA systems. The proposed framework enhances the robustness of the language models and improves the generalization of the results to other low-resource domains.

1.2 The Problem

QA systems exhibit a multi-layered architecture comprising three integral phases: question processing, document retrieval, and answer extraction. Each of the phases is based on sophisticated algorithms. Initially, in the question-processing phase, the system comprehends and interprets the user's question. Subsequently, during the document retrieval phase, it explores the information to identify relevant documents or data sources. The final phase, answer extraction, provides specific answers from the retrieved documents, delivering concise and relevant responses to the user's request.

Central to the efficacy of QA systems is the adoption of pre-trained language models, particularly in contexts demanding specialized, domain-specific knowledge. In such domains, basic search methodologies are inadequate. The models process unstructured texts, generating outputs useful for various NLP tasks. However, to utilize language models in QA systems, a framework is necessary that adapts pre-trained language models to a specific domain and incorporates the enhanced results into a user interface. It is important to recognize that these models achieve optimum performance when they are fine-tuned with text that is specific to the domain in which they are deployed. Such customization enables the models to provide a more precise and contextually relevant understanding of user queries, effectively allowing the models to process and respond accurately to inquiries. The fine-tuning requires large datasets, which are usually not available in low-resource domains.

In this framework, the role of language models, especially those based on the Transformer architecture, is dominant. These models are initially pre-trained on extensive general-purpose text corpora to mimic human language patterns and word co-occurrence. They can then be fine-tuned with domain-specific vocabulary to better suit particular linguistic needs. However, it is not always clear if fine-tuned models outperform their pre-trained counterparts. Fine-tuning helps these models adjust to unique domain characteristics, improving their accuracy and relevance in responding to specific queries.

In the context of this research, fine-tuning QA systems within a specific domain, such as Software-as-a-Service (SaaS) platforms, is explored. Given the limited availability of extensive datasets in such domains, the study also emphasizes the building and augmenting domain-specific datasets.

This research aims to enhance an NLP-based QA system by developing a scalable and adaptable framework. The framework's design facilitates the processing of user requests in free-format English language within a closed domain, ensuring efficient and relevant responses.

1.3 Research Questions and Objectives

This study focuses on the development of a hybrid approach to build a QA system which can be incorporated into an interface for a cloud-based information system, specifically with Oracle Fusion software. Oracle Fusion is a comprehensive cloud-based software suite that integrates multiple applications to streamline various business functions such as enterprise resource planning (ERP), human capital management (HCM), customer relationship management (CRM), and supply chain management (SCM). This platform was developed to replace earlier ERP products and is built on open standards, incorporating a wide range of Oracle and third-party technologies (Thakker, 2015).

This study involves a comprehensive investigation of the following research questions:

1) What is the role of synthetic data in the low-resource domain?
2) Which data augmentation approach is useful?
3) Are fine-tuned models more suitable for closed domains compared to generic pre-trained models?
4) How do various transformer models differ in terms of accuracy and performance in specific closed domains?
5) What is the role of meta-data in question processing?
6) How can information retrieval approaches be used in a low-resource domain?

The thesis experimental setup investigates the effectiveness of fine-tuned models, for specific domains versus generic pre-trained models, using data augmentation and performance analysis to enhance QA systems in low-resource areas. The research focuses on comparing the accuracy and performance of these models, demonstrating the benefits of fine-tuning in domain-specific QA applications. This comprehensive analysis helps illustrate the potential of AI in specialized contexts and lays the groundwork for future advancements in AI-driven, domain-specific QA systems. The overall steps that will be explained in the thesis are demonstrated in Figure 1.1.

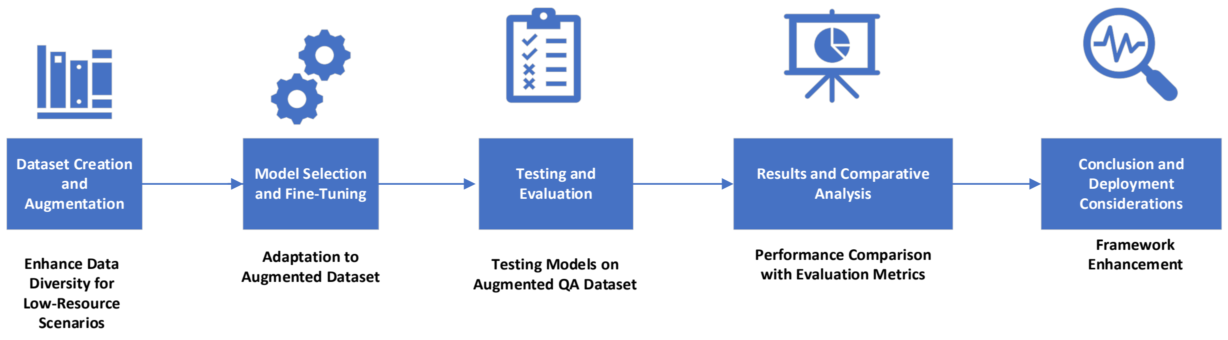

Dataset Creation and Augmentation
Enhance Data Diversity for Low-Resource Scenarios
Model Selection and Fine-Tuning
Adaptation to Augmented Dataset
Testing and Evaluation
Testing Models on Augmented QA Dataset
Results and Comparative Analysis
Performance Comparison with Evaluation Metrics
Conclusion and Deployment Considerations
Framework Enhancement

Figure 1.1 QA System enhancement and analysis workflow

In the next chapter, a comprehensive review of pertinent scientific literature is presented. The methodology adopted for this research is described in the third chapter. Chapter four is dedicated to presenting the experimental results and their subsequent analysis. Chapter five summarizes the conclusions drawn from the research findings.

2. Literature review

2.1 Introduction to Question Answering Systems

2.1.1 Evolution and Significance of QA Systems

Question-answering (QA) systems are a specialized form of computer system designed to respond to inquiries posed by humans in natural language. These systems are engineered to understand the intent behind the question and deliver accurate and relevant responses.

QA is the task of finding concise answers to natural language questions. Most QA systems focus on factoid questions that can be answered with simple facts expressed in short texts. The two main paradigms used for factoid question answering are information retrieval (IR) and knowledge-based question answering. In IR-based approaches, relevant documents and passages are first located, and then reading comprehension algorithms draw an answer directly from spans of text. In contrast, knowledge-based question answering builds a semantic representation of the query, which is then used to query databases of facts (Jurafsky & Martin, 2014).

Factoid question answering through information retrieval aims to locate brief textual snippets from either online sources or other document collections to provide answers to user inquiries. This approach has three main phases: question processing, passage retrieval and ranking, and answer extraction. The question-processing phase extracts the query keywords to match potential documents, and some systems also extract additional information such as answer type, focus, and question type. Passage retrieval can be done by either passing along every passage to the answer extraction stage or filtering passages using named entity or answer type classification. Various methods have been proposed for passage retrieval, including simple word overlap, density-based techniques, IDF-based retrieval, cosine similarity, passage/sentence ranking based on different weighted features, stemming, and morphological query expansion (Soricut & Brill, 2004).

The concluding phase in question answering involves extracting a specific answer from the identified passage. Typically, this is achieved through span labeling, which pinpoints the exact text span containing the answer. A fundamental strategy involves utilizing a named entity tagger to identify the correct answer type span. Recent advancements in answer extraction employ enhanced algorithms, predominantly founded on supervised learning, accompanied by the introduction of basic classifiers and contemporary neural algorithms. Various supervised learning techniques are applied for answer extraction, encompassing answer type matching, pattern matching, keyword distance, novelty factor, apposition features, punctuation location, and the sequences of question terms (Jurafsky & Martin, 2014).

Knowledge-based QA systems execute five distinct tasks: question analysis, phrase mapping, disambiguation, query construction, and querying distributed knowledge (Diefenbach et al., 2018). Entity linking, comprising Named Entity Recognition (NER) and disambiguation of identified entities to the accurate entities in the Knowledge Base (KB), is a pivotal task in QA systems (Chen et al., 2021; Bach et al., 2022).

Several existing tools and methodologies aid in mapping entities, including similarity measures and approaches grounded in neural networks (NN), such as Similarity Measure-based Attribute Ranking Technique (S-MART), hierarchical recurrent neural networks (RNN), bidirectional long short-term memory (BiLSTM), and bidirectional encoder representations from transformers (BERT). For question classification and chunk formation, statistical methods like support vector machine (SVM) and N-gram mining have proven effective (Wang et al., 2019; Zheng et al., 2018).

Rule-based methods can be used for extracting relations from the question when the relations are very frequent. On the other hand, supervised methods can be used when there is a set of questions paired with their correct logical form, such as in the case of simple questions about relations. Most supervised algorithms first parse the questions and then align the parse trees to the logical form, bootstrapping with a small set of rules for building this mapping and an initial lexicon as well (Jurafsky & Martin, 2014).

Supervised approaches exhibit proficiency in managing complex questions involving multiple relations. Zettlemoyer and Collins (2005) illustrated the utilization of advanced syntactic structures and intricate default rules for mapping text sentences to elaborate logical forms. This is achieved by decomposing training examples into smaller tuples, which can be reassembled for parsing unfamiliar sentences.

QA systems predominantly incorporate four main architectures: Semantic Parsing-based, Subgraph Matching-based, Template-based, and Information Extraction-based. Graph Neural Network-based methods have emerged as innovative solutions for multi-hop QA systems, superseding preceding RNN-based models. These models leverage graph representations and graph theory techniques to encode and analyze intricate structures within the input data. In this methodology, input data are mapped to a graph, with nodes symbolizing entities or concepts and edges delineating the relationships between them. The graph is processed by a GNN model that facilitates message passing between nodes and learns representations considering their local neighborhood. Pioneering efforts to employ GNN for multi-hop QA were made by Cao et al. (2018) and Song et al. (2018).

Maheshwari et al. (2019) introduced a technique to generate multiple candidate paths from the principal entity mapped in the knowledge base, avoiding the creation of numerous trees. These paths, along with the corresponding question, are encoded utilizing a BiLSTM model and subsequently ranked based on the dot product of the question and path encoding vectors.

Conversely, Zafar et al. (2018) utilized the Tree-LSTM model to derive latent representations of candidate paths and questions, which were ranked using a similarity function.

Extractive Question Answering focuses on pinpointing a specific text segment within a provided context that acts as the answer to a posed question. This field, a subset of Natural Language Processing (NLP), has seen considerable advancements, finding utility in search engines like Google and chatbots such as IBM Watson. The evolution in this domain is largely attributed to the advent of extensive pre-trained language models like BERT (Jurafsky & Martin, 2014). Wu et al. (2019) explored the transition in answer extraction methods from traditional pattern matching to machine learning, emphasizing the critical role of algorithm selection in enhancing the precision of automatic question-answering systems.

Closed Question Answering (QA) requires domain-specific training to achieve accuracy. This process involves breaking down documents into paragraphs and creating corresponding question-answer pairs for each paragraph. Additionally, limited domain-specific knowledge resources such as ontologies can hinder training efforts. According to the closed extractive approach, the system is designed to work with a specific set of documents or knowledge sources, and the range of potential answers or information is limited to the content contained within those sources. The system typically uses techniques such as keyword-based search, named entity recognition (NER), part-of-speech (POS) tagging, and other rule-based or statistical methods to identify relevant segments of text and extract the information required to answer a given question or fulfill an information request. The application of language models has been proven efficient in processing questions, significantly enhancing the system's ability to understand and respond to complex queries (Allam & Haggag, 2012).

In this context, the study conducted by Kia et al. (2022) brings forth a novel model for closed-domain QA, known as CA-AcdQA. This model addresses the challenges that often arise due to limited datasets. Their unique methodology combines pre-existing contextualized language models, a Convolutional Neural Network (CNN), and a self-attention mechanism to discern the relevance between a question and its contextual sentences. To rewrite ambiguous questions and minimize context, they leverage question expansion techniques and candidate answer identification. The beauty of CA-AcdQA lies in its adaptability across various domains using only a handful of training datasets, presenting an efficient solution to closed-domain QA. CA-AcdQA was benchmarked against nine other QA systems on four closed-domain datasets: Tesla, California, EU-law, and COVID-QA. Notably, the ALBERT variant of the model outperformed all others. It demonstrated a significant increase in Exact Match and F1 score, especially when additional biomedical training resources were employed for the specialized COVID-QA dataset.

In a similar pursuit of improving closed domain Question Answering (QA) systems, Derici et al. (2015) described and evaluated the techniques developed for the question analysis module of a closed-domain QA system. The system was specifically designed to support the education of high school students. It is based on innovative methods for two key problems in question analysis: focus extraction and question classification. These methods integrate a rule-based approach and a Hidden Markov Model (HMM) based sequence classification approach. Both approaches utilize the dependency relations among the words in the question. Derici et al. also compare these solutions with baseline models to assess their performance.

Further extending the discourse on closed-domain QA systems, Esan et al. (2021) proposed that the system architecture for their closed-domain question-answering system consists of three main components: the Input module, the CdQA module, and the Output module. The Input module allows users to input their questions to be answered by the system. The CdQA module, which is the core functional aspect of the system, involves question processing, passage retrieval, and answer extraction. This module uses deep learning algorithms such as the BERT end-to-end transformer model to process the questions, retrieve relevant documents from the database, and extract answers from the retrieved passages. The Ranker module then compares the answers based on an internal score function and outputs the most probable one.

Calijorne Soares and Parreiras (2020) believe that QA systems rely on either a pre-structured database or a collection of natural language documents to generate their answers. QA systems analyze the question and look for the best response using a variety of methods from disciplines like information retrieval (IR), information extraction (IE), and natural language processing (NLP).

Phade and Haribhakta (2021) navigate through the complexities of utilizing neural methodologies in QA, given their demand for extensive annotated data and their tendency to perform optimally only in trained domains. Introducing a novel strategy that integrates data augmentation via question-answer generation with Active Learning, the research seeks to enhance performance in low-resource settings and diverse target domains. Through strategic experimentation and engaging humans at an early stage in the annotation process, the findings reveal that this innovative approach elevates performance in low-resource, domain-specific settings, offering insights into the development of efficient QA systems in new, specialized domains and the influence of human annotation on QA performance.

QA systems are designed to process and respond to user queries expressed in natural language, necessitating the integration of language models. These models, particularly advanced ones like transformers, are necessary for accurately interpreting and handling the complexities of human language. The ability of transformers to enhance QA systems has been notably demonstrated by Devlin et al. (2018), who showed that these models significantly improve the understanding and response to diverse language queries. This reliance on language models for processing natural language forms the basis for an in-depth exploration in the next section of this thesis.

2.2 The Transformer Model: Revolutionizing NLP

The Transformer architecture was popularized in the field of natural language processing by Vaswani et al. (2017), marking its significant application to language models. In their paper, the application of the Transformer architecture represented a significant departure from traditional neural network approaches such as recurrent and convolutional networks. This shift began a new era in NLP, characterized by models that efficiently handle long-range dependencies in text, enhancing language understanding and generation capabilities. The introduction of Google's Transformer model, as outlined in the paper 'Attention Is All You Need', revolutionized NLP. This model differs from previous methods because it processes text sequences in their entirety simultaneously, rather than one piece at a time like RNNs (Vaswani et al., 2017).

First, we will go through the background of the transformer model study, and then we will provide a thorough explanation of the transformer model.

The explanation of the transformer model will be divided into three sections for clarity. Firstly, we investigate the origins of the transformer model. This includes examining the encoder-decoder-based deep learning frameworks and the attention mechanism, which are foundational concepts for understanding transformers. Next, we shift to a detailed overview of the transformer's architecture itself. This is essential for grasping how transformers function and why they are effective for language processing.

Finally, we explore various pre-trained transformer-based language models, which have demonstrated remarkable success across a broad spectrum of NLP tasks in recent years.

2.2.1 Background and Development

A deeper look into transformer-based models, BERT (Bidirectional Encoder Representations from Transformers), introduced by Devlin et al. (2019), has been particularly influential due to its use of a stack of encoders and its masked language modeling (MLM) and next-sentence prediction (NSP) tasks for pre-training. The MLM approach enhances BERT's ability to interpret words from both directions within a sentence, contributing to a deep understanding of language context. Meanwhile, NSP assists in grasping how sentences relate to one another. The introduction of BERT marked a substantial advancement, setting new benchmarks across numerous NLP applications by effectively decoding the context surrounding words, making it especially adept at answering questions.

Several BERT variants have been developed to enhance its functionalities and address its limitations. Examples include ALBERT, a lightweight version of BERT designed to tackle its computational challenges, and RoBERTa, which eliminates the NSP task and employs dynamic masking for training. Introduced by Liu et al. (2020), other noteworthy variants include ELECTRA, SpanBERT, DistilBERT, TinyBERT, ERNIE, and GLaM. Moreover, transformer-based models have emerged beyond encoder-based models that utilize both encoders and decoders. One such example is BART, a denoising auto encoder for pre-training sequence-to-sequence models, considered by Lewis et al., (2020). Also, T5 (Text-to-Text Transfer Transformer) models treat each text processing problem as a ‘text-to-text’ problem. Other models in this category include mT5, MT-NLG, and MiniLM, each with unique features and applications (Raffel et al., 2020).

An innovative approach to answer selection in QA systems using a Transformer-based neural network was proposed in a study by Shao et al. (2019). Traditional neural networks were identified as having limitations in obtaining global text information. To combat this, a Transformer-based neural network that utilizes a bidirectional long short-term memory (BiLSTM) to capture global information and sequential features was proposed. The model outperformed several competitive baselines on the WikiQA dataset, demonstrating the potential of Transformer-based neural networks in QA systems.

Wang et al. (2019) extended the application of BERT to open-domain question-answering tasks by proposing a multi-passage BERT model that globally normalizes answer scores across all passages for a given question. This approach significantly improved BERT's performance in open-domain question-answering tasks, surpassing all state-of-the-art models on multiple benchmarks.

RoBERTa, a robustly optimized variant of BERT, was introduced by Liu et al. (2019). The study showed that BERT was significantly undertrained and could benefit from longer training with larger batches over more data. RoBERTa has displayed improved performance over BERT on several tasks, including question answering.

Finally, Anchal Rawat and Surender Singh Samant (2022), on the comparative analysis of transformer models for QA systems, highlight the shift from traditional methods like 'Bag of Words' to advanced transformer libraries like BERT for efficient QA in large databases. Their study also explores the application of various transformer models like BERT, ALBERT, RoBERTa, XLNET, and others in QA systems using datasets like SQUAD2 (Rawat & Samant, 2022).

2.2.2 Fundamental Concepts: Encoder-Decoder Model and Attention Mechanism

In this section, the foundational aspects of the transformer model will be discussed, focusing on its roots in the encoder-decoder model and the attention mechanism.

2.2.2.1 Encoder-Decoder Model

The encoder-decoder framework, particularly used for sequence-to-sequence tasks such as neural machine translation or abstractive text summarization, usually consists of two Recurrent Neural Networks (RNNs). The encoder in this setup converts a series of symbols into a fixed-length vector, while the decoder transforms this vector back into a new sequence of symbols. A significant challenge with this approach, especially in RNN-based models, is that the encoder compresses all input sentence information into a single, fixed-length vector. This compression can lead to performance issues, particularly with longer sentences, as the model may lose earlier parts of the sequence, resulting in a loss of context.

2.2.2.2 Attention Mechanism

To address the limitations of the encoder-decoder model, particularly issues with processing long-range dependencies, a new technique involving an attention mechanism was developed. This method utilizes a set of context vectors, each representing a sentence from the source text. Each context vector is constructed by combining the decoder's last hidden state with all hidden states from the encoder. This enables the decoder to selectively focus on different parts of the encoder's output during word generation. This design allows the decoder to retrieve the most pertinent information at every step of the sequence, removing the need to reduce the whole source sentence into a single, static vector.

Building on this approach, the transformer model, introduced by Vaswani et al. (2017), marks a notable advancement in sequence-to-sequence modeling tasks like neural machine translation.

This model is particularly distinguished for its enhanced capability to manage long-term dependencies more effectively compared to earlier models.

2.2.3 Architecture of Transformer Models

The architecture consists of two main components: an encoder and a decoder, each made up of six identical layers.

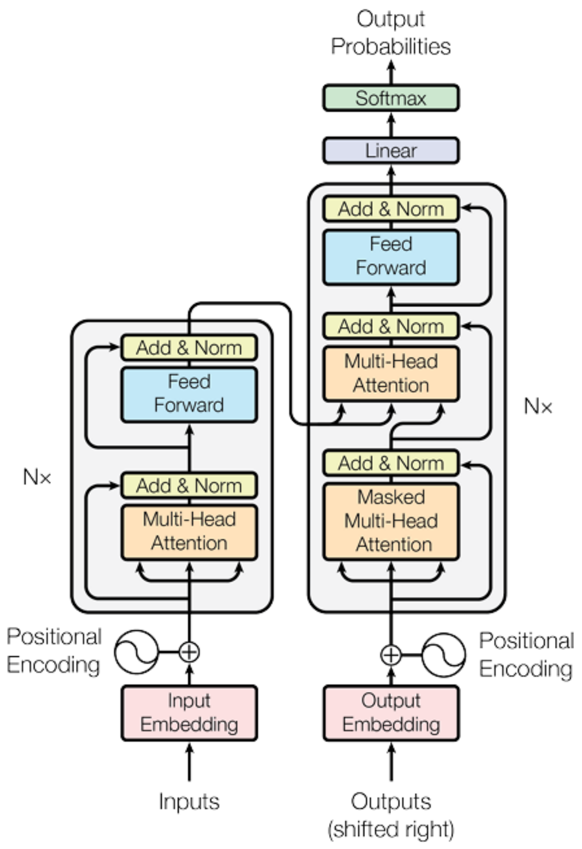


Figure 2.1. Transformer models architecture (Vaswani et al., 2017)

2.2.3.1 Transformer Encoder

Each of the six layers in the encoder includes two sub-layers:

1. **Multi-Head Self-Attention Layer**: This is where each token in the input sequence can 'attend' to all other tokens. This self-attention mechanism, known as 'Scaled Dot-Product

Attention', allows for a more accurate understanding and processing of the input text. It works by creating three matrices for each token: the query (Q), key (K), and value (V) matrices. These are derived from the input embeddings and three weight matrices. The output (Z) for each token in this layer is calculated using these matrices in a formula that incorporating the SoftMax function and the dimensionality of the key matrix.

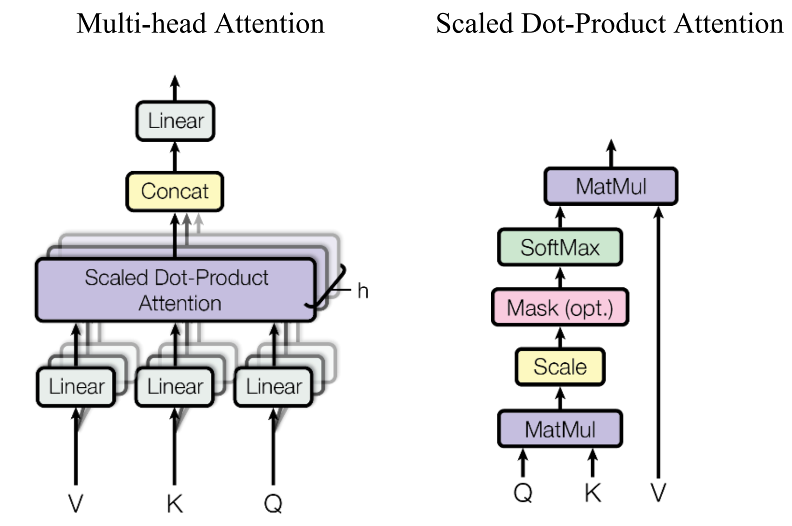


Figure 2.2 (Left) the multi-head attention mechanism.
(right) the scaled dot-product attention mechanism (Vaswani et al, 2017)

2. **Position-wise Fully Connected Feed-Forward Network**: This layer includes two linear transformations with a Rectified Linear Unit (ReLU) activation function in between. In subsequent encoder layers, the key, query, and value vectors are derived from the outputs of the previous layers.

In the multi-head self-attention layer of the transformer model, each word in the input text can focus on every other word within the same text. This particular type of attention is known as Scaled Dot-Product Attention as shown in Figure 2.2. To compute this attention, three matrices are created for each word in the input: the query matrix (Q), the key matrix (K), and the value matrix (V). These matrices are formed by multiplying the word's embedding vector (X) with three different weight matrices (W^Q for query, W^K for key, and W^V for value), which are fine-tuned during the training process.

For each word, the output (Z) in this self-attention layer is determined using a formula that involves the SoftMax function (1) and a scaling factor related to the dimension of the key matrix (K). The encoder enhances this process by employing the multi-head attention mechanism. Essentially, this means that the self-attention calculation is performed eight separate times with eight different sets of Q, K, and V matrices, resulting in eight distinct Z matrices. These matrices are then combined into a single matrix, which is further processed by being multiplying with another weight matrix.

$$softmax(\frac{Q \times K^T}{\sqrt{d_k}})V = Z \tag{1}$$

This combined matrix is then passed through a fully connected feedforward layer, consisting of two linear transformations with a ReLU activation function in between. The formula (2) for this feedforward network (FFN) involves applying the ReLU activation to the linear transformation of the input, followed by another linear transformation. Importantly, in each subsequent layer of the encoder, the Q, K, and V vectors are derived from the outputs of the preceding layer, ensuring a continuous flow of information and context throughout the encoding process.

$$FNN(x) \ = \ max(0, xW_1 + b_1)W_2 \ + \ b_2 \tag{2}$$

2.2.3.2 Transformer decoder

The decoder in the transformer model, like its encoder counterpart, comprises six identical layers, but uniquely includes an extra sub-layer that plays a vital role in the model's overall function. The first key feature of the decoder is its multi-head self-attention layer, similar to the encoder. This layer allows each position in the decoder to focus on earlier positions in the sequence, maintaining the auto-regressive nature essential for sequential tasks such as language generation. Another distinct aspect of the decoder is the additional sub-layer that performs multi-head attention over the encoder's output. In this sub-layer, the query inputs are derived from the decoder's preceding layer, while the keys and values come from the encoder. This configuration enables the decoder to effectively translate and integrate the encoder's processed input.

Furthermore, each decoder layer includes a fully connected feed-forward layer, consisting of two linear transformations with a ReLU activation function in between, mirroring the structure seen in the encoder. In the decoder, the multi-head self-attention layer and the fully connected feed-forward layer each incorporate residual connections, which are then followed by layer normalization. This configuration aids in stabilizing the training process and enhances the gradient flow during backpropagation, contributing to more effective learning in neural network models.

The transformer also emphasizes maintaining the sequence's positional information by adding sinusoidal positional encodings to the input embeddings at the base of both the encoder and decoder stacks. The model maintains specific dimensionality, with input and output dimensions set at 512 and inner-layer dimensions set at 2048.

2.2.4 Pre-trained Transformer Models and Their Evolution

In the next section, a concise overview of various pre-trained language models to be compared in this thesis will be provided.

2.2.4.1 BERT (Bidirectional Encoder Representations from Transformers)

The BERT model has greatly enhanced language processing capabilities (Devlin et al., 2019). A distinctive feature of BERT is its use of the transformer architecture's encoder component. Unlike typical transformer-based models, which use both an encoder and a decoder, BERT focuses solely on the encoder for tasks such as natural language inference, question answering, and classification. Instead, BERT is adept at encoding contextual representations bidirectionally, capturing the nuances of language from both preceding and following text in a sentence. The training regime for BERT centers on two principal activities: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM, approximately 15% of the words in any given text are masked, and the model must infer these masked words using the contextual hints provided by the surrounding words. The masking occurs such that 80% of the time a word is replaced with a placeholder, 10% of the time with a random word, and the remaining 10% it is left as is, challenging the model to accurately predict and understand the hidden words based on context. This approach helps the model focus on the actual word in its context. The second task, Next Sentence Prediction (NSP), involves determining whether one sentence naturally follows another.

The input to BERT includes three special representations: token, segment, and positional encodings, which combine to form a vector fed into the encoder. BERT's pre-training used extensive corpora, including the BookCorpus and Wikipedia. Importantly, the same architecture applies to both the pre-training and the fine-tuning stages, with all parameters adjustable during fine-tuning. Special symbols like CLS (added at the beginning of each input example) and SEP (a separator token) play crucial roles in the model's understanding of text structure.

BERT is available in two sizes: BERTBASE (with 12 layers, a hidden size of 768, and 12 self-attention heads, totalling 110 million parameters) and BERTLARGE (with 24 layers, a hidden size of 1024, and 16 self-attention heads, totalling 340 million parameters).

In testing, BERT has shown notable performance improvements in various tasks like classification and textual analysis, for both short and long texts. However, BERT has its limitations. Since it is not auto-regressive and processes tokens separately, it can sometimes generate less coherent bigrams.

2.2.4.2 Distilbert: Efficient Variant of BERT

DistilBERT emerges as a streamlined alternative to larger pre-trained models in NLP, addressing the need for efficient operation in environments with limited computational resources. This model, which holds 97% of BERT's language understanding capabilities while being 40% smaller and 60% faster, represents a significant step in making advanced NLP more accessible and practical for various applications.

Developed through a method called knowledge distillation during the pre-training phase, DistilBERT leverages the foundational knowledge of BERT. It incorporates a triple loss approach that combines language modeling, distillation, and cosine-distance losses. This approach not only shrinks the model size but also ensures that it maintains a high degree of language comprehension.

The practical implications of DistilBERT are substantial, as it is both cheaper and faster. The model's reduced size and increased efficiency do not significantly compromise its performance, making it a robust option for a wide range of NLP tasks. In essence, DistilBERT stands out as a cost-effective, efficient solution for bringing complex language understanding to environments where computing resources are constrained (Sanh et al., 2019).

2.2.4.3 Albert-base: Lite Version for Computational Efficiency

The ALBERT model is a transformer-based machine learning model, specifically designed for understanding and processing English text. It is pre-trained in a self-supervised manner on a large English corpus, using two main objectives: Masked Language Modeling (MLM) and Sentence Ordering Prediction (SOP).

In MLM, 15% of the words in a sentence are randomly masked, and the model predicts these masked words, learning a bidirectional representation of the sentence. Unlike traditional recurrent neural networks or autoregressive models like GPT, ALBERT handles sentences as a whole. The SOP objective involves predicting the order of two consecutive text segments.

ALBERT is unique in that it shares layers across its Transformer architecture, meaning all layers have identical weights. This design reduces memory usage but maintains a computational cost similar to BERT-like models. The version described here, "ALBERT-base-v2", has improved performance due to different dropout rates, more training data, and extended training time compared to its predecessor. Its configuration includes 12 repeating layers, 128 embedding dimensions, 768 hidden dimensions, 12 attention heads, and 11 million parameters.

Intended for fine-tuning on specific tasks, ALBERT is suitable for applications involving whole sentences, such as sequence classification, token classification, or question answering.

The model's training data comprises BookCorpus and English Wikipedia, excluding lists, tables, and headers. Its preprocessing involves lowercasing and tokenizing texts with a 30,000-word vocabulary. During training, the masking procedure involves replacing 15% of tokens with a [MASK] token, a random token, or leaving them as is in varying proportions (Lan et al., 2019).

2.2.4.4 MobileBERT: Optimized for Mobile Devices

MobileBERT is a compact, task-agnostic adaptation of the BERT_LARGE model, designed specifically for resource-limited devices such as mobile phones. It maintains a lean structure by incorporating bottleneck designs and optimizing the balance between self-attentions and feed-forward networks. Despite the growing success of large pre-trained NLP models, their substantial size and latency render them unsuitable for mobile devices. MobileBERT addresses this by significantly reducing size and increasing processing speed while retaining versatility for various NLP tasks.

The development of MobileBERT involved training a specially designed teacher model, a variant of BERT_LARGE with an inverted-bottleneck structure. Knowledge was then transferred from this teacher model to MobileBERT. This resulted in MobileBERT being 4.3 times smaller and 5.5 times faster than the standard BERT_BASE, yet still delivering competitive performance on key benchmarks. For instance, on the GLUE natural language inference tasks, it scores only slightly lower than BERT_BASE, and on the SQuAD v1.1/v2.0 question answering tasks, it even surpasses BERT_BASE.

The success of MobileBERT demonstrates the effectiveness of keeping the model deep but thin and using bottleneck/inverted-bottleneck structures for efficient knowledge transfer. These strategies, along with progressive knowledge transfer, are potentially applicable to other model compression challenges. The practical implication of MobileBERT is significant: it allows the deployment of advanced NLP applications on mobile devices, making them more accessible and versatile. Figure 2.3 illustrates the structures of BERT, Inverted Bottleneck BERT, and MobileBERT. In both the IB-BERT and MobileBERT diagrams, the red lines represent the flow of information between different blocks, while the blue lines indicate the flow within each block.

The training process of MobileBERT involves emulating the layer-by-layer structure of IB-BERT (Sun et al., 2020).

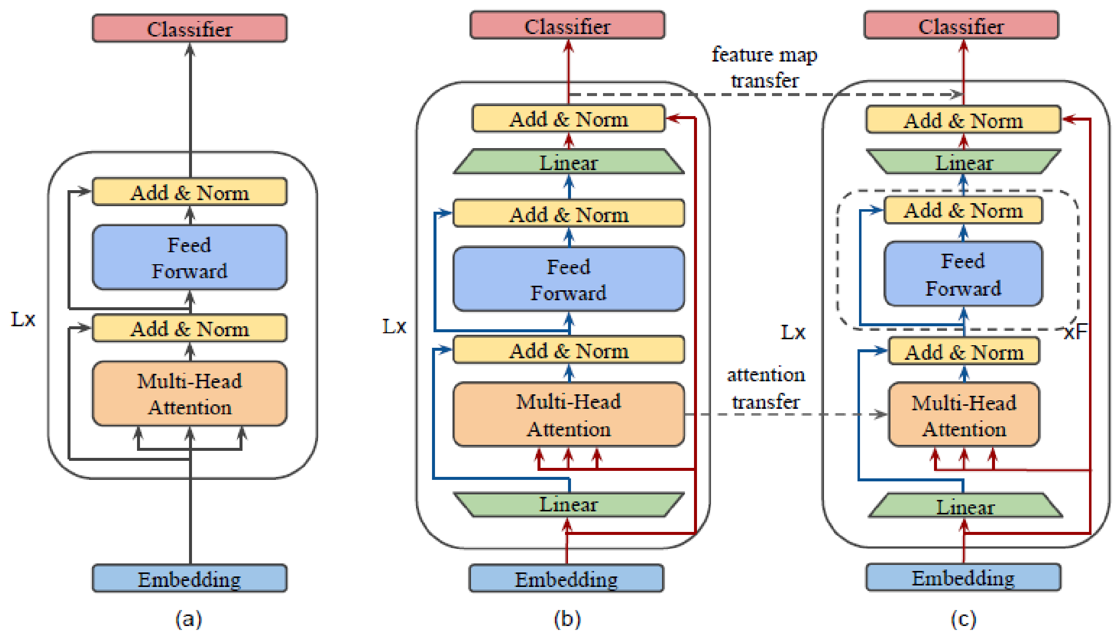


Figure 2.3 Visual comparison of three models: (a) BERT; (b) Inverted-Bottleneck BERT (IB-BERT); and (c) MobileBERT (Sun et al, 2020).

2.2.4.5 Roberta: Robustly Optimized BERT

RoBERTa is a transformer-based model that has been pre-trained on a vast English text corpus using self-supervision. This approach involves training on raw texts without human labeling, allowing for the use of extensive publicly available data. It employs an automatic method for generating input data and corresponding labels from these texts.

The significant pre-training technique used in RoBERTa is MLM. In this method, 15% of the words in a sentence are randomly hidden, and the model is tasked with predicting these masked words. This process differs from that of traditional RNNs and autoregressive models like Generative Pre-trained Transformer (GPT), which process words sequentially or mask future tokens, respectively. RoBERTa's approach enables it to learn a bidirectional understanding of sentences.

Such training equips RoBERTa with a deep internal representation of the English language, making it highly effective for feature extraction in various downstream tasks. For instance, when provided with a dataset of labeled sentences, one can employ the features extracted by RoBERTa to train a standard classifier (Liu et al., 2019).

2.2.4.6 Bart (Bidirectional and Auto-Regressive Transformers)

BART is a transformer-based sequence-to-sequence (seq2seq) model, combining a bidirectional encoder similar to BERT and an autoregressive decoder similar to GPT. It functions as a denoising autoencoder that is pre-trained in two steps: first, by intentionally corrupting text with a noising function, and second, by learning to reconstruct the original, uncorrupted text.

Effective in both text generation tasks (like summarization and translation) and comprehension tasks (such as text classification and question answering), BART is both simple and versatile. It generalizes the features of BERT through its bidirectional encoder and GPT via its left-to-right decoder, encompassing various other pretraining schemes.

BART's training involves novel approaches to text corruption, including random sentence shuffling and an in-filling scheme where spans of text are replaced with a mask token. These methods have proven effective, with BART matching RoBERTa's performance in GLUE and SQuAD benchmarks with similar training resources. It sets new state-of-the-art records in diverse tasks like abstractive dialogue, question answering, and summarization, showing up to 6 ROUGE points improvement. In machine translation, it surpasses back-translation systems by 1.1 BLEU points with only target language pretraining (Lewis et al., 2020).

2.2.4.7 T5(Text-to-Text Transfer Transformer)

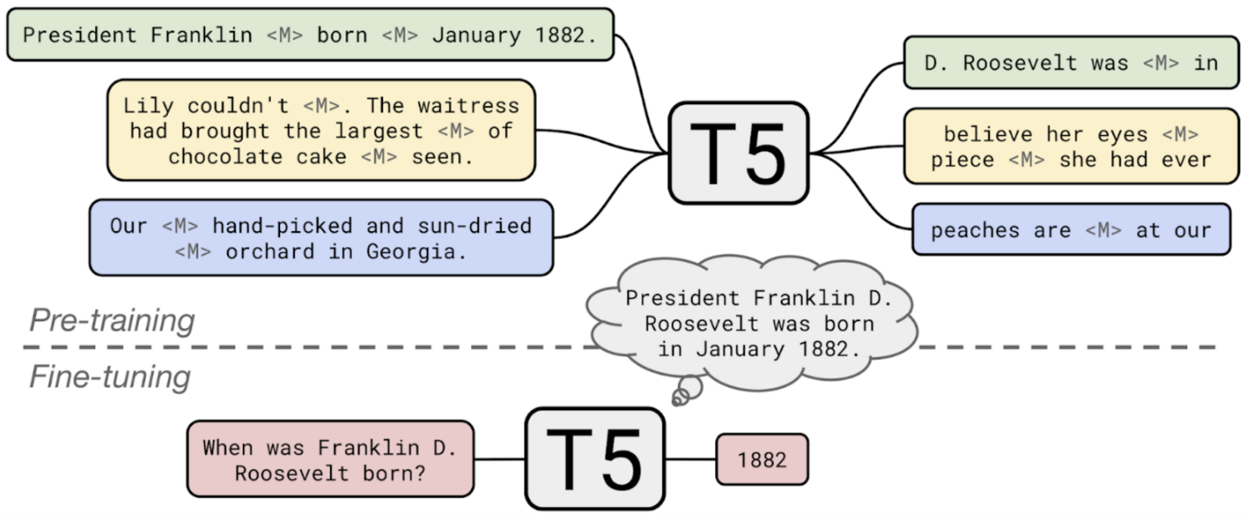
T5, or Text to Text Transfer Transformer, is a cutting-edge model based on the Transformer architecture, introduced by Raffel et al. in 2019. It excels in sequence-to-sequence (seq2seq) tasks, which involve both input and output as text strings. Unlike BERT, which predicts single tokens, T5 can generate multiple words at once, focusing on outputting text sequences rather than individual tokens.

The training of T5 utilized the 'Colossal Cleaned Common Crawl' (C4) dataset, a massive collection of web-scraped, pre-processed text totaling 750GB. The dataset was refined by removing markup and non-text elements, retaining only sentences with proper punctuation. T5's architecture is aligned with BERT-LARGE, featuring configurations such as df = 4096, dmodel = 1024, dkv = 64, and a 16-head attention mechanism. It comes in two variants: t5BASE with 16 layers and t5LARGE with 32 layers each in the encoder and decoder (df: the inner dimension of the feed-forward neural network layers in the model; dmodel: the dimension of the input and output of each layer in the model; dkv: the dimension of key and value vectors in the attention mechanism).

A key feature of T5 is its ability to handle a wide range of NLP tasks simultaneously. It uses task-specific prefixes, like "translate English to German" or "summarize," to understand and adapt to different tasks. The model's self-attention mechanism, illustrated in its documentation, shows how it processes input and output elements. Unlike BERT, which uses a fully visible attention pattern allowing bi-directionality, T5 employs a causal attention approach with a prefix. In this method, T5 generates a prefix for the task, then follows a causal pattern, where each token only sees preceding tokens.

T5's hyperparameter selection was similar to BERT's, with a notable difference in masking strategy. Instead of masking individual tokens, T5 masks sequences of tokens, finding that a 15% masking rate yields optimal results. The model's span lengths are kept short to facilitate accurate predictions with limited surrounding word context.

Overall, T5 stands out as a unified model, demonstrating state-of-the-art performance across a variety of NLP tasks. Figure 2.4 shows the T5 framework, where tasks like translation, question answering, and classification are uniformly handled by inputting text and training the model to generate target text (Raffel et al., 2019).


President Franklin <M> born <M> January 1882.
Lily couldn't <M>. The waitress had brought the largest <M> of chocolate cake <M> seen.
Our <M> hand-picked and sun-dried <M> orchard in Georgia.
T5
D. Roosevelt was <M> in
believe her eyes <M> piece <M> she had ever
peaches are <M> at our
Pre-training
Fine-tuning
President Franklin D. Roosevelt was born in January 1882.
When was Franklin D. Roosevelt born?
T5
1882
Figure 2.4 Generating the desired output using the T5 model (Raffel et al, 2019)

2.2.4.8 Minilm: Efficient Deep Self-Attention Distillation

MiniLM introduces a novel and efficient method for compressing large pre-trained Transformer-based language models, like BERT. This method focuses on knowledge distillation, specifically deep self-attention distillation, where a smaller model (the student) learns by imitating the self-attention mechanisms of a larger model (the teacher). Self-attention, a critical component of Transformer networks, is where MiniLM's distillation process concentrates.

The unique aspect of MiniLM's approach is the distillation of the self-attention module from the teacher's final transformer layer. This includes not just the attention distributions (the scaled dot-products of queries and keys, as used in prior methods) but also introduces the scaled dot-product of values within the self-attention module as new knowledge to guide the student model. Additionally, the use of a 'teacher assistant' has been found beneficial in this distillation process.

In Figure 2.5, the student model is trained by closely replicating the self-attention behavior of the teacher's final transformer layer. Additionally, Wang et al. (2019) incorporate self-attention value-relation transfer for more profound mimicry, resulting in the student models known as MiniLM.

The effectiveness of MiniLM is demonstrated through its performance in various tasks. Despite using only about 50% of the teacher model's parameters and computational resources, MiniLM retains over 99% accuracy on SQuAD 2.0 and several GLUE benchmark tasks. This efficiency makes MiniLM particularly valuable for real-life applications where large models may be impractical due to latency and capacity limitations. Overall, MiniLM offers a practical solution for achieving high model performance with significantly reduced model size (Wang et al., 2019).

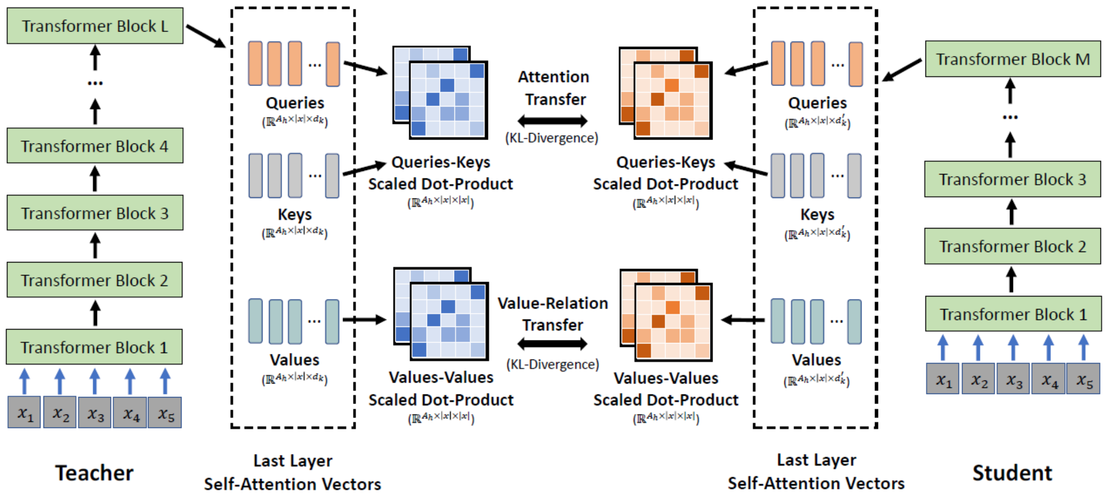

Transformer Block L
...
Transformer Block 4
Transformer Block 3
Transformer Block 2
Transformer Block 1
x_1 x_2 x_3 x_4 x_5
Teacher
Queries ($\mathbb{R}^{A_h \times |x| \times d_k}$)
Keys ($\mathbb{R}^{A_h \times |x| \times d_k}$)
Values ($\mathbb{R}^{A_h \times |x| \times d_k}$)
Last Layer Self-Attention Vectors
Queries-Keys Scaled Dot-Product ($\mathbb{R}^{A_h \times |x| \times |x|}$)
Values-Values Scaled Dot-Product ($\mathbb{R}^{A_h \times |x| \times |x|}$)
Attention Transfer (KL-Divergence)
Value-Relation Transfer (KL-Divergence)
Queries-Keys Scaled Dot-Product ($\mathbb{R}^{A_h \times |x| \times |x|}$)
Values-Values Scaled Dot-Product ($\mathbb{R}^{A_h \times |x| \times |x|}$)
Queries ($\mathbb{R}^{A_h \times |x| \times d'_k}$)
Keys ($\mathbb{R}^{A_h \times |x| \times d'_k}$)
Values ($\mathbb{R}^{A_h \times |x| \times d'_k}$)
Last Layer Self-Attention Vectors
Transformer Block M
...
Transformer Block 3
Transformer Block 2
Transformer Block 1
x_1 x_2 x_3 x_4 x_5
Student

Figure 2.5 The Overview of deep self-attention distillation (Wang et al, 2019)

2.2.4.9 Electra: Replaced Token Detection Approach

Clark et al. (2020) introduced the ELECTRA model, which presents a novel approach to pre-training language models. It involves two transformer models: a generator and a discriminator. The generator modifies tokens in a text sequence and is trained like a masked language model. The discriminator, the primary focus of this approach, aims to discern which tokens have been altered by the generator (Clark et al., 2020).

Unlike traditional Masked Language Modeling (MLM) methods (e.g., BERT), which mask some tokens and train the model to predict them, ELECTRA uses a 'replaced token detection' task. This method is more efficient as it involves corrupting input by substituting certain tokens with plausible alternatives from the generator and then training a discriminative model to identify whether each token in the corrupted input is original or replaced. This approach is found to be more sample-efficient than MLM, covering all input tokens rather than just a subset, leading to better performance with the same model size, data, and compute resources.

According to Clark et al. (2020), ELECTRA's model structure is similar to BERT, with the key difference being in the separation of embedding size and hidden size; the embedding size is generally smaller while the hidden size is larger. An additional projection layer is used to adjust the sizes, except when they are equal. The pretraining involves alternating training steps between the small masked language model (the generator) and the discriminator.

2.2.5 Fine-tuning

Fine-tuning in machine learning involves refining a pre-trained model on a smaller, specific dataset to adapt its learned features to specific tasks. This technique, particularly prevalent in NLP, allows the model to perform specialized tasks without the need for training from scratch. Such efficiency not only preserves resources but also utilizes less data while maintaining or enhancing performance. Key studies illustrating this include Howard and Ruder (2018), who introduced Universal Language Model Fine-tuning (ULMFiT), demonstrating significant text classification improvements by fine-tuning on task-specific datasets. Further research by Reddy (2023) and Sun et al. (2019) explores fine-tuning on various tasks, showing that even limited adjustments to critical model layers can yield state-of-the-art results. Radiya-Dixit and Wang (2020) further confirmed that targeting only essential layers for fine-tuning could maximize efficiency, while Hu et al. (2023) and Huang et al. (2022) discuss enhancing models' reasoning capabilities through meta-learning and self-improvement strategies.

2.3 Question Answering in Low-Resource Domains

This section will explore the challenges and strategies associated with low-resource QA, and examine the impact of data augmentation on enhancing QA capabilities.

2.3.1 Challenges and Strategies for Low-Resource QA

In low-resource settings for Machine Reading Comprehension (MRQA), extensively annotated data are scarce, especially in specialized domains. This scarcity is due to the unique nature of these domains and the high costs of expert annotations. Augmentation through Language Models (LM) fine-tuned for generating question-answer pairs becomes essential in these contexts. It enriches limited datasets, enhancing the model's domain-specific understanding and prediction capabilities. Without such augmentation, the model's performance in accurately answering specialized questions is compromised.

Schmidt et al. (2022) explore an innovative approach to enhance low-resource question answering by combining data augmentation through question-answer generation with Active Learning. They emphasize the importance of incorporating human expertise early in the model training process, especially in scenarios where annotated data are limited but unlabeled documents are abundant. This approach aims to improve performance in low-resource, domain-specific settings and reduce the human annotation effort. Their experiments investigate how human annotation affects the performance of QA systems across various domains, revealing that early integration of human input significantly boosts system performance in specialized areas. This research offers valuable insights for developing QA systems in new domains with minimal labeling effort (Schmidt et al., 2022).

Tran et al. (2022) address the challenges of building a Question Answering System (QAS) for Vietnamese, a language considered low-resource due to limited linguistic data and tools. Their work focuses on developing a closed-domain QAS, which involves creating two datasets: vi-SQuAD v1.1 and HUFI-PostGrad, the latter being manually collected. The system employs two primary models: Intent Classification and Machine Reading Comprehension. Their experiments yielded encouraging results, demonstrating the feasibility of creating effective QAS for low-resource languages like Vietnamese (Tran et al., 2022).

2.3.2 Role of Data Augmentation in Enhancing QA Capabilities

Data augmentation plays an important role in improving the performance and robustness of QA systems. It is critical to differentiate between meaning and form in text data. This is essential for the development of deep learning models in NLP (Shorten, Khoshgoftaar, & Furht, 2021).

The research by Barmawi and Muhammad (2019) introduces an approach to paraphrasing by incorporating contextual synonym substitution. Traditional methods often lead to ambiguous meanings due to a lack of contextual understanding. Their method significantly enhances the fluency of paraphrased sentences. By considering both the n-gram structure and the broader sentence context, the chosen synonyms fit more seamlessly into the new sentence, retaining the original meaning while improving readability. This advancement in paraphrasing demonstrates the importance of context in NLP, moving beyond simplistic word-for-word replacement to a more sophisticated understanding of language nuances.

Wei and Zou's (2019) introduction of Easy Data Augmentation (EDA) techniques mark a significant contribution to text classification tasks. EDA, which includes synonym replacement among its methods, is especially beneficial for small datasets. The technique's ability to achieve comparable accuracy with only half the training data, as opposed to using the entire dataset without augmentation, underscores its efficiency. EDA stands out for its simplicity and effectiveness, encompassing operations like synonym replacement, random insertion, swap, and deletion. This approach has been shown to significantly boost the performance of both convolutional and recurrent neural networks in text classification. The success of EDA highlights the impact that straightforward data augmentation methods can have on improving the capabilities of machine learning models in understanding and classifying textual content.

2.4 Data Augmentation Techniques

Recent studies have made significant advancements in the application of data augmentation, particularly paraphrasing and synonym replacement, in QA systems. Gan and Ng (2019) focused on enhancing QA model robustness against paraphrased questions, employing neural models for data augmentation. In the context of natural language QA systems, Oh et al. (2015) developed paraphrased sentences based on synonym knowledge for Korean language processing. Soni and Roberts (2020) explored paraphrasing methods to enhance QA in electronic health records, indicating the broad applicability of these techniques. Furthermore, Pappas et al. (2022) investigated multiple data augmentation methods, including word substitution, in biomedical factoid QA systems, highlighting the diversity of applications in the field.

Zhang et al. (2020) presented a technique of text data augmentation that involves replacing words or phrases with their synonyms from a thesaurus. This approach can quickly generate large amounts of data, particularly useful for tasks like text classification. The main advantage of this technique, as noted by Shorten et al. (2021), is that it provides a straightforward way to increase the diversity of language in a dataset without changing the underlying meaning of the text.

The use of word embeddings for text augmentation, as explored by Wang and Yang (2012), involves using techniques like k-nearest-neighbor (KNN) and cosine similarity to identify similar words for replacement. Traditional word embeddings like word2vec (words to vector), GloVe (Global Vectors for Word Representation), and fasttext are employed to perform similarity searches. This method has shown relative improvement compared to baselines without data augmentation. By leveraging these embeddings, the model can capture semantic relationships between words and use these relationships to generate text variations that maintain the core content while varying the expression (Wang & Yang, 2012).

Abdurrahman and Purwarianti's (2019) study on synonym-based text augmentation in Indonesian text classification provides another layer of depth to this field. They employed a two-step process involving the determination of the number of words to be substituted in a sentence and the subsequent selection of appropriate synonyms. Their method is distinguished by its use of an augmentation degree to decide the extent of synonym substitution, combined with a language model to ensure the selected synonyms are contextually appropriate. Notably, their research found that a 5-gram neural model excelled in selecting synonyms, leading to a notable improvement in text classification performance.

A study by Zhang et al. (2020) tailored Pegasus (Pre-training with Extracted Gap-sentences for Abstractive Summarization) model, which is a transformer-based model developed by Google

Research, for abstractive text summarization. Unlike traditional transformer models, Pegasus is unique in its pre-training method: it learns to generate summaries by being trained on a task where sentences are omitted from a text and then recreated by the model. For tasks like paraphrasing, Pegasus can be fine-tuned or used directly due to its advanced capabilities in interpreting and crafting human-like text. It excels in paraphrasing because it does not just replace words with synonyms; it can restructure sentences while maintaining their original intent. This skill stems from its training focused on summarization, equipping it to adeptly handle the nuances of paraphrasing (Zhang et al., 2020).

Data augmentation methods have been successfully applied in areas such as paraphrase detection in short texts (Shakeel et al., 2019). In QA systems, data augmentation through paraphrasing, as demonstrated by Shakeel et al. (2019), plays a key role in enhancing system performance. By creating varied versions of questions while maintaining their meaning, these systems learn to recognize and correctly respond to different phrasings of the same query. This technique, particularly vital in limited data scenarios, enriches training datasets, helping the system better grasp language nuances and improve response accuracy.

2.5 Sentence Transformers and Their Applications

Sentence transformers are employed in QA systems to understand and generate natural language responses. They convert sentences into meaningful embeddings, capturing contextual information crucial for accurate response generation. These models have significantly advanced the capability of QA systems to handle complex language patterns and generate contextually relevant answers.

Thomas Wolf et al. (2019) presented the transformative impact of transformer architectures in NLP, with a particular focus on HuggingFace's Transformers library. This library has become a foundation in the NLP community, offering access to a wide range of state-of-the-art pre-trained transformer models. These models are integral to handling intricate language patterns for generating contextually pertinent responses in QA systems. The HuggingFace Transformers library, with free open-source access to advanced NLP tools, has played an important role in facilitating the development of more sophisticated and efficient QA systems. It allows for the deployment of models that can adeptly navigate and interpret the complexities inherent in human language, thereby enabling more accurate and contextually relevant answers in QA scenarios.

Nils Reimers and Iryna Gurevych (2019) present an innovative adaptation of the BERT model, termed Sentence-BERT (SBERT). This modification addresses a significant computational challenge posed by the original BERT framework in processing sentence-pair regression tasks like semantic textual similarity (STS). Traditional BERT models require that both sentences of a pair be processed together, which leads to substantial computational demands, especially when comparing large sets of sentences.

To overcome this limitation, SBERT incorporates Siamese and triplet network structures into the pre-trained BERT network. These structures are adept at generating sentence embeddings that are not only semantically rich but also comparable using cosine similarity. This approach marks a considerable advancement in terms of computational efficiency. While BERT could take upwards of 65 hours to find the most similar sentence pair in a collection of 10,000 sentences, SBERT reduces this to a mere 5 seconds, significantly streamlining the process without compromising the accuracy inherent to the BERT model.

Yang et al. (2021) presented a novel approach, BECR (BERT-based Composite Re-Ranking), to address the relevance-efficiency trade-off in transformer-based ranking models for ad-hoc document search. The central challenge that BECR addresses is the high computational cost associated with BERT models, particularly when used in online inference for document ranking.

The research conducted by Liello et al. (2022) presented a significant advancement in the field of NLP, specifically for QA systems. Their study introduced novel sentence-level pre-training objectives for transformer models, aimed at enhancing Answer Sentence Selection (AS2). AS2 is a critical task in QA systems where the model must identify and select sentences containing or constituting the answer from a set of relevant documents. By incorporating paragraph-level semantics within and across documents, their approach markedly improved the ability of transformers to understand and contextualize sentences in a broader narrative framework. This methodological innovation in pre-training transformers demonstrated its efficacy in AS2 tasks, highlighting the potential of fine-tuned transformer models in processing complex language structures and accurately identifying relevant information in QA systems.

Wang et al. (2022) address a critical aspect of NLP related to the evaluation of word and sentence embeddings. The paper first identifies problems with using semantic similarity as the standard for evaluating word and sentence embeddings. The authors propose a new evaluation method, EvalRank, which they claim shows a stronger correlation with downstream tasks than traditional methods. This approach is designed to provide a more accurate reflection of an embedding model's utility in practical applications.

This work is significant as it challenges the conventional methods of evaluating word and sentence embeddings in NLP and offers an alternative that may lead to more accurate and useful models. By rethinking evaluation strategies, the paper contributes to the ongoing development and refinement of NLP technologies.

The literature review outlines the advancement of QA systems from traditional models to neural architectures utilizing transformer models for better accuracy and efficiency. The research aims to explore and compare pre-trained language models for application to QA downstream tasks in a low-resource domain. Also, it notes existing gaps in understanding the impacts of synthetic data, meta-data, and data augmentation in low-resource domains.

3. Methodology

This chapter presents the methodology used to develop and evaluate a QA framework specifically tailored for a low-resource domain, which is the core objective of this research (see Figure 3.1).

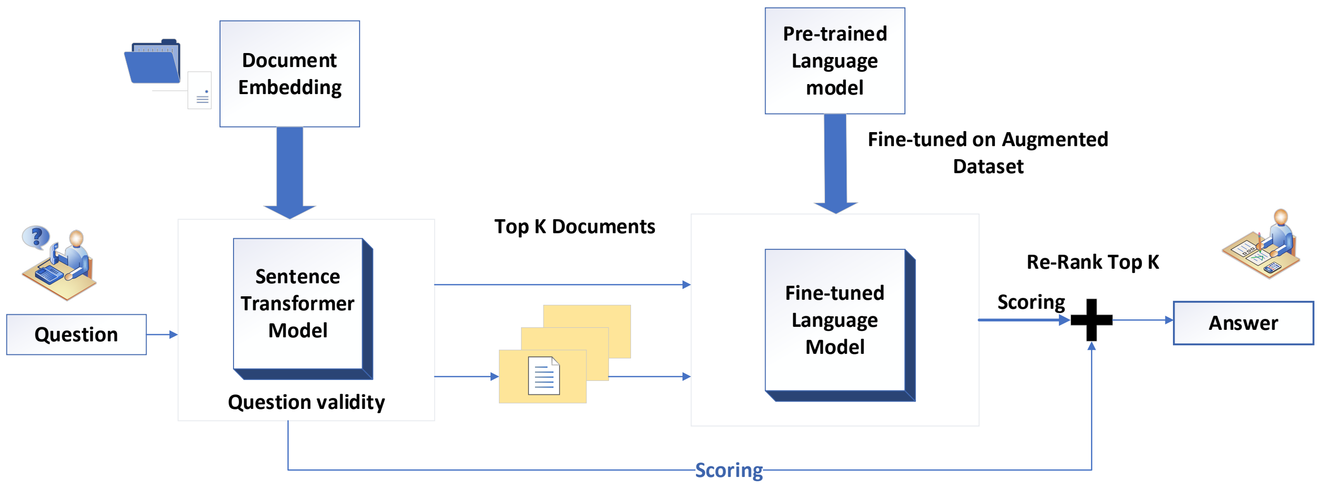

Document Embedding
Pre-trained Language model
Fine-tuned on Augmented Dataset
Top K Documents
Sentence Transformer Model
Question
Question validity
Fine-tuned Language Model
Re-Rank Top K
Scoring
Answer
Scoring

Figure 3.1 QA Framework in Low-resource Domain

The proposed framework is based on a traditional QA system presented by Malik et al. (2013) to address challenges in low-resource domains, utilize ANN-based language models, and improve the efficacy of the QA system for a cloud-based user interface. The study investigated the role of synthetic data and data augmentation techniques, language model selection, and methods to improve QA system performance.

3.1 Overview of the Framework for Building a QA System

A QA system must comprehend the question's intent and offer a pertinent, correct response. The traditional QA system architecture consists of three key components: question processing, document processing, and answer extraction (Malik et al., 2013). In this architecture, question processing involves analyzing and classifying user questions to determine their focus and type, with automatic classification methods available. Questions and document processing will result in selecting relevant script descriptions and extracting phrases or sentences based on the question's focus. The retrieved data will be ranked according to their relevance (Calijorne Soares & Parreiras, 2020).

Integrating transformer-based language models into traditional architectures marks a significant advancement by enhancing the handling of human-formulated, free-format questions. These models excel in question analysis and response due to training on large datasets, which prepares them to adeptly interpret inquiries. Fine-tuning these pre-trained models for specific domains optimizes their performance by tailoring them to domain-specific characteristics. This targeted fine-tuning significantly improves the capability of QA systems to process complex questions effectively (Nassiri and Akhloufi 2022).

In this study, implementing a transformer-based QA system requires selecting an appropriate language model and performing comparative analyses of pre-trained language models and their fine-tuned variants across different texts and contexts. The performance of these models varies based on the corpus and context, highlighting the importance of evaluating them under various conditions to identify their strengths and weaknesses. This helps in choosing the most suitable model for a specific domain. Additionally, comparing fine-tuned models to their original versions is critical to assess their enhanced performance and adaptability in domain-specific, low-resource scenarios. Such evaluations are essential for understanding improvements in model capabilities after fine-tuning (Talmor et al., 2019; Qiu et al., 2020). The process is shown in Figure 3.2.

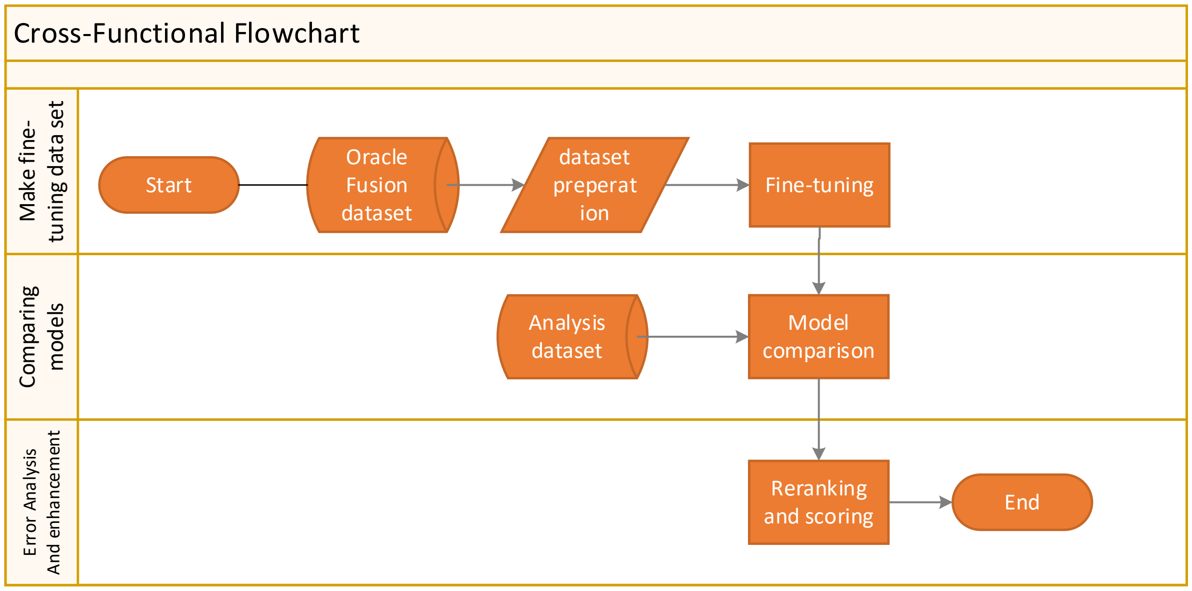


Figure 3.2 Cross-functional flowchart of the methodology

The development and evaluation of the QA framework require two datasets. The first dataset is used for fine-tuning pre-trained language models in a specific domain, while the second dataset evaluates the models' performance before and after fine-tuning. The selection of the ultimate model will be based on the evaluated metrics.

3.2 Building a Dataset for Fine-tuning

This study began with the preparation of a domain-specific dataset for fine-tuning pre-trained language models for QA tasks.

3.2.1 Manually Building QA

The fine-tuning dataset was created due to the absence of an existing collection of relevant QA datasets, by generating question-and-answer pairs from a software manual (Joshi, 2020) as the low-resource domain. This method involved selecting sections such as short paragraphs or tables, then having humans formulate corresponding questions and extract answers, ensuring alignment between the questions and the manual content.

3.2.2 Paraphrasing and Synonym Replacement

To enrich the dataset with diverse linguistic expressions, two data augmentation techniques were utilized: paraphrasing and synonym replacement. Paraphrasing was done using the Pegasus model fine-tuned for paraphrasing (Zhang et al., 2020) from the Hugging Face library. It interpreted the context of the given sentence and produced an output that, while semantically aligned with the original sentence, differed syntactically.

For synonym replacement, the Global Vectors for Word Representation (GloVe) (Pennington et al., 2014) model was applied, which used word embeddings to capture semantic and syntactic patterns by analyzing co-occurrence in text corpora. This method enriched the dataset by introducing lexical diversity and maintaining semantic links between words, producing synthetic data ideal for the fine-tuning task. These steps are shown in Figure 3.3.

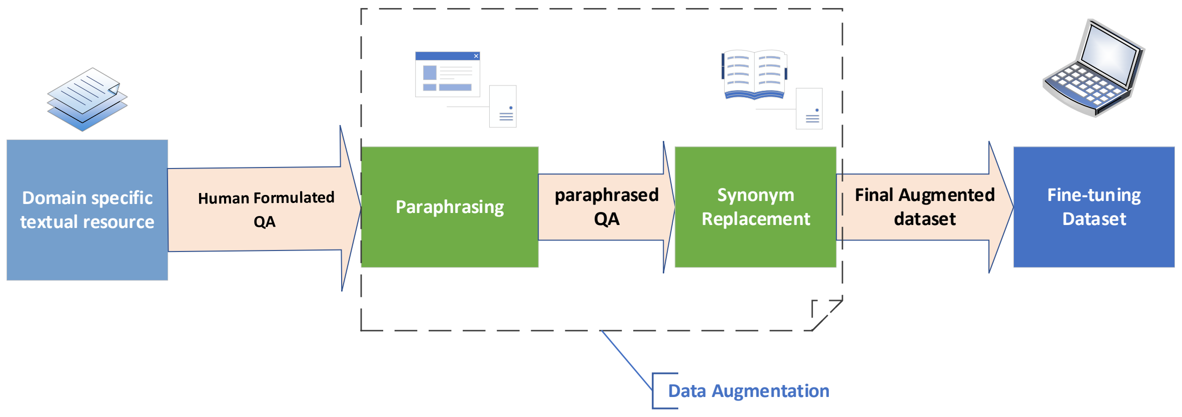


Figure 3.3 Creating fine-tuning dataset

3.3 Building Question (Evaluation) Dataset

The comparative analysis of models involved using a synthetic dataset to evaluate and test how pre-trained and fine-tuned models responded to queries. A synthetic question-and-answer dataset will test the models' effectiveness, simulating real-world scenarios relevant to the target domain.

The dataset is created by generating questions based on descriptions of script functionality specific to the QA system's domain.

3.3.1 Manual Question Generation from Script Descriptions

In the first phase of manual question generation, questions were carefully crafted based on the functionalities described in the script descriptions. A systematic strategy is employed here, focusing on creating 'wh' questions by using ‘what’, ‘why’, and ‘how’. This involved taking sentences that describe a function and transforming them into questions by starting with these 'wh' words.

3.3.2 Automated Question Generation Using AI Tools

Zhang (2019) and Goodfellow (2019) emphasized the need for large, varied datasets to effectively evaluate models, as size and diversity contribute to more accurate and comprehensive assessments. To expand the dataset, a Text-to-Text Transfer Transformer (T5) was employed for automated question generation from descriptions of script functionality. These questions were combined with the ones manually created. Questions of both types were then paraphrased using the Pegasus model to introduce different expression styles, further enriching the dataset, as outlined in Figure 3.4.

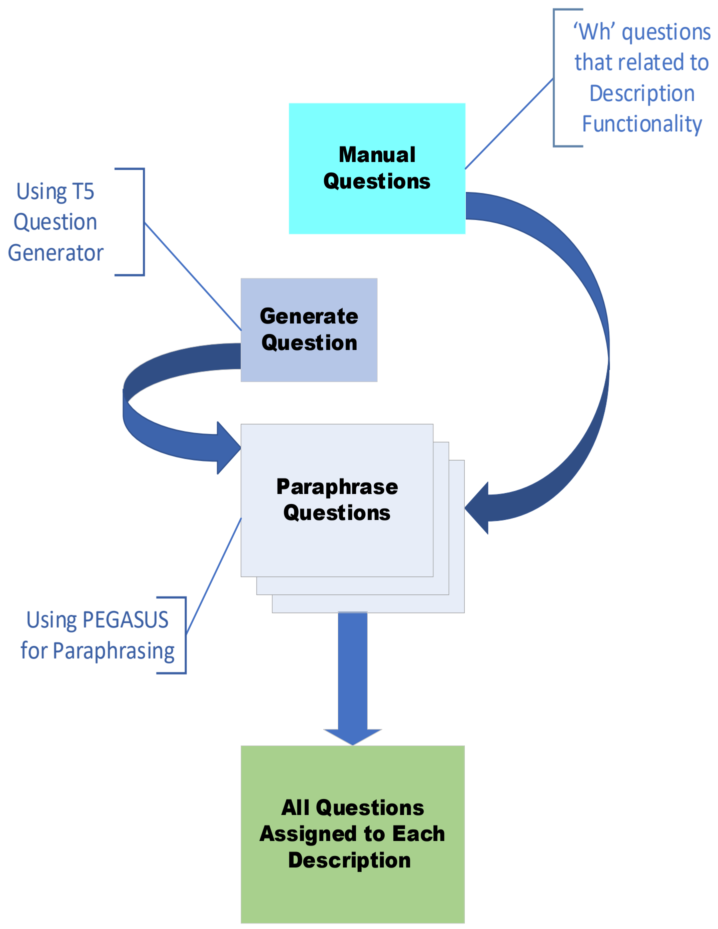

'Wh' questions that related to Description Functionality
Manual Questions
Using T5 Question Generator
Generate Question
Paraphrase Questions
Using PEGASUS for Paraphrasing
All Questions Assigned to Each Description

Figure 3.4 Steps for creating the question dataset

3.4 Comparative Analysis of Pre-Trained Models

A comparative study of transformer-based language models from Hugging Face was conducted using an augmented set of question-answer pairs. The study evaluated how these models comprehended and responded to user queries by observing evaluation metrics before and after fine-tuning. A list of models was compiled based on their functionalities and computational demands. The models were evaluated based on their training loss, maximum exact match score, and maximum F1 score.

3.4.1 Selection of Models Based on Their Technical Characteristics

In this study, the implementation of a transformer-based QA system involved selecting pre-trained language models (PLMs) suitable for cloud-based applications. These models had to be compact, efficient, and responsive in real-time environments. The choice of PLMs was determined by balancing model size and data requirements, ensuring their effective performance in a cloud-based environment. Additionally, GPU resource availability influenced model choice to meet specific dataset requirements.

Model evaluation was guided by Wolpert's 'No Free Lunch' theorem. This theorem, established by David H. Wolpert in 1996, states that no single machine-learning algorithm excels universally across all tasks and datasets. Effectiveness was determined by the specific task and dataset, requiring exploratory computational experiments for model assessment. It was empirically investigated whether the generalized strengths of PLMs are applicable in specific data settings or if domain-specific fine-tuning offers substantial benefits over standard applications, assessing the potential advantages of tailored fine-tuning.

3.5 The Evaluation Metrics

After fine-tuning, the model's performance was evaluated using a validation set, supplemented by an additional small dataset comprising 100 instances from the validation set. The predictions made by the pre-trained model on this smaller dataset were used for comparison. The effectiveness of the model was measured using the F1 score and exact match. The F1 score provides a harmonic mean of precision and recall, reflecting the model's accuracy using the formula (3):

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

In this study, precision (4) was defined as the proportion of correct answers (true positives) out of all answers predicted by the model (true positives and false positives). Recall (5) measured the proportion of correct answers identified by the model out of all actual correct answers in the dataset (true positives and false negatives). The formulas are shown below:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{4}$$

$$Recall = \frac{True\ Positive}{False\ Negative + True\ Positive} \tag{5}$$

The exact match score indicated the percentage of predictions that exactly matched the true answers out of all answers. This metric is a straightforward way of evaluating the accuracy of predicted strings against reference strings. It assigned a score of 1 for a perfect match and 0 for any deviation, regardless of its extent.

The formula can be expressed as:

$$Exact\ Match\ = \left(\frac{\text{Number of Exact Matches}}{\text{Total Number of Instances}}\right) \times 100 \tag{6}$$

The model outputs logits for the start and end positions of the answer in the input IDs. In the post-processing step, the logits were converted into probabilities using a softmax function. We then attributed a score to each (start_token, end_token) pair by taking the product of the corresponding two probabilities. The pair with the maximum score that yields a valid answer (i.e., a start_token lower than end_token) was selected. However, in our optimized process, we skipped the softmax step and use logit scores directly, summing the start and end logits. This approach sped up the process by only considering the highest n_best logits (with n_best=20).

To demonstrate this, we generated some predictions using the default model for the QA pipeline on a small part of the validation set. We preprocessed the validation set with the appropriate tokenizer and used the model to obtain predictions. The predicted answers were then evaluated using the exact match and F1 score metrics, which provide a comprehensive assessment of the model's performance.

3.6 Fine-tuning the models

Fine-tuning a pre-trained model for a specific NLP task means adjusting its weights to reflect domain-specific lexical constructs. This process indicates that although pre-trained models provide a strong base of general linguistic knowledge, they often require further customization to perform optimally in domain-specific contexts (Devlin et al., 2018).

A method known as supervised fine-tuning was used to fine-tune models on a QA task using Hugging Face. Initially, the QA dataset mentioned in section 3.2 was clearly labeled. The questions and contexts were tokenized and formatted with the specific model's tokenizer to fit the model requirements. Specific configurations and settings were set, and then the models were fine-tuned on this dataset, with their weights adjusted to improve answer prediction based on the specific data.

This format is typically a list of dictionaries, with each dictionary containing two key-value pairs: one for the ID of the example (question) and the other for the predicted text (answer). Similarly, the theoretical (true) answers are arranged in a parallel format: a list of dictionaries, each with an ID key and an answers key. The answers key contains the actual answers, aligning with the predictions for comparison.

The fine-tuning process was implemented using PyTorch on a Google Colab GPU. After saving the fine-tuned model, it was used for further evaluations, especially in comparison to other models and those not fine-tuned with the question dataset.

3.7 Comparing Pre-trained Models and Their Fine-tuned Variants

In the comparison section, the focus was on evaluating models that had been fine-tuned. Pre-trained and fine-tuned models were compared based on exact match (6) and the accuracy defined in the following formula:

$$\text{Accuracy} = \frac{\text{Number of Correct Answers}}{Total\ Number\ of\ Answers\ Given} \tag{7}$$

After fine-tuning, the models were tested on the augmented question-answer dataset described in section 3.3. This comparison primarily involved analyzing how each model responds to the questions. The effectiveness of each model was quantitatively evaluated by comparing the accuracy of their predictions against the actual correct answers. To evaluate the accuracy of QA model predictions, the similarity between predicted and actual answers was quantified using cosine similarity of their TF-IDF vectors, which ranges from -1 (entirely different) to 1 (identical). The results, including both questions and answers along with their similarity scores, were compiled into a DataFrame for comprehensive analysis. These data were used to assess how often predictions completely miss the mark (zero similarity) compared to being fully correct (non-zero similarity).

For this analysis, Python libraries such as Pandas and NumPy were employed, facilitating a detailed assessment of both the pre-trained and fine-tuned variants of each model.

3.8 Framework Enhancement

The final phase of the study improved the efficiency of extractive QA systems, which typically slow down due to the need for extensive context scanning in time-sensitive scenarios. This was achieved by implementing an information retrieval strategy that ranks semantic similarity, assessing texts by meaning rather than just words (Hammad et al., 2021). The QA framework, detailed in Figure 3.5 and studies by Matsubara et al. (2020), consists of:

1. **Answer Retriever**: This component uses cosine similarity to identify potential answer candidates from a data corpus.
2. **Closed-Domain Extractive QA model:** The selected answers were then processed by an extractive, close-domain QA model, which has been fine-tuned for the specific domain.

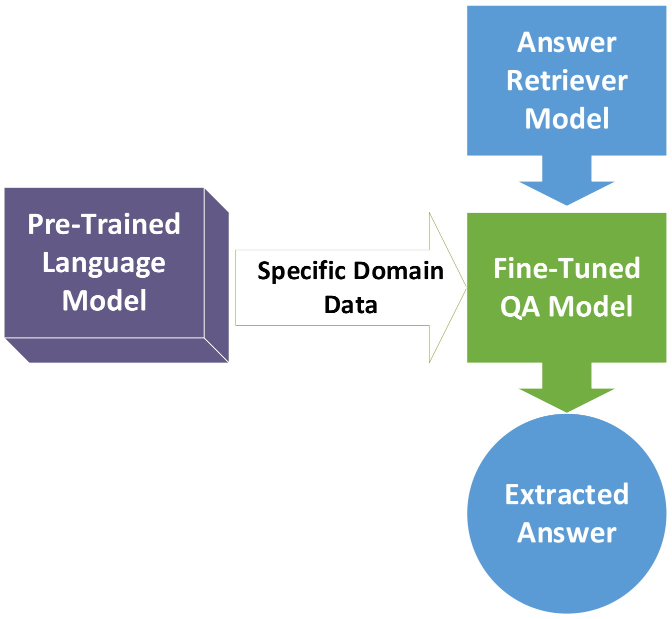


Figure 3.5 The architecture of the QA framework

The architecture uses a Sentence Transformer model for answer retrieval, as developed by Nils Reimers & Iryna Gurevych (2019). It evaluated the semantic similarity of BERT embeddings of questions and contexts to efficiently rank them for answer extraction. The top-ranked contexts were fed to the fine-tuned model to re-rank the top 3 and top 5 choices to enhance accuracy in selecting the correct answer, as illustrated in Figure 3.6.

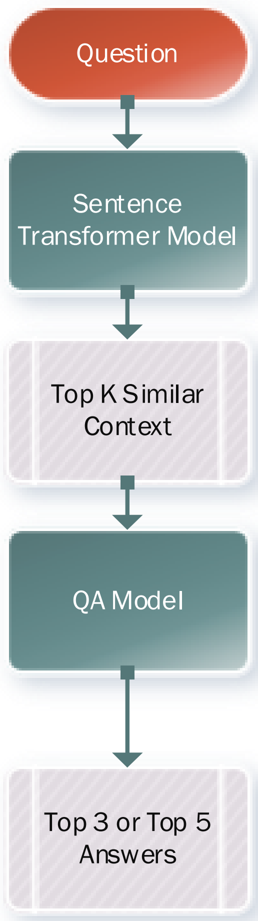


Figure 3.6 The process of ranking answers

4 Results of Computational Experiments

4.1 Experimental Setup

The computational experiments were conducted on Google Colab, a cloud-based platform equipped with NVIDIA T4 GPUs and similar computing resources.

The analysis utilized the Transformers library from Hugging Face, tailored for NLP tasks, along with TensorFlow and PyTorch libraries known for their comprehensive features and compatibility with Google Colab. The Colab environment also efficiently managed data handling, preprocessing, and storage, optimizing memory use and cloud storage.

4.2 Data Preparation and Preprocessing

4.2.1 Dataset Description

A systematic method was used to build the dataset as outlined in Section 3.2, starting with 550 question-answer-context pairs. These pairs were taken from the software manual (Joshi, 2020). The dataset was augmented to 1100 pairs using synonym replacement and further to 2200 pairs through paraphrasing.

The data were then formatted into a JSON file where each pair was given a unique identifier and an indicator of the answer token's start position within the context. The dataset was divided into a training set with 1760 entries and a validation set with 440 entries.

4.2.2 Preprocessing Training Set

The preprocessing of the training set involved converting text into a numerical format that the model could process, using the AutoTokenizer from the Hugging Face Transformers library. Texts were transformed into numerical identifiers, or input IDs, with special tokens to differentiate sections like questions and contexts. For instance, ‘learning’ and ‘machine’ might be assigned IDs of 8573 and 1124, respectively.

The structured input for a question-answering task generally appears as “[CLS] question [SEP] context [SEP]”, where “[CLS]” marks the start and “[SEP]” separates the question and context. If a context exceeded the maximum length (384 tokens), it was managed by dividing it into overlapping segments with a sliding window technique. These input IDs were for the model to process and predict answers, pinpointing start and end positions within the context. Special labeling ensured accurate predictions even when the context did not fully include the answer, aiming to predict correct start and end positions as depicted in Figure 4.1.

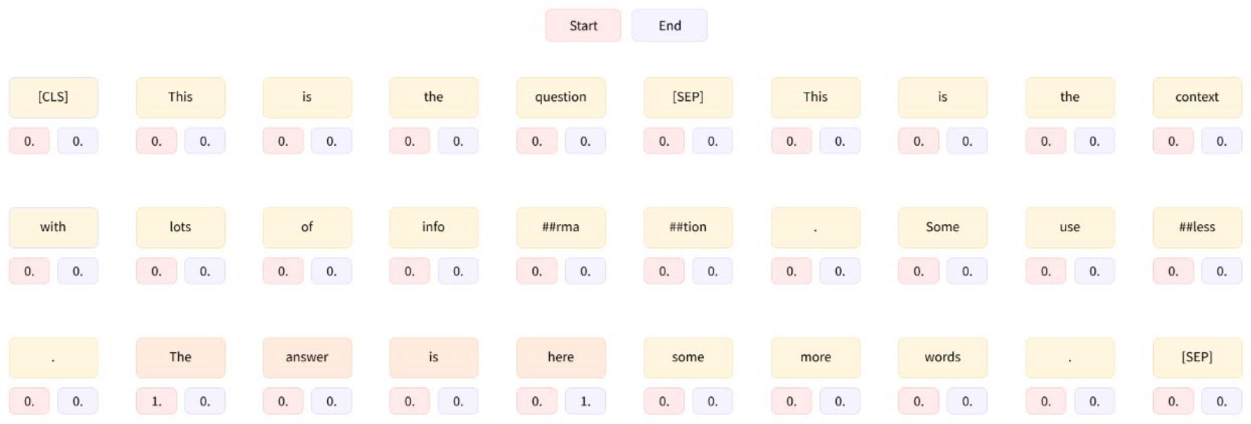


Figure 4.1 The answer start and end token (Hugging face, 2024)

4.2.3 Preprocessing Validation Set

Preprocessing the validation set is relatively more straightforward compared to the training set. The main aspect of processing the validation set lies in interpreting the model's predictions back to the spans in the original context. This required retaining the offset mappings, which help track the location of words in the original text, and maintaining a reference link to associate each processed feature with its originating example. Utilizing the unique ID column present in the original dataset facilitated this linkage.

Fine-tuning a pre-trained language model with the Trainer API is a streamlined process that involves several steps. Firstly, using the class 'AutoModelForQuestionAnswering' automatically adjusted the pre-trained model to perform the QA task by focusing on predicting the correct start and end positions of answers in a given text passage. This required processing the model's initial predictions, known as logits. The aim was to select the most sensible combination of these points, ensuring the start was before the end and the answer length was appropriate. The model's effectiveness was then evaluated by comparing its selected answers against the actual correct answers, using the evaluation metric mentioned in section 3.5. This metric scored the model based on the accuracy of its predictions through exact match and F1 score. To facilitate understanding and benchmarking, a pre-trained model was initially used on a small portion of the data. This approach allowed for a preliminary assessment of the process and expected outcomes.

4.3 Model Configuration and Fine-tuning

The fine-tuning was conducted using the Trainer class from the Hugging Face Transformers library, with specific configurations set through TrainingArguments. The fine-tuning typically involved 3 to 5 training epochs to optimize model performance while preventing overfitting by using early stopping. Key settings included a learning rate of 2e-5 and a batch size of 16. Fine-tuning usually completed in about 1 hour on the T4 GPU of Google Colab.

After setting the configuration, the model underwent fine-tuning, adjusting its pre-trained weights to better match the patterns in the dataset. For evaluation, the format of the predicted answers was standardized to a list of dictionaries; each dictionary included an ID for the question and the predicted text for the answer. This setup was mirrored by the format of the true (theoretical) answers, which also appeared as a list of dictionaries but included an actual answers key for direct comparison with the model's predictions.

4.4 Evaluation Metrics

The evaluation of the fine-tuned model was conducted using the 'squad' metric from Huggingface's evaluate library, where the model's predictions were compared against reference answers. Predictions were structured to include an ID and the predicted text, along with the evaluation metrics.

For most models, this fine-tuning process involved adjustments in pre-trained settings, but MobileBERT differed as it did not require fine-tuning from a trained teacher model or additional data augmentation. Instead, MobileBERT generally needed a higher learning rate and more training epochs than BERT for effective fine-tuning.

4.5 Results

The evaluated metrics of the selected fine-tuned language models used in the study are shown in Table 4.1. Figure 4.2 presents the comparison of these models.

Table 4.1. Comparison of fine-tuned pre-trained models

Model	Training Loss (Last Epoch)	Exact Match (Max)	F1 Score (Max)
Bert-base	1.9536	5.0	47.1795
Distilbert	2.0092	4.0909	46.4128
Albert-base-v2	1.6613	6.1364	47.2541
Bart	2.1706	4.0909	46.8734
Mobilebert	N/A	0.6818	41.2806
Electra-small	2.1860	5.2273	46.3498
Roberta	2.0791	5.2273	47.1992
T5	4.9266	2.0455	36.4841
Minilm	2.7341	2.7273	48.2355

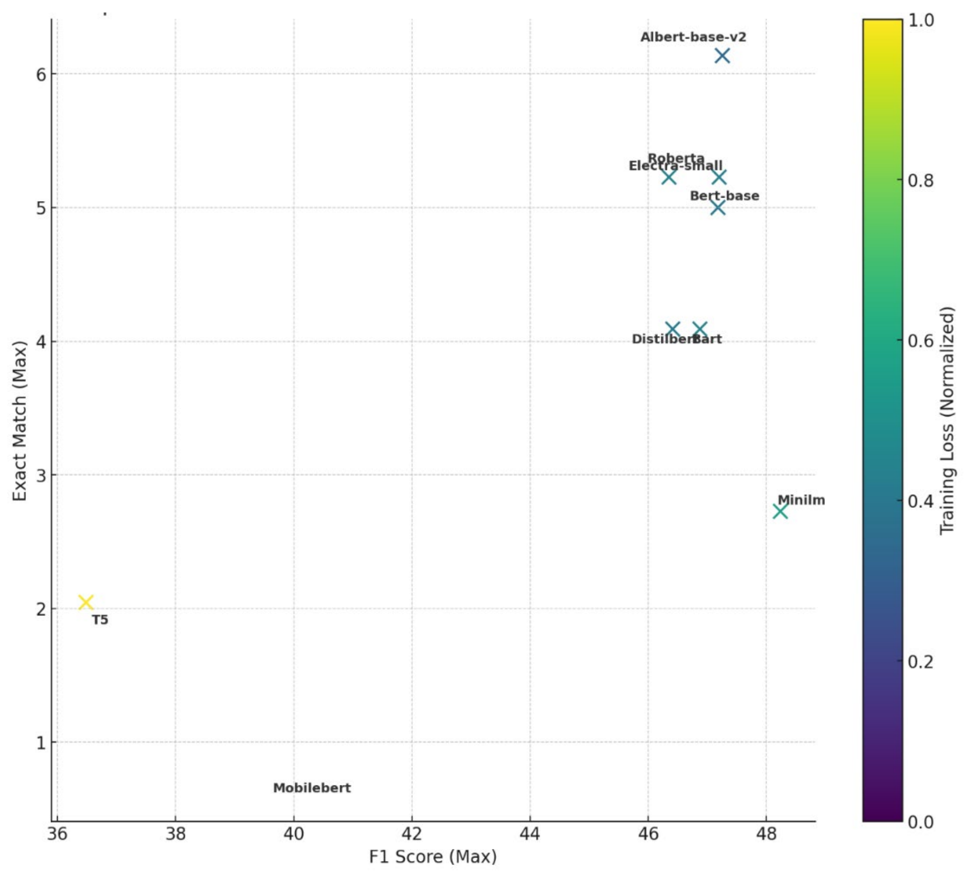


Figure 4.2 Relationship between F1 score and exact match models

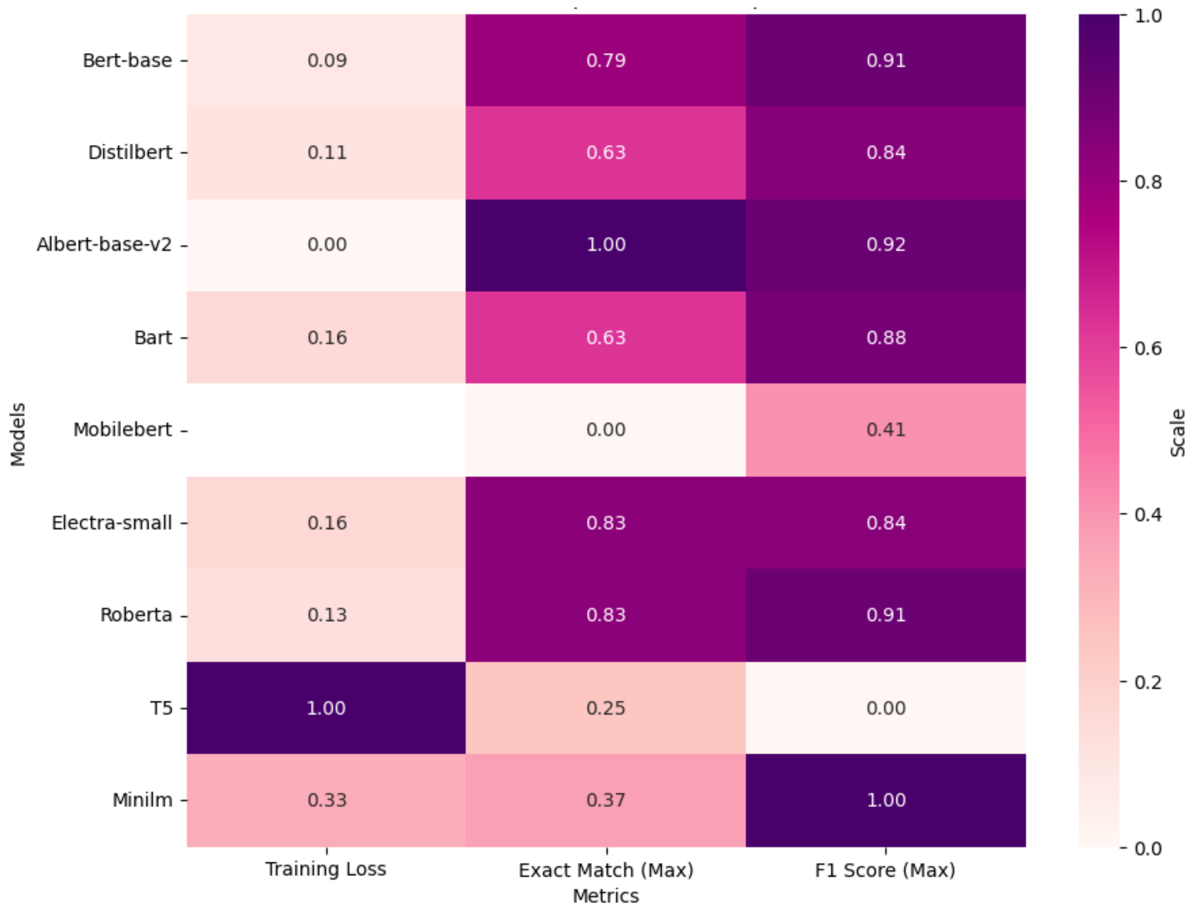


Figure 4.3. Fine-tuned model comparison heatmap

Table 4.1 describes performance metrics of nine pre-trained language models following fine-tuning. Albert-base-v2 reported the lowest training loss at 1.6613, suggesting optimal fitting among the models evaluated. On the other hand, T5 displayed the highest loss at 4.9266, which might indicate issues such as potential overfitting or inefficiencies in its training. Also, Albert-base-v2 excelled again with the highest EM value of 6.1364%, demonstrating its superiority in generating precise answers under the evaluated conditions. Conversely, T5 lagged significantly behind in this metric, which could point to its difficulties in achieving precision, possibly due to the lack of targeted fine-tuning. Minilm led with an F1 Score of 48.2355, indicating it had the best balance between recall and precision among the models tested. T5 was at the bottom of the scale with an F1 Score of 36.4841. From the analysis, Albert-base-v2 and Minilm were highlighted as the top performers due to their efficiency and balanced accuracy, respectively as shown in Figure 4.3.

As shown in Figure 4.4, T5 appeared to underperform across all considered metrics. Moderate performances by models like Bert-base, Distilbert, Roberta, and Electra-small suggested they do not match the top-tier efficiency or accuracy of Albert-base-v2 or Minilm but are still viable depending on specific use cases.

To finalize the model selection, a comparison of the performance of fine-tuned models against pre-trained ones using the question (evaluation) dataset (section 3.3) was done. In this step, each question from the evaluation dataset was fed to the QA model, along with its corresponding context. The model's answers were then compared with the actual answers.

4.6 Pre-processing Question Dataset for Testing QA System Framework

Following the methodology outlined in section 3.3, a total of 2460 question answer pairs were developed. During the pre-processing stage, questions that lacked a test script name or context were removed, resulting in a refined dataset comprising 1769 questions. Each model generated its output in the form of a CSV file, which included both the predicted answers and the actual answers for comparison. To evaluate the accuracy of QA model predictions, the similarity between predicted and actual answers was quantified using the cosine similarity of their TF-IDF vectors, which ranges from -1 (entirely different) to 1 (identical).

This enabled us to conduct a detailed comparison between the predicted and actual answers. These data were used to assess how often predictions completely miss the mark (zero similarity) compared to being fully correct (non-zero similarity). The results of this comparative analysis are shown in Table 4.2.

Table 4.2. The result of comparing models with the question dataset

Model	Zero Similarity Instances	Non-zero Similarity Instances
Minilm (not fine-tuned)	1508	261
Minilm (fine-tuned)	103	1666
Albert Base (not fine-tuned)	567	1202
Albert Base (fine-tuned)	43	1726
Bart (not fine-tuned)	1502	267
Bart (fine-tuned)	5	1764
Bert (not fine-tuned)	1502	267
Bert (fine-tuned)	12	1757
DistilBert (not fine-tuned)	1680	89
DistilBert (fine-tuned)	643	1126
Electra Small (not fine-tuned)	1765	4
Electra Small (fine-tuned)	18	1751
Mobilebert	1500	269
Roberta (not fine-tuned)	583	1186
Roberta (fine-tuned)	66	1703
T5 Small	1769	0
T5 (fine-tuned)	1199	570

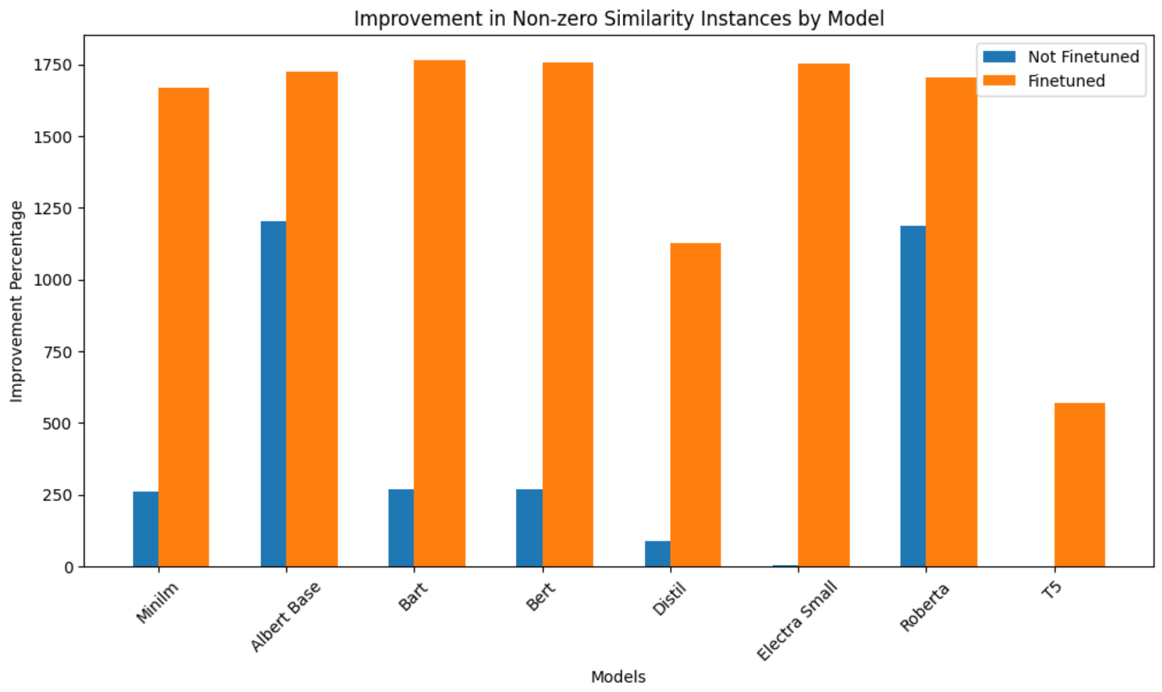


Figure 4.4. The result of comparison between the fine-tuned model and the pretrained model with question dataset

4.7 Discussion of Results

Table 4.2 reveals that fine-tuning markedly improves the efficiency of models on the question dataset. The table shows significant reductions in zero similarity instances for several models, indicating an enhanced ability to match expected answers. Notably, Bart, Albert Base, and Minilm saw dramatic declines in zero similarity rates (Minilm from 1508 to 103, Albert Base from 567 to 43, Bart from 1502 to 5) and increases in non-zero similarity instances. Bert and Distilbert models also improved, though Distilbert was less effective overall. Conversely, T5 Small without fine-tuning showed no non-zero similarity instances, although performance improved slightly after fine-tuning. Overall, the data underscores the importance of fine-tuning in boosting model accuracy and relevance in QA within specific domains, with Bart, Bert, and Albert Base showing significant enhancements. Figure 4.5 elaborates on the comparative analysis of fine-tuned models' metrics and responsiveness to the question dataset before and after fine-tuning

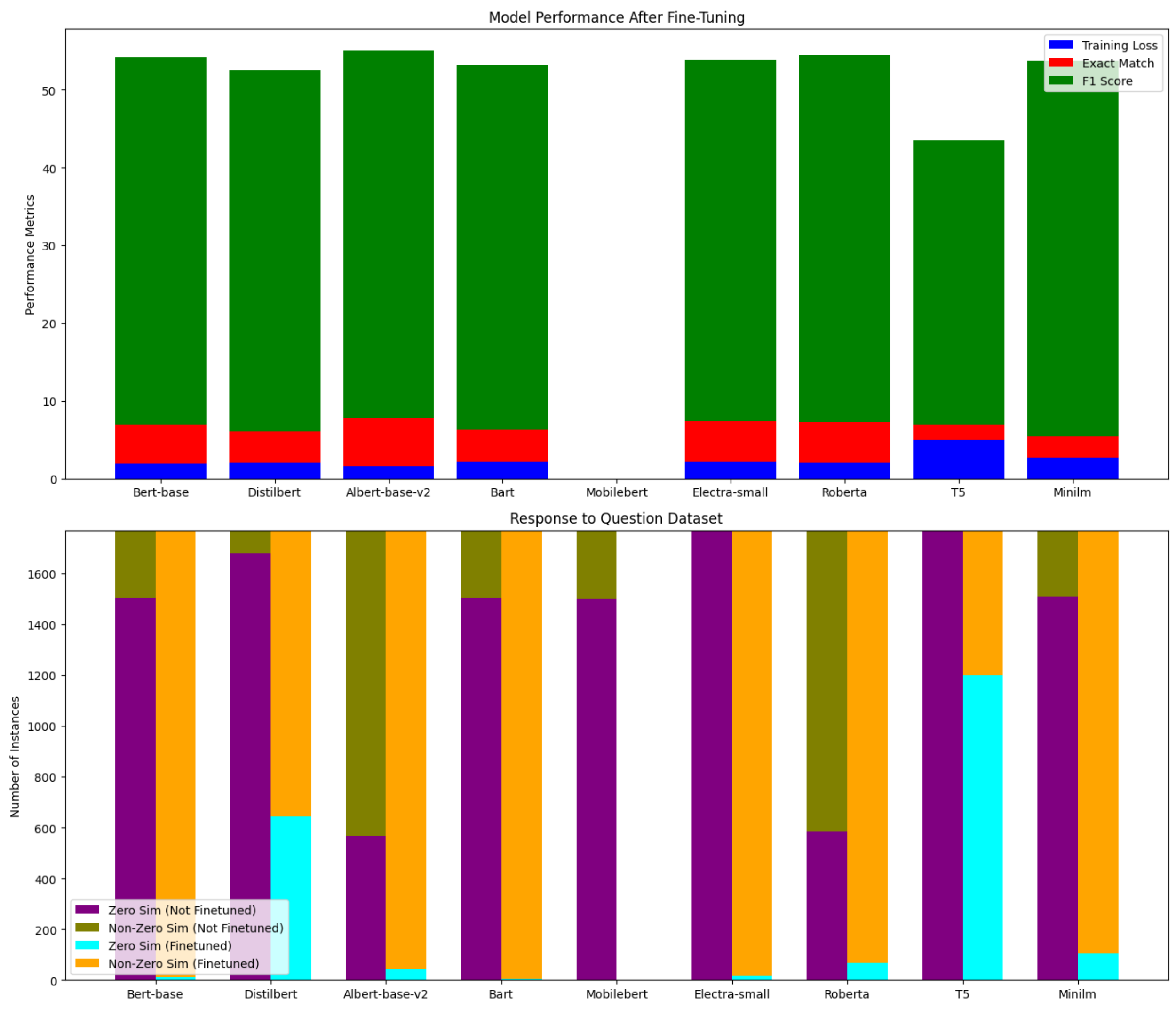


Figure 4.5 Comparative analysis of model's metrics and responsiveness to question dataset before and after fine-tuning

Our findings in the selected closed low-resource domain demonstrated significant performance variation across different metrics and models. These findings align with Araci (2019), who illustrated the power of domain-specific fine-tuning of BERT-based models to enhance performance, as seen in FinBERT's application to financial sentiment analysis. The study revealed that fine-tuning only the last two layers of BERT could achieve comparable performance with less training time, highlighting efficient training strategies. Additionally, another study found that leveraging pre-training data in fine-tuning can sometimes lead to better generalization on target tasks (Liu et al., 2021).

4.8 Final Implementation

4.8.1 Choice of Model for Framework

The evaluation of the BART model on the question dataset showed its performance after fine-tuning, decreasing zero similarity instances to only 5 and increasing non-zero similarity instances to 1764. This performance exceeded that of other models like MiniLM, Albert Base, and Bert, highlighting BART's effectiveness in tasks requiring deep contextual understanding and text generation, essential for the question-answering system.

4.8.2 Implementing a Sentence Transformer Model

Although large language models provide the foundation for accurate QA systems, their complexity and the long response time create obstacles in their direct application for real-time online applications. Therefore, the following adjustment to the workflow was proposed. The process began by saving the embeddings of all descriptions using the 'all-MiniLM-L6-v2' sentence transformer from Hugging Face, which is a one-time task. Subsequently, the system employs BERT embeddings to transform text into 384-dimensional vectors, enhancing the efficiency of semantic search tasks. Cosine similarity calculations were then performed to match questions with these pre-saved embedded descriptions. The model leveraged these calculations to yield the top 10 descriptions that most closely aligned with the input question. These were then provided to the QA model for answer extraction.

Table 4.3. Accuracy distribution of the sentence transformer model across different prediction ranks

Accuracy Type	Top-3 Accuracy (%)	Top-5 Accuracy (%)	Top-10 Accuracy (%)
Accuracy Value	83.55	87.62	91.12

Prediction Rank	Frequency (Number of Times Answer appears)
1st Prediction	1081
2nd Prediction	266
3rd Prediction	131
4th Prediction	41
5th Prediction	31
6th Prediction	17
7th Prediction	18
8th Prediction	15
9th Prediction	9
10th Prediction	3

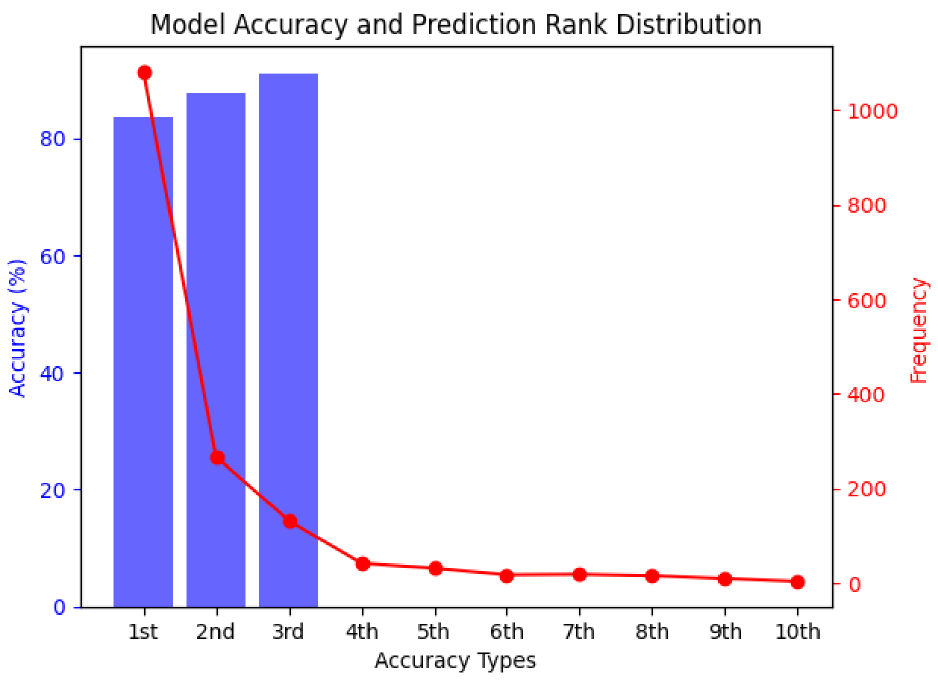


Figure 4.6: Correlation between model prediction accuracy and rank

Table 4.3 presents a detailed view of the accuracy distribution across various prediction ranks. For Top-3 accuracy, at 83.55%, the data shows that the correct answer appears as the first prediction 1081 times out of 1769 instances, demonstrating a strong predictive capability. If not the first, the correct answer ranks second 266 times and third 131 times, suggesting a high likelihood of accuracy in the top predictions, which is also obvious in Figure 4.6.

To display the top three and top five responses, scoring mechanisms from both the sentence transformer model and the QA model are combined. This integration is essential because the QA model alone does not provide accurate scoring. By leveraging the strengths of both models, the most relevant answers are better identified based on their compound scores.

Expanding to Top-5 accuracy, slightly higher at 87.62%, the trend continues with most correct answers still in the first prediction. The figures for the second and third predictions are consistent with Top-3 results, with additional occurrences at the fourth (41 times) and fifth (31 times) predictions. This shows a modest increase in accuracy from Top-3 to Top-5, indicating diminishing returns at lower ranks.

The analysis extends to Top-10 accuracy, which stands at 91.12%, confirming that the transformer sentence model typically achieves correct predictions 91% of the time and frequently ranks them in the top 10.

4.9 Error Analysis

Upon thorough analysis, several key issues were identified that influenced the accuracy of the model's predictions.

A key challenge identified was the generality of questions, leading to accurate answers missing from the top 10 predictions, especially in supplier transactions and invoice processing. Similar keywords in these contexts caused the embedding model to misjudge relevance. In addition, differences between user instructions and provided descriptions, coupled with a lack of critical phrases for context matching, reduced the accuracy of the model's responses. The model also faced difficulties with vague questions and non-specific contexts, resulting in mismatched or irrelevant answers. To overcome these issues, refining descriptions to be more narrative and detailed, and avoiding bullet points, is advised. Ensuring alignment between the keywords in questions and descriptions will further improve response accuracy.

4.9.1 QA Model Analysis

The analysis of model performance revealed that Bart, Bert, and Electra Small models underperformed due to long, bullet-pointed descriptions. A shift to shorter, narrative-style descriptions is suggested for improvement. Additionally, the inability of models to rank the correct answer first suggests the need for better-aligned descriptions and question formulations. Implementing a validation scoring system could prompt respondents to verify the validity of questions, with invalid ones being rephrased for enhanced clarity and response accuracy.

4.9.2 Enhancing Question Validity with Meta-data Utilization

This study improved question validity assessment by using the dataset's meta-data that details the area and specific tasks of descriptions. By evaluating questions using transformer models and calculating BERT embeddings of meta-data, the framework's efficiency in finding answers was significantly increased by setting a threshold of 0.85 out of 1. The analysis aimed to refine question formulation for the model, resulting in an initial count of 1818 valid and 722 invalid questions. After setting a validity threshold of 0.85, the dataset was reduced to 1179 entries with 98.13% considered valid. Adjusting the threshold to 87% differentiated between 1037 valid and 142 invalid questions, while a lower threshold of 80% classified all questions as valid. Further reduction to 0.5 maintained the 100% validity rate.

Table 4.4. Comprehensive table summarizing the findings related to question validity

Threshold Setting	Description	Number of Valid Questions	Number of Invalid Questions	Percentage of Valid Questions	Percentage of Invalid Questions	Additional Notes
Before Data Cleaning	Initial Analysis	1818	722	-	-	-
After Data Cleaning (0.85 Threshold)	Post-cleaning Analysis	1157	22	98.13%	1.87%	Rows with NaN removed: 1361; Rows remaining: 1179
Threshold 0.87	Adjusted Threshold Analysis	1037	142	87.96%	12.04%	-
Threshold 0.80	Lower Threshold Analysis	1179	0	100%	0%	All questions considered valid
Threshold 0.50	Further Reduced Threshold	1179	0	100%	0%	All questions considered valid

The subsequent analysis included labeling the dataset as valid and invalid questions and assessing their performance. For valid questions, the likelihood of identifying the top answer was 73%, an improvement from the previously combined valid and invalid result of 61%. Specifically, there were 863 true matches (indicating high accuracy) and 316 false matches (highlighting areas for improvement). Additionally, in the top 3 answers, the true rate stood at approximately 84.90%, with the false rate at about 15.10%. Invalid questions often lacked specificity, exemplified by vague queries such as "How is the test done?" In contrast, successful questions used a structured format like "What are the steps for [specific process] in [Test Script Name]?", leading to clearer and more targeted inquiries. Common failed questions included generic language and missing keywords, making them too broad for precise test script alignment, as seen in "Can you give a description of the steps to pay a vendor?"

For enhanced question accuracy, it is recommended to:

- Employ clear and direct queries using standard formats starting with "How," "What," or similar prompts.
- Include enough detail to reduce ambiguity and closely mimic the script language.

Providing sample templates based on effective questions can guide users to improve their inquiries. By following these practices and utilizing structured templates, users can greatly improve the model’s accuracy in identifying relevant answers. The overall flow of the framework is illustrated in Figure 4.7.

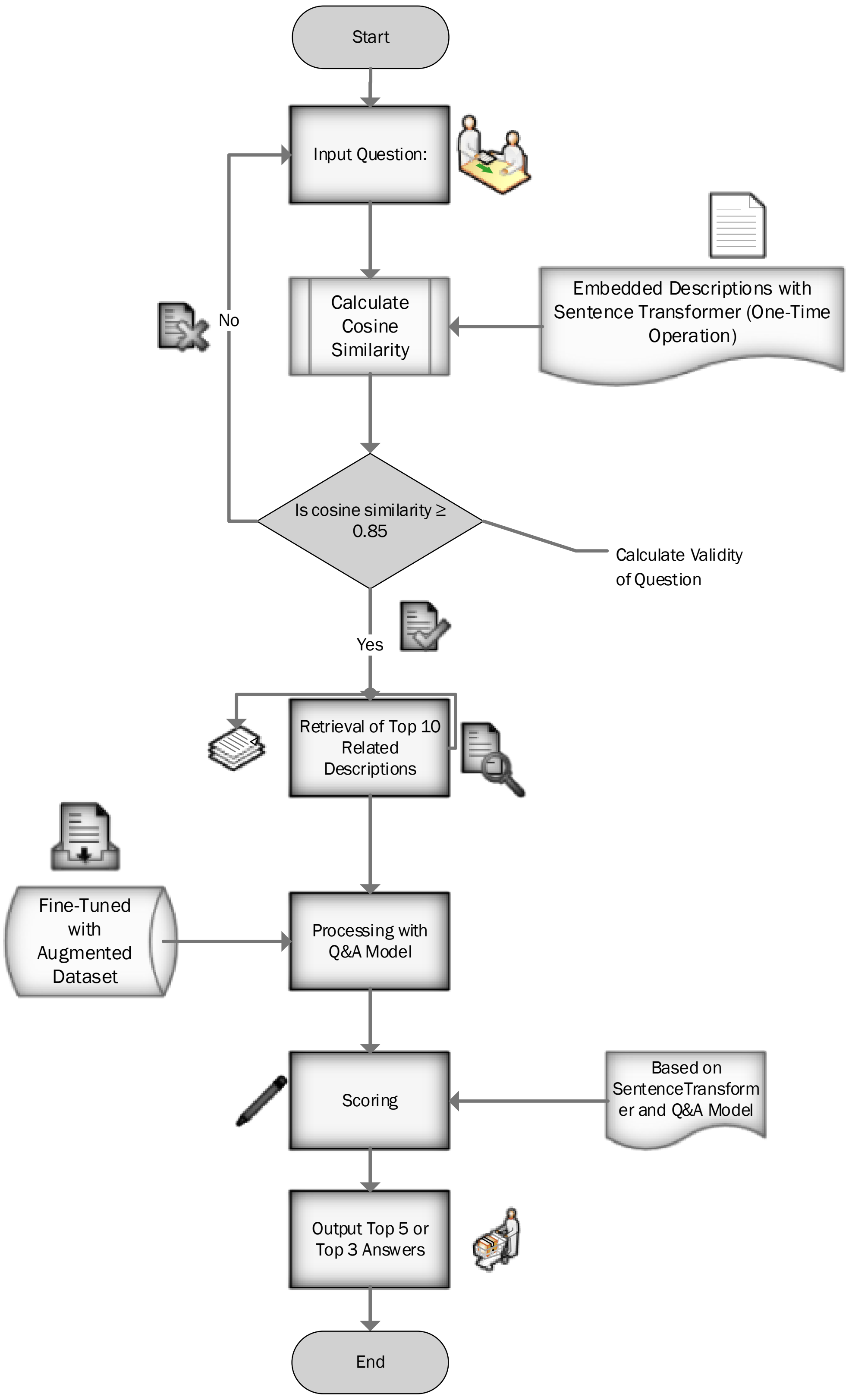

Start
Input Question:
Calculate Cosine Similarity
Embedded Descriptions with Sentence Transformer (One-Time Operation)
No
Is cosine similarity ≥ 0.85
Calculate Validity of Question
Yes
Retrieval of Top 10 Related Descriptions
Fine-Tuned with Augmented Dataset
Processing with Q&A Model
Scoring
Based on SentenceTransformer and Q&A Model
Output Top 5 or Top 3 Answers
End

Figure 4.7. Flowchart of the QA System Testing Process

5 Conclusion and Future Work

This study highlights the role of fine-tuning in enhancing the performance of QA systems within low-resource domains. By incorporating synthetic data into the fine-tuning process, the models demonstrated significant improvements, evidenced by a reduction in zero similarity instances and better alignment of question-answer pairs. These enhancements underscore the effectiveness of fine-tuning in improving the accuracy and relevance of QA systems, even with limited data availability. Transformer models optimized with domain-specific data align answers more accurately and reduce irrelevant responses, highlighting the critical need for targeted fine-tuning as demonstrated by evaluation metrics.

The chosen data augmentation methods proved beneficial as they maintained the integrity of the constructed data. Evidently, when the augmented question dataset was fed into the investigated models, the majority of the fine-tuned models performed better than their original variants. Fine-tuned models excel in closed domains as they are specifically adjusted to align with unique domain requirements, thereby outperforming generic pre-trained models by delivering more precise answers.

Error analysis provided insightful disclosures regarding the formulation of descriptions. A narrative style is more comprehensible to the language-based QA system than bullet points. For optimized question-answering interactions and precise information retrieval, question formulation is key. Successful questions are typically detailed, incorporate key processes or features, and align with the target description. This specificity increases cosine similarity and accuracy in finding matches. Conversely, questions below the 0.85 similarity threshold often lack specificity, are ambiguous, or use generalized language, leading to lower scores.

Meta-data improve question processing by enhancing the accuracy of information retrieval and QA interactions. By comparing the cosine similarity between meta-data (details the area and specific tasks of descriptions) with the questions, it helps validate questions and avoid vague queries that could lead to incorrect answers. Using consistent, meta-data-aligned terminology increases cosine similarity scores, boosting query success. Users are advised to use specific keywords that align with meta-data, formulate clear questions, and maintain terminology consistency.

In low-resource domains, information retrieval involves using a sentence transformer model to overcome the challenge of providing context to the QA model for effective answer extraction. These strategies collectively improve the system's ability to accurately identify and retrieve the most relevant answers, thus enhancing the effectiveness and efficiency of the QA system.

In conclusion, while large transformer models like GPT-3 offer substantial advantages in terms of performance, their requirements for large datasets and extensive computational resources limit their applicability in many practical scenarios. This study therefore employs smaller, more manageable models that, despite their relatively lower performance, are significantly better suited for environments requiring quick responses and where resources are limited. Despite the potential drawbacks of using smaller models, the chosen approach offers significant benefits for specific applications. This could include greater accessibility, lower cost, and the ability to operate effectively in resource-constrained environments.

One of the challenges in developing the QA framework was encountered in the faulty scoring mechanisms that often incorrectly rank answers by textual pattern matching rather than factual accuracy. Future research should focus on developing algorithms that understand content validity and context, with the potential aid of NLP and ML advancements. Incorporating diverse metrics and user feedback could lead to more accurate response evaluations. Therefore, there is a need for advanced scoring systems to evaluate the correctness of an answer and assess its relevance to the query, thus ensuring that the QA system's outcomes are both accurate and contextually appropriate.

References

Allam, A. M. N., & Haggag, M. H. (2012). The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS), 2*(3).

Araci, D. (2019). FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. https://doi.org/10.48550/*arXiv*.1908.10063

Assem, H., Sarkar, R., & Dutta, S. (2021). Qasar: Self-supervised learning framework for Extractive Question Answering. *2021 IEEE International Conference on Big Data (Big Data).* https://doi.org/10.1109/bigdata52589.2021.9671570.

Barmawi, A., & Muhammad, A. (2019). Paraphrasing Method Based on Contextual Synonym Substitution. *Journal of ICT Research and Applications, 13,* 257-282. https://doi.org/10.5614/itbj.ict.res.appl.2019.13.3.6.

Calijorne Soares, M. A., & Parreiras, F. S. (2020). A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University - Computer and Information Sciences, 32*(6), 635–646. https://doi.org/10.1016/j.jksuci.2018.08.005

Cao, N., Aziz, W., & Titov, I. (2018). Question Answering by Reasoning Across Documents with Graph Convolutional Networks. *Association for Computational Linguistics: Stroudsburg, PA, USA.*

Chen, Y. H., Lu, E. L., & Ou, T. A. (2021). Intelligent SPARQL Query Generation for Natural Language Processing Systems. *IEEE Access, 9,* 158638-158650. doi: 10.1109/ACCESS.2021.3106957

Bach, N., Thanh, P., & Oanh, T. (2020). Question Analysis towards a Vietnamese Question Answering System in the Education Domain. *Cybernetics and Information Technologies, 20*, 112-128. doi: 10.2478/cait-2020-0011

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. https://arxiv.org/abs/2003.10555

Derici, C., Çelik, K., Kutbay, E., Aydın, Y., Güngör, T., Özgür, A., & Kartal, G. (2015). Question analysis for a closed domain question answering system. *Computational Linguistics and Intelligent Text Processing*, 468–482. https://doi.org/10.1007/978-3-319-18117-2_35

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North.* https://doi.org/10.18653/v1/n19-1423

Devlin, J., et al. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
arXiv preprint arXiv:1810.04805.

Diefenbach, D., Lopez, V., Singh, K., & Maret, P. (2018). Core Techniques of Question Answering Systems over Knowledge Bases: A Survey. *Knowledge and Information Systems*, 55, 529-569. doi: 10.1007/s10115-017-1109-8

Esan, A., Adanigbo, O., Okomba, N., Koledoye, O., & Omodunbi, B. (2021). Development of a Closed Domain Question Answering System based on Deep Learning. *LAUTECH Journal of Computing and Informatics (LAUJCI), 2*(1), 15-24. ISSN: 2714-4194. Retrieved from https://laujci.lautech.edu.ng/index.php/laujci/article/view/34

Gan, W., & Ng, H. (2019). Improving the Robustness of Question Answering Systems to Question Paraphrasing. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 6065-6075. https://doi.org/10.18653/v1/P19-1610.

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification*. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers). https://doi.org/10.18653/v1/p18-1031

Hu, N., Mitchell, E., Manning, C., & Finn, C. (2023). Meta-Learning Online Adaptation of Language Models. *ArXiv*, abs/2305.15076. https://doi.org/10.48550/arXiv.2305.15076.

Huang, J., Gu, S., Hou, L., Wu, Y., Wang, X., Yu, H., & Han, J. (2022). Large Language Models Can Self-Improve. *arXiv*, abs/2210.11610. https://doi.org/10.48550/arXiv.2210.11610.

Jboback. (2023, December 19). *What is question answering? - azure AI services.* Azure AI services | Microsoft Learn. https://learn.microsoft.com/en-us/azure/ai-services/language-service/question-answering/overview

Joshi, M. (2020). *Oracle fusion cloud financials using payables invoice to pay 24a* (F88639-02). Oracle and/or its affiliates.

Jurafsky, D., & Martin, J. H. (2014). *Speech and Language Processing.* (2nd ed., Vol. 1) Prentice Hall.

Kia, M. A., Garifullina, A., Kern, M., Chamberlain, J., & Jameel, S. (2022). Adaptable closed-domain question answering using contextualized CNN-attention models and question expansion. *IEEE Access, 10*, 45080–45092. https://doi.org/10.1109/access.2022.3170466

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv*, abs/1909.11942. Retrieved from http://arxiv.org/abs/1909.11942

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)* (pp. 7871–7880). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703

Liello, L., Garg, S., Soldaini, L., & Moschitti, A. (2022). *Pre-training Transformer Models with Sentence-Level Objectives for Answer Sentence Selection*. ArXiv, abs/2205.10455. https://doi.org/10.48550/arXiv.2205.10455.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach.* arXiv:1907.11692 [cs.CL]. https://doi.org/10.48550/arXiv.1907.11692

Liu, Z., Xu, Y., Xu, Y., Qian, Q., Li, H., Chan, A., & Jin, R. (2021). *Improved Fine-tuning by Leveraging Pre-training Data: Theory and Practice. ArXiv*, abs/2111.12292.

Maheshwari, G., Trivedi, P., Lukovnikov, D., Chakraborty, N., Fischer, A., & Lehmann, J. (2019). Learning to Rank Query Graphs for Complex Question Answering over Knowledge Graphs. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11778, 487-504. https://doi.org/10.1007/978-3-030-30796-7_32

Malik, N., Sharan, A., & Biswas, P. (2013). Domain knowledge enriched framework for restricted domain question answering system. *2013 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC).* https://doi.org/10.1109/iccic.2013.6724163

Matsubara, Y., Vu, T., & Moschitti, A. (2020). Reranking for efficient transformer-based answer selection. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* https://doi.org/10.1145/3397271.3401266

Nassiri, K., & Akhloufi, M. (2022). Transformer models used for text-based question answering systems. Applied Intelligence. https://doi.org/10.1007/s10489-022-04052-8

Oh, K., Choi, H., Gweon, G., Heo, J., & Ryu, P. (2015). Paraphrase generation based on lexical knowledge and features for a natural language question answering system. *2015 International Conference on Big Data and Smart Computing (BIGCOMP)*, 35-38. https://doi.org/10.1109/35021BIGCOMP.2015.7072846

Pappas, D., Malakasiotis, P., & Androutsopoulos, I. (2022*).* Data Augmentation for Biomedical Factoid Question Answering. *ArXiv*, abs/2204.04711. https://doi.org/10.48550/arXiv.2204.04711

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors for Word Representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Pergola, G., Kochkina, E., Gui, L., Liakata, M., & He, Y. (2021). Boosting Low-Resource Biomedical QA via Entity-Aware Masking Strategies. *ArXiv*, abs/2102.08366. https://doi.org/10.18653/v1/2021.eacl-main.169

Phade, A., & Haribhakta, Y. (2021). Question answering system for low resource language using transfer learning. *2021 International Conference on Computational Intelligence and Computing Applications (ICCICA)*. https://doi.org/10.1109/iccica52458.2021.9697268

Purwarianti, A. (2019). Effective Use of Augmentation Degree and Language Model for Synonym-based Text Augmentation on Indonesian Text Classification. *2019 International Conference on Advanced Computer Science and information Systems (ICACSIS),* 217-222. https://doi.org/10.1109/ICACSIS47736.2019.8979733

Question Answering - Hugging Face. (n.d.). *Hugging Face NLP Course*. Retrieved from https://huggingface.co/learn/nlp-course/chapter7/7?fw=pt#post-processing

Radiya-Dixit, E., & Wang, X. (2020*). How fine can fine-tuning be? Learning efficient language models*. *ArXiv*, abs/2004.14129

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research, 21*(140), 1–67.

Rawat, A., & Singh Samant, S. (2022). Comparative analysis of transformer-based models for question answering. *2022 2nd International Conference on Innovative Sustainable Computational Technologies (CISCT)*. https://doi.org/10.1109/cisct55310.2022.10046525

Reddy, R. (2023). Universal Language Model Fine-Tuning for Text Classification. *International Journal for Research in Applied Science and Engineering Technology.* https://doi.org/10.22214/ijraset.2023.53887

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using Siamese Bert-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).* https://doi.org/10.18653/v1/d19-1410

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* preprint arXiv:1910.01108.

Schmidt, M., Bartezzaghi, A., Bogojeska, J., Malossi, A., & Vu, T. (2022). Improving Low-Resource Question Answering using Active Learning in Multiple Stages. *ArXiv*, abs/2211.14880. https://doi.org/10.48550/arXiv.2211.14880

Shakeel, M., Karim, A., & Khan, I. (2019). A Multi-cascaded Model with Data Augmentation for Enhanced Paraphrase Detection in Short Texts. *ArXiv*, abs/1912.12068. https://doi.org/10.1016/j.ipm.2020.102204

Shao, T., Guo, Y., Chen, H., & Hao, Z. (2019). Transformer-Based Neural Network for Answer Selection in Question Answering. IEEE Access, 7, 26146-26156. https://doi.org/10.1109/ACCESS.2019.2900753.

Shorten, C., Khoshgoftaar, T., & Furht, B. (2021). Text Data Augmentation for Deep Learning. *Journal of Big Data, 8*. https://doi.org/10.1186/s40537-021-00492-0

Song, L., Wang, Z., Yu, M., Zhang, Y., Florian, R., & Gildea, D. (2018). Exploring Graph-structured Passage Representation for Multi-hop Reading Comprehension with Graph Neural Networks. *ArXiv* preprint arXiv:1809.02040.

Soni, S., & Roberts, K. (2020). Paraphrasing to improve the performance of Electronic Health Records Question Answering. AMIA Joint Summits on Translational Science proceedings. *AMIA Joint Summits on Translational Science, 2020*, 626-635.

Soricut, R., & Brill, E. (2006). Automatic question answering using the web: Beyond the Factoid. *Information Retrieval, 9*, 191-206. https://doi.org/10.1007/s10791-006-7149-y.

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to Fine-Tune BERT for Text Classification?, 194-206. https://doi.org/10.1007/978-3-030-32381-3_16

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., & Zhou, D. (2020). Mobilebert: A compact task-agnostic Bert for Resource-limited devices. *Proceedings of the 58th Annual Meeting of the Association for*

Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.195

Thakker, T. (2015). Introduction to Oracle Fusion Applications, 3-22. https://doi.org/10.1007/978-1-4842-0983-7_1

Tran, P., Nguyen, D., Tran, H.-A., Nguyen, T., & Tran, T. (2023). Building a closed-domain question answering system for a low-resource language. *ACM Transactions on Asian and Low-Resource Language Information Processing, 22*(3), 1–14. https://doi.org/10.1145/3566123

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *In Advances in Neural Information Processing Systems* (30)

Wang, B., Kuo, C., & Li, H. (2022). Just Rank: Rethinking Evaluation with Word and Sentence Similarities. *ArXiv*, abs/2203.02679. https://doi.org/10.48550/arXiv.2203.02679

Wang, Q., Yin, F., & Liu, C. (2012). Handwritten Chinese Text Recognition by Integrating Multiple Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34*, 1469-1481. https://doi.org/10.1109/TPAMI.2011.264

Wang, R. Z., Ling, Z. H., & Hu, Y. (2019). Knowledge Base Question Answering with Attentive Pooling for Question Representation. *IEEE Access, 7*, 46773-46784. https://doi.org/10.1109/ACCESS.2019.2906345

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., & Zhou, M. (2020). MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. *ArXiv* preprint arXiv:2002.10957. https://doi.org/10.48550/arXiv.2002.10957

Wang, Z., Ng, P., Ma, X., Nallapati, R., & Xiang, B. (2019). Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering. *In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).*

Wanjawa, B., & Muchemi, L. (2020). Using semantic networks for question answering - case of low-resource languages such as Swahili. *Advances in Intelligent Systems and Computing,* 278–285. https://doi.org/10.1007/978-3-030-51328-3_39

Wei, J., & Zou, K. (2019). EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks., 6381-6387. https://doi.org/10.18653/v1/D19-1670

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., & Brew, J. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*, abs/1910.03771

Wu, T., Yu, H., Wan, F., & Yang, F. (2019). Research on answer extraction for automatic question answering system. *Proceedings of the 3rd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019).* https://doi.org/10.2991/iccia-19.2019.34

Yang, Y., Qiao, Y., Shao, J., Anand, M., Yan, X., & Yang, T. (2021). Composite Re-Ranking for Efficient Document Search with BERT. *ArXiv*, abs/2103.06499

Zafar, H., Napolitano, G., & Lehmann, J. (2018). Formal Query Generation for Question Answering over Knowledge Bases. Lecture Notes in Computer Science (Including Subseries Lecture Notes in *Artificial Intelligence and Lecture Notes in Bioinformatics), 10843*, 714-728. https://doi.org/10.1007/978-3-319-91947-8_44

Zettlemoyer, L., & Collins, M. (2005). Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. *ArXiv*, abs/1207.1420.

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020). PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. *Proceedings of the 37th International Conference on Machine Learning, in Proceedings of Machine Learning Research, 119, 11328-11339.* https://proceedings.mlr.press/v119/zhang20ae.html

Zheng, H. T., Fu, Z. Y., Chen, J. Y., Sangaiah, A., Jiang, Y., & Zhao, C. Z. (2018). Novel knowledge-based system with relation detection and textual evidence for question answering research. *PLoS ONE,* 13. https://doi.org/10.1371/journal.pone.0205097