VIETNAM NATIONAL UNIVERSITY
HANOI UNIVERSITY OF SCIENCE

Computer and Information Science (Honors Program)

Natural Language Processing and Application

Question Answering Model and an optimized version of it for Vietnamese Language

Students:
Tran Hoang Anh - 20001528
Ngo Phuong Anh - 20001523
Nguyen Thi Phuong Hoa - 20001549

Hanoi, December 16th, 2023

Contents

List of Figures

LIST OF ABBREVIATIONS

- **NLP:** Natural Language Processing
- **QA:** Question Answering
- **IR:** Information Retrieval
- **MRC:**Machine Reading Comprehension
- **LLM:** Large Language Model
- **BERT:** Bidirectional Encoder Representation from Transformers
- **GPT:** Generative Pretrained Transformer
- **QLoRA:** Quantization-aware Low-rank Approximation
- **LoRA:** Low-Rank Adaption

ABSTRACT

In the ever-expanding realm of information, Question Answering (QA) and Information Retrieval (IR) are crucial in helping users access and understand relevant knowledge. The research methodology involves a comprehensive review of existing QA models, identifying their strengths and limitations.

Key findings include the development of a hybrid model that seamlessly integrates state-of-the-art deep learning approaches. This integration not only boosts the precision of answer extraction but also accelerates the retrieval process by optimizing the relevance ranking of documents. The experimental results demonstrate a significant improvement in both accuracy and speed compared to baseline models. Moreover, the report also investigates the impact of diverse datasets on QA performance, highlighting the adaptability of the proposed framework across different domains. We mainly employ BERT to analyze **UIT-ViQuAD** - A Vietnamese Dataset for Evaluating Machine Reading Comprehension and **ViMMRC** - A Vietnamese Multiple-choice Machine Reading Comprehension Corpus.

In summary, this study makes a significant contribution to the evolving discourse on advancing Question-Answering (QA) techniques, presenting a robust framework utilizing deep learning. Our research highlights the considerable potential for enhancing information extraction tasks, providing valuable insights that contribute to the broader field of Natural Language Processing (NLP). The implications of our findings extend to various applications, including but not limited to Web search, Customer Service and Support, Healthcare and Medical Research, Financial analysis, and investment research. Importantly, this study lays the groundwork for the development of an effective Vietnamese QA model.

Chapter 1

INTRODUCTION

1.1 Problem Definition

Amidst the era of explosive information growth and the rapid evolution of generative AI models, a profound understanding of Question-Answering (QA) and Information Retrieval (IR) systems has become more crucial than ever. These systems act as knowledge gatekeepers, essential filters enabling us to navigate the vast ocean of data and uncover the answers we seek. Generative AI models, particularly those based on transformer architectures, are increasingly integrated into chatbot development. Grasping the intricacies of QA and IR systems is paramount for enhancing the capabilities of these chatbots, empowering them to deliver more accurate and contextually relevant responses to user queries.

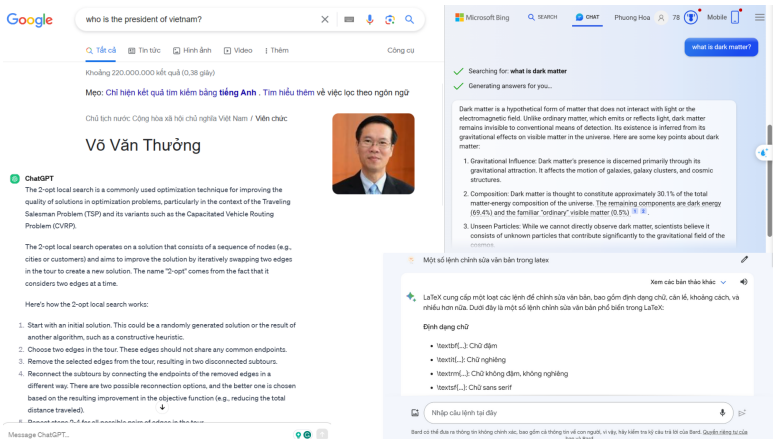

Figure 1.1: Some popular chatbots recently

Among the myriad of QA models available today, our research team has chosen the XLM-RoBERTa model. Built on the transformer architecture, XLM-RoBERTa exhibits outstanding multilingual processing capabilities, aligning seamlessly with our goal of constructing a Vietnamese reading comprehension model.

Additionally, the pre-trained nature of the XLM-RoBERTa model allows us to harness pre-existing hyperparameters, optimizing the efficiency of the training process. Furthermore, our team proposes a novel approach to enhance training efficiency by implementing QLoRA. This innovative solution addresses challenges related to training time and memory storage, particularly significant for the Vietnamese language, where tokenization incurs higher resource demands compared to other languages. Although the combination of Vietnamese and XLM is relatively unexplored, our experimental results showcase promising outcomes, underscoring the potential of this approach.

In the next part, we would like to present Outline of our research.

1.2 Outline

The rest of the paper is organized as follows:

- **Chapter 1** of the report includes: Defining the problem and clarifying the structure of the report.
- In the **Background section (Chapter 2)**, we will give the base knowledge we use in this report. This will give reader have a brief point of view before dive more into our technical part.
- In **Chapter 3 - Methodology**. we present details on the dataset used for training, introduce our base model XLM-RoBERTa, and outline our novel approach for optimization during fine-tuning, incorporating QLoRA into the training process.
- **After that, in Chapter 3** - we will present the outcomes of our fine-tuning process and compare the effectiveness between two models: one with the application of QLoRA and one without.
- At the end of the paper, we will summarize our research and provide future direction.

Chapter 2

Background

In this chapter, we provides a comprehensive overview of the foundational elements crucial to our study. It delves into the nature of the Question-Answering (QA) task, elucidating the challenges and objectives inherent in this domain. Additionally, the architecture of transformers, a pivotal paradigm shift in natural language processing, is expounded upon. Of particular significance is the attention mechanism, a key innovation that sets transformer models apart from traditional approaches. The attention mechanism plays a pivotal role in capturing contextual relationships, enabling transformers to excel in understanding and processing complex language structures. This section serves as a crucial foundation for understanding the subsequent discussions on our chosen model, XLM-RoBERTa, and the novel integration of QLoRA in the fine-tuning process.

2.1 Question Answering

Question answering systems are designed to fill human information needs that might arise in situations like talking to a virtual assistant, interacting with a search engine, or querying a database.

Question Answering (QA) is a fast-growing research problem in computer science that aims to find short concrete answers. There are two major approaches for QA systems: text-based QA, and knowledge-based QA. Knowledge-based QA is rely on knowledge bases(KB) for finding the answer to the user's question. [1]

Despite their effectiveness in certain contexts, classic QA models faced challenges with scalability, adaptability to different domains, and handling complex language nuances. The advent of deep learning and the rise of transformer-based models have since revolutionized QA, achieving remarkable performance improvements and overcoming many limitations associated with classic models.

2.1.1 Classic model of QA task

While neural architectures are the state of the art for question answering, pre-neural architectures using hybrids of rules and feature-based classifiers can some-

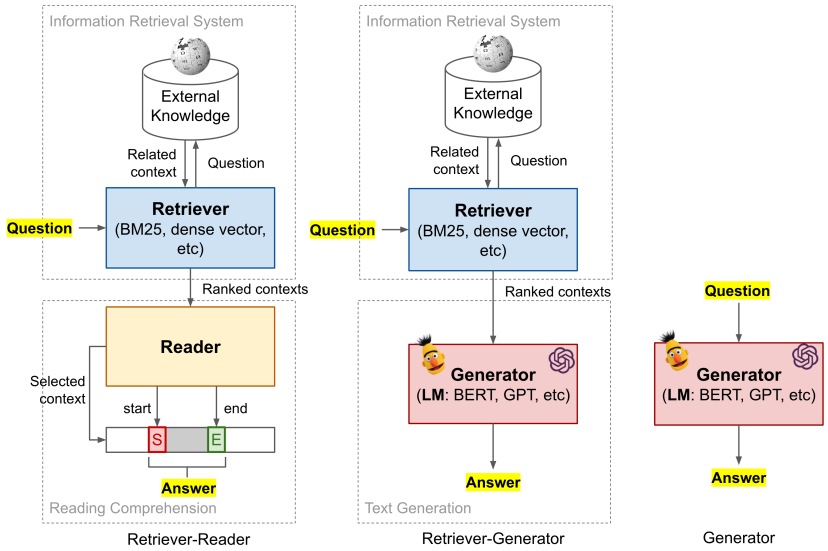

Figure 2.1: Overview QA

times achieve higher performance.

In the statistical approach, classic QA models utilized machine learning algorithms to rank candidate answers based on features extracted from the question and context. These models often incorporated techniques like term frequency-inverse document frequency (TF-IDF) for document ranking and feature engineering for answer extraction.

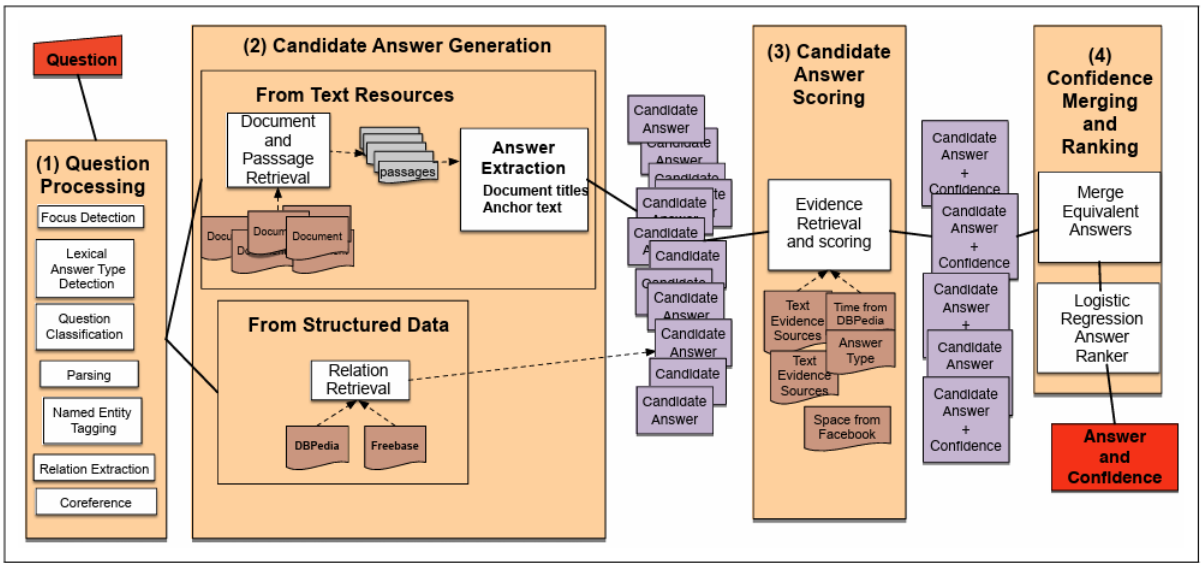

Figure 2.2: The 4 broad stages of classic QA: (1) Question Processing, (2) Candidate Answer Generation, (3) Candidate Answer Scoring, and (4) Answer Merging and Confidence Scoring.

2.2 Using Language Models to do QA

Using language models for Question Answering (QA) has revolutionized the field of natural language processing. Pre-trained models like BERT and GPT, based on advanced transformer architectures, serve as powerful tools for understanding and generating human-like text. These models are initially trained on vast amounts of text data, allowing them to capture complex contextual relationships.

In the context of QA tasks, language models are fine-tuned on specific datasets to adapt to the nuances of answering questions. This fine-tuning process enables the models to grasp the intricacies of question-context relationships, making them effective in tasks such as reading comprehension. They excel in extracting relevant information from passages to provide accurate answers.

Language modeling is not yet a complete solution for question answering; for example in addition to not working quite as well, they suffer from poor interpretability (unlike standard QA systems, for example, they currently can't give users more context by telling them what passage the answer came from). Nonetheless, the study of extracting answer from language models is an intriguing area for future question.

2.3 Transformer

Our model, **XLM-RoBERTa**, adopts the transformative transformer architecture—an innovation in deep learning, redefining how neural networks handle sequential data. So in this section, we will inform a transformer architecture as a foundation for XLM-RoBERTa model in Chapter 3 - Methodology.

The transformer architechture stands out as a groundbreaking innovation in the field of deep learning, revolutionizing the way neural networks process sequential data. Introduced by Vaswani et al. in the paper "Attention is All You Need," transformers have become the de facto model architecture for a multitude of natural language processing (NLP) and other sequential tasks. Unlike traditional recurrent or convolutional neural networks, transformers rely on the self-attention mechanism to capture contextual information from input sequences. Notably, transformers have proven effective in machine translation, text summarization, and question-answering, showcasing their versatility and dominance in the landscape of deep learning architectures.

The Transformer adheres to this overarching design by employing stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. These components are illustrated in the left and right sections of Figure 1, respectively.

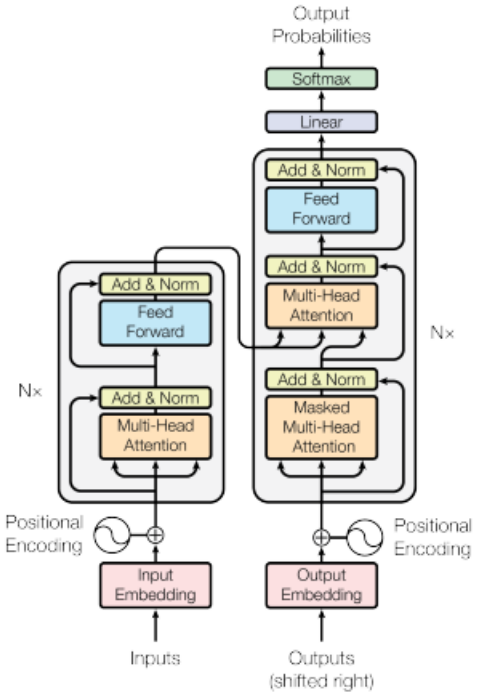

Figure 2.3: The Transformer - model architecture.

2.4 Attention Mechanism

In this section, we present the Attention Mechanism—a transformative discovery that catapulted the transformer into the most superior NLP model compared to others. Highly notable in NLP, attention networks, a pivotal element in deep learning, rose to prominence within the transformer architecture. This mechanism empowers models to dynamically prioritize input elements, fostering superior performance in sequential tasks and overcoming constraints associated with fixed-size context windows. The adaptability afforded by varied attention weights markedly enhances the modeling of long-range dependencies across different architectures. Attention mechanisms wield substantial influence across applications, spanning machine translation to image recognition.

Let's consider a real-life example to better understand the attention mechanism. Imagine when we, as Vietnamese speakers, engage in conversation with someone from another country. When learning a new language, and they speak a complex sentence to us, such as:

- "What nutritious elements are essential for a well-balanced breakfast, and how can one create a wholesome morning meal to kickstart their day with energy and vitality?"

In this scenario, our attention is not evenly distributed across every word in the sentence. Instead, we instinctively focus on the most important words such as ***breakfast*** and ***morning meal*** to discern the question's theme and provide a relevant

answer. And that is how the attention mechanism operates.

The Attention mechanism relies on the intuitive notion that we naturally "attend to" specific parts when handling substantial information.

The necessity for the Attention mechanism arose from the need to provide transformers with extensive long-term memory. This mechanism allows a transformer model to effectively "attend" or "focus" on all previously generated tokens.

While Recurrent Neural Networks (RNN), including Gated Recurrent Units (GRUs) and Long-short Term Memory (LSTM) networks, can consider previous inputs, they have limited reference windows. As the sequence length increases, RNNs struggle to access words generated earlier. In contrast, the attention mechanism, theoretically and with sufficient compute resources, offers an infinite window to reference from, allowing it to utilize the entire context of the story during text generation.

Currently, there are various attention mechanisms such as multi-head, etc. In our research report, we utilize the XLM RoBerta model as the base model (further detailed in Chapter 3 below). Within the XLM RoBerta model, cross-attention is employed alternately with self-attention in each layer. Therefore, we will elucidate the concepts of cross-attention and self-attention in this section.

2.4.1 Self Attention

Self-Attention involves considering the connections among words in a given sentence. To delve deeper into this concept, it's crucial to comprehend three key elements:

- Understanding QUERY, KEY, and VALUE.
- Grasping the concept of Positional Encoding.
- Identifying the content passed to QUERY, KEY, and VALUE.

Let's explore the encoder model introduced in the paper "Attention is All You Need" to address these aspects.

QUERY, KEY, VALUE

In the given image, the three linear layers work with inputs called **"QUERY, KEY & VALUE."** Let's break it down with a simple example.

Imagine you're searching on YouTube or Google. The text you type into the search box represents the QUERY. The results that appear, like video or article titles, are the KEY, and the content inside them is the VALUE. The goal is for the Query to find the best matches by assessing the similarity with the Keys.

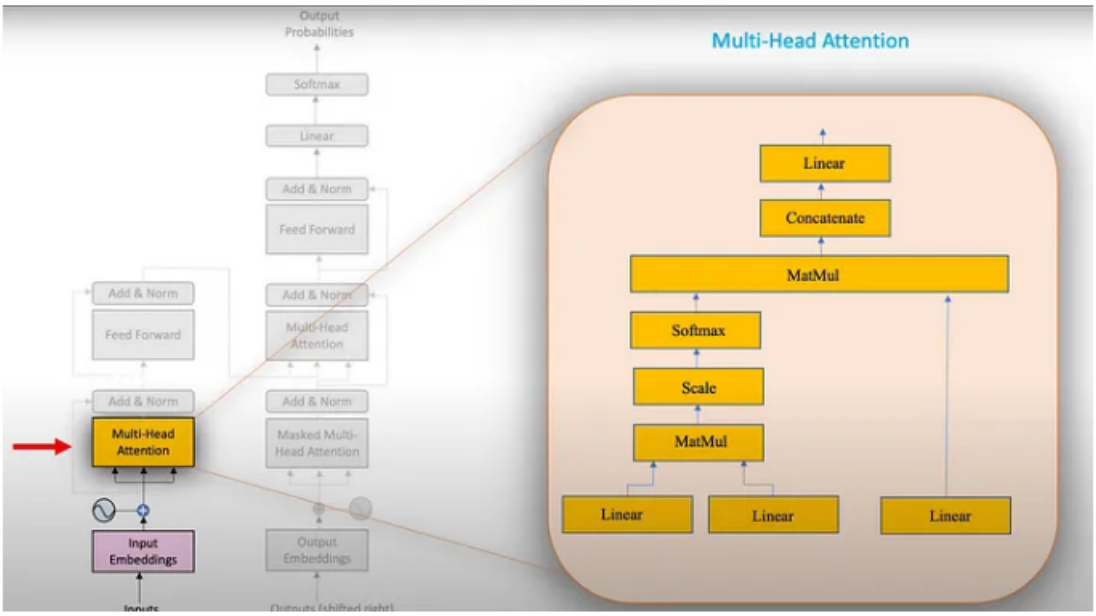

Figure 2.4: An Attention Block

To measure this similarity, we use the Cosine Similarity method. It's like a tool that helps us figure out how similar two things are. Cosine similarity ranges from +1 (very similar) to -1 (very dissimilar).

These steps below are the transformation.

Step 1: First, starting from the basic cosine similarity formula

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

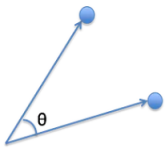

Figure 2.5: Cosine similarity formula

Step 2: As A and B are matrices, not single points, we can transpose matrix B as follows.

$$Similarity(A,B) = \frac{A.B^T}{scaling}$$

Step 3: Since Query and Key are vectors matricies, we can rewrite the formula as below to compute the similarity between Query and Key

$$Similarity(Q, K) = \frac{Q.K^T}{scaling}$$

Imagine you're searching for something on YouTube or Google. The words you type into the search bar are like a question (QUERY). The search engine then looks at all the titles of videos or articles (KEY) and the actual content inside them (VALUE).

Now, to figure out which results are most similar to your question, they use something called Cosine Similarity. It's like a tool that measures how alike two things are. The similarity can range from +1 (meaning very similar) to -1 (meaning very different).

So, when you search, the search engine uses this tool to compare your question with the titles and content, giving you the best-matched results. It's like finding the closest match based on the words you used and what's in the titles and content of the results.

Positional Encoding

Now that we have grasped the concept of Similarity and clarified the roles of QUERY, KEY, and VALUE from the upper section, let's dive into the second question: What exactly is Positional Encoding?

When we deal with text data in machine learning, especially in neural networks, we need to convert words into numbers. The embedding layer plays a crucial role in this process by transforming each word into a fixed-size vector with real values, offering a more nuanced representation than simple 0s and 1s. Think of the embedding layer as a lookup table where words serve as keys, and the resulting dense vectors are the values.

Traditional models like LSTM process one input at a time in a sequence. However, Transformers process all inputs simultaneously, making them faster but potentially losing information about the order of words. To address this, the creators of the "Attention is All You Need" paper introduced Positional Encoding, which uses wave frequencies to capture the positional information of words. This clever idea helps Transformers maintain an understanding of word order while benefiting from their parallel processing capabilities.

The content pass to Query, Key, Value

We provide our position-aware embeddings to the QUERY layer. Subsequently, we duplicate this embedding and supply it to the KEY and VALUE layers as well.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Figure 2.6: Wave frequencie formula

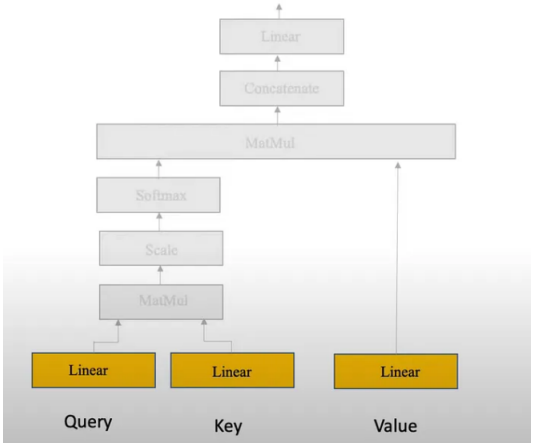

Figure 2.7: The Query, Key and Value in Attention

This might seem counterintuitive—why use the same embedding for all three layers? Here's where SELF-ATTENTION becomes crucial.

Consider the below example:

- **Input:***"Hi, How are you?"*
- **Wanted Output:***"I am fine."*

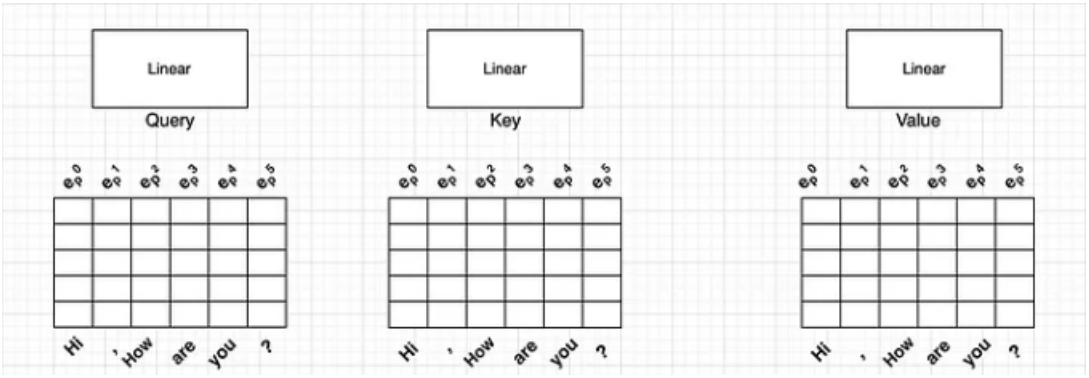

Figure 2.8: An example of Q,K and V

In essence, the input sequence is forwarded through the input embedding layer, followed by the application of position encoding. Subsequently, the resulting positional-aware embeddings are fed into the linear layers.

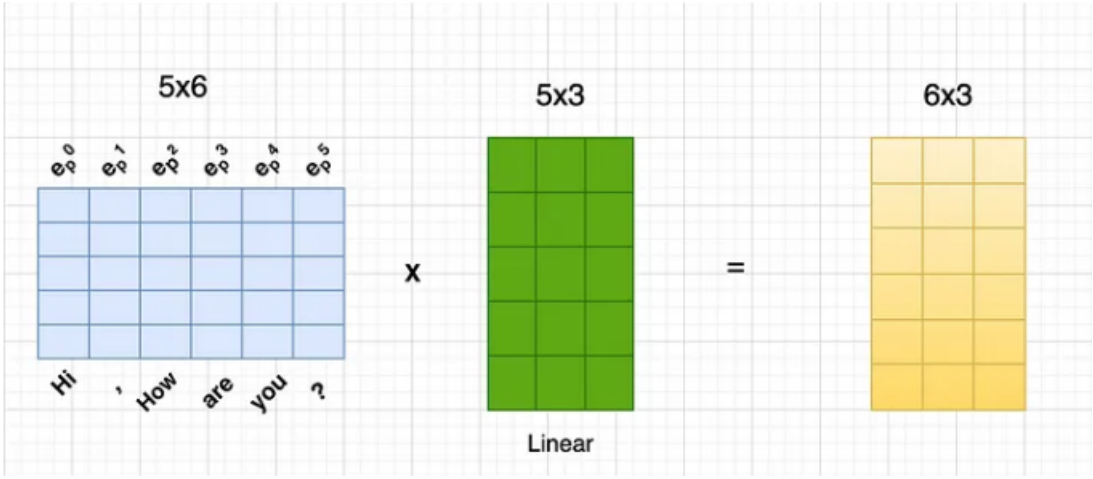

Figure 2.9: The Linear computation

In Figure 2.9, we can observe that the output from the QUERY and KEY linear layers is directed toward the matrix multiplication step in the network. The output of this dot product can be called an ***Attention filter.***

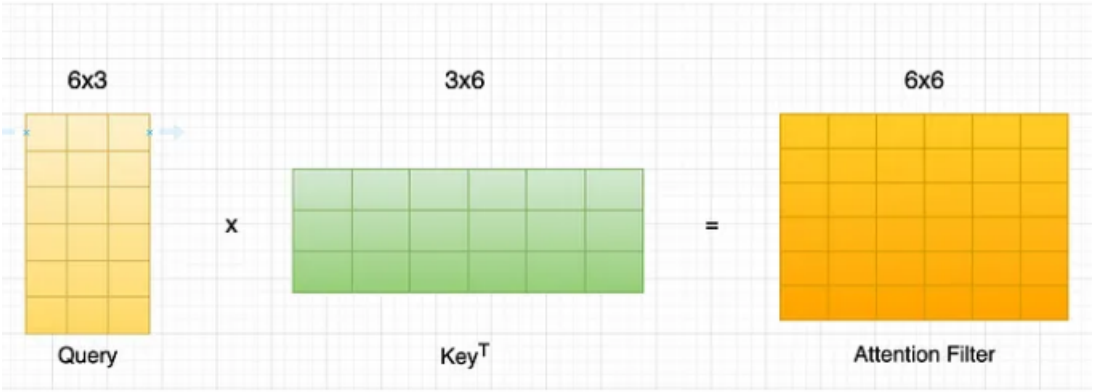

Figure 2.10: Attention filter

After all the computation with matrices, we have the output-attention filter as below:

Initially, the Attention filter weights start as relatively arbitrary numbers. As the training progresses, these weights evolve into more meaningful values, eventually transforming into Attention scores.

Following that, the attention scores undergo scaling, mirroring the process depicted in Figure 2.9. In the "Attention is all you need" paper, the authors normalize the attention score by dividing it by the square root of the dimension of the key

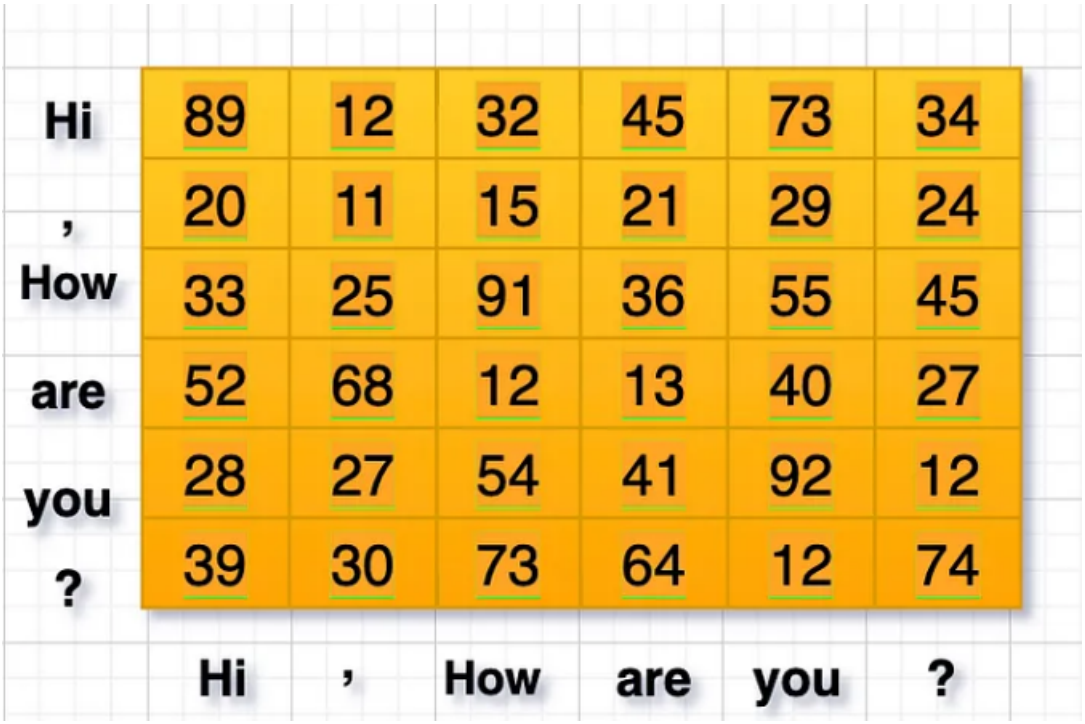

Figure 2.11: Attention Scores

vector, which, in this case, would be divided by 6. Lastly, to confine our attention scores within the range of 0 to 1, the softmax function is applied.

Scaling the attention scores and applying the softmax function serves a couple of key purposes in the context of attention mechanisms:

- Stability: Scaling helps to stabilize the learning process and prevents the model from becoming too sensitive to extreme values. It ensures that the attention scores are within a reasonable range, avoiding issues like vanishing or exploding gradients during training.
- Soft Selection: The softmax function is employed to convert the attention scores into probabilities. These probabilities represent the importance or weight assigned to each element in the sequence. Softmax provides a smooth and differentiable way to turn scores into a probability distribution, making it easier to interpret and propagate gradients during backpropagation.

In summary, scaling and softmax together make the attention mechanism more robust, stable, and amenable to efficient training. They enable the model to softly focus on relevant parts of the input sequence while learning meaningful representations. As mentioned earlier, consider a sentence as our input. When we perform the multiplication of the attention filter with the value matrix, the result is that irrelevant or unnecessary information is effectively filtered out. This step helps the model focus on the most relevant parts of the input sequence while discarding less important elements.

2.4.2 Cross Attention

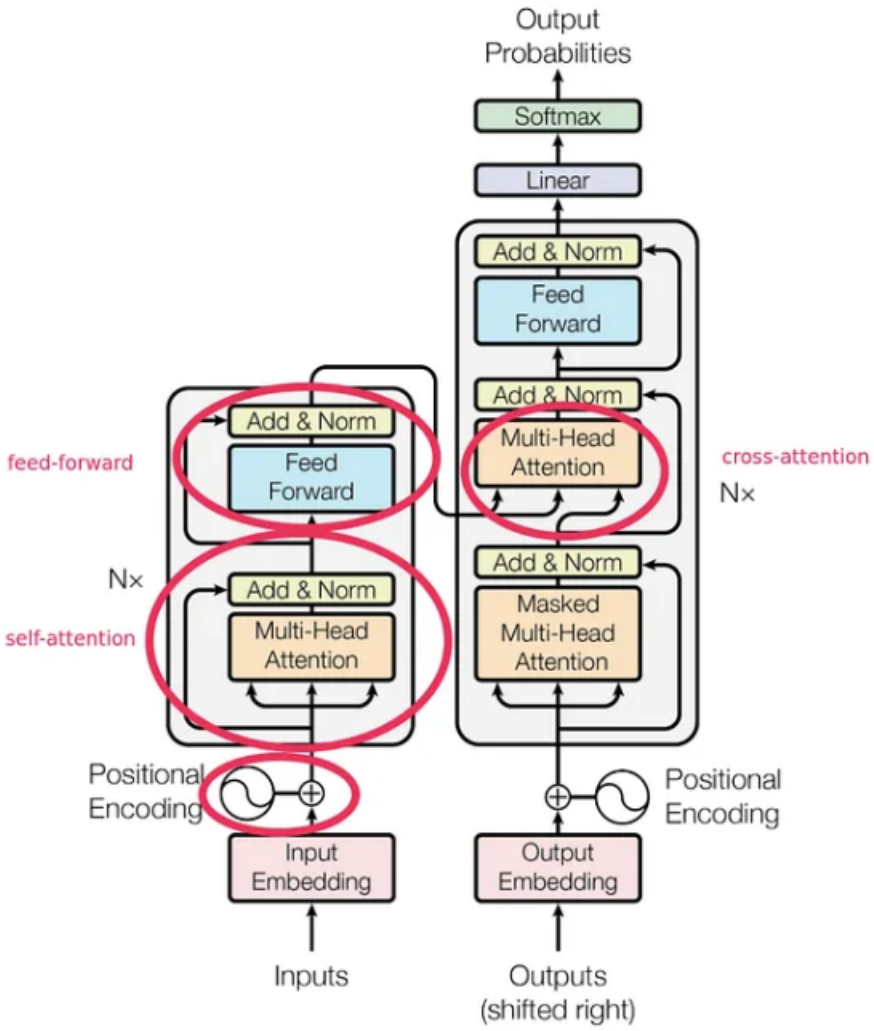

Figure 2.12: Attention in Transformer

To grasp the concept of Cross-Attention, consider one of the most renowned NLP applications: word translation from one language to another, such as English to Vietnamese. For this task, the model needs to acquire the knowledge of how English letters correspond to their representations in Vietnamese.

- **English:** Hello, how are you?
- **Vietnamese:** Xin chào, bạn có khỏe không?

With the transformer model, The English sentence is inputted into the Encoder, and the Vietnamese sentence is provided to the Decoder. For further elucidation, let's focus on the Encoder.

For QA tasks, it's crucial to devise a strategy to determine the quality of answers to the provided questions. In the upcoming section, we will discuss the F1-Score and Exact Match methods used for this purpose.

2.5 Evaluate

In question answering tasks, metrics like F1 Score and exact match are used to assess the model's accuracy and precision in providing answers. F1 Score balances precision and recall, while exact match checks if the predicted answer exactly matches the ground truth answer. These metrics measure the model's effectiveness in comprehending and accurately responding to given questions.

2.5.1 F1-Score

In Question Answering (QA) tasks, the F1 score is a widely used metric to evaluate the model's performance in providing accurate and relevant answers to given questions. The F1 score is particularly valuable when assessing the balance between precision and recall. The F1 score is calculated based on precision (P) and recall (R), which are defined as follows:

$$P = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$R = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

The F1 score is then computed as the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{P \times R}{P + R}$$

In the context of QA tasks:

- True Positives (TP) are the number of correct answers predicted by the model.
- False Positives (FP) are the cases where the model predicts an answer incorrectly.
- False Negatives (FN) are the cases where the model fails to predict a correct answer.

A higher F1 score indicates a better balance between precision and recall, demonstrating the model's ability to provide correct and relevant answers. It is a valuable metric for evaluating the overall effectiveness of a QA model in understanding and responding accurately to user queries.

2.5.2 Exact Match

The Exact Match (EM) metric is a fundamental evaluation measure in question-answering (QA) tasks, providing a direct assessment of the model's ability to generate responses that match the ground truth answers exactly. The EM score is binary, indicating whether the predicted answer exactly matches the reference answer. It is calculated as follows:

$$EM = \begin{cases} 1, & \text{if the predicted answer is an exact match with the reference answer} \\ 0, & \text{otherwise} \end{cases}$$

In the context of QA tasks:

- An EM score of 1 indicates that the model's answer is an exact match with the reference answer.
- An EM score of 0 signifies that the model's answer does not match the reference answer precisely.

While the F1 score provides a more nuanced evaluation by considering partial matches, the EM score offers a strict criterion for correctness. A higher EM score indicates a better performance in terms of providing answers that precisely match the ground truth.

It's worth noting that a high EM score does not necessarily imply high F1 score, and vice versa. Therefore, both metrics are often reported together to provide a comprehensive evaluation of a QA model's performance. In summary, Exact Match is a valuable metric for evaluating the accuracy and precision of a QA model, particularly when the task requires providing answers that are exact matches to the ground truth.

2.5.3 Relevance Ranking

Regarding relevance ranking, transformers like XLM-RoBERTa are not only capable of question answering but can also be employed for tasks such as document ranking or relevance ranking. In such tasks, metrics like Mean Average Precision (MAP) or Normalized Discounted Cumulative Gain (NDCG) are commonly used to assess the model's ability to rank relevant documents or passages.

The relevance ranking evaluation involves presenting the QA system with a set of questions and evaluating its output based on the ranked relevance of the predicted answers. This ranking can be compared to a ground truth ranking, typically created by human annotators who assess the relevance of each potential answer. Commonly used metrics for relevance ranking include:

- **Mean Reciprocal Rank (MRR):** Measures the average rank of the first correct answer. The formula is given by:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

where $|Q|$ is the number of questions and rank_i is the rank of the first correct answer for question i.

- **Normalized Discounted Cumulative Gain (NDCG):** Accounts for both the relevance and the position of the correct answer in the ranking. The formula is given by:

$$NDCG = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{\text{DCG}_i}{\text{IDCG}_i}$$

where DCG_i is the Discounted Cumulative Gain for question i and IDCG_i is the Ideal Discounted Cumulative Gain for question i.

A high MRR and NDCG score indicates that the QA system is effective in ranking relevant answers higher in the list, reflecting a strong ability to prioritize correct and meaningful responses.

Relevance ranking evaluation provides insights into the system's performance beyond correctness, considering the importance of presenting the most relevant answers prominently. However, it's essential to use a combination of metrics, including traditional accuracy metrics like F1 score and EM, to comprehensively assess the QA system's overall performance. In conclusion, incorporating relevance ranking metrics into the evaluation of a QA system offers a more nuanced understanding of its ability to prioritize and present relevant information, aligning with real-world scenarios where multiple plausible answers exist.

Chapter 3

METHODOLOGY

In the Methodology chapter, we aim to introduce key details about our approach. This encompasses insights into our dataset, the utilization of the XLM-RoBERTa base model, and the incorporation of LoRA and QLoRA techniques employed to optimize the training process.

3.1 Dataset

During the training process, we utilize a composite dataset comprising three extensive Vietnamese datasets, namely:

- Translated version of the Stanford Question Answering Dataset (SQuAD)
- UIT-ViQuAD (developed by UIT)
- MultiLingual Question Answering
- mailong25

This is the information about our dataset

- **In total: 15466**
- **Train set:** 13919 samples
- **Valid:** 1547 samples

Vietnamese is a language with few resources for natural language processing. To tackle this challenge, our emphasis lies in the finetuning of models tailored specifically for the Vietnamese language. In this section, we would like to give a detailed exploration of the Vietnamese dataset we employ, the UIT-ViQuAD dataset.

The UIT-ViQuAD dataset introduced by (Nguyen et al., 2020) consists of 2,783 multiple-choice questions and answers based on a set of 417 Vietnamese texts which are used for evaluating the reading comprehension skills of 1st to 5th graders. However, this dataset is relatively small in size to evaluate deep learning models for the Vietnamese MRC. UIT-ViQuAD comprises 23,074 human-generated question–answer pairs based on 5,109 passages from 174 Vietnamese Wikipedia articles. [4]


```
"qas": [
  {
    "question": "Phạm Sơn Dương, con trai của Phạm Văn Đồng, đang giữ chức vụ gì?",
    "answers": [
      {
        "answer_start": 101,
        "text": "thiếu tướng Quân đội Nhân dân Việt Nam, phó giám đốc Viện Khoa học và Công nghệ Quâ
      }
    ],
    "id": "uit_01__05272_2_1"
  },
  {
    "question": "Phạm Văn Đồng từng cố gắng đưa bà Cúc đến nơi nào để chữa bệnh?",
    "answers": [
      {
        "answer_start": 565,
        "text": "Trung Quốc, Liên Xô"
      }
    ],
```


Figure 3.1: UIT-ViQuAD Dataset

3.1.1 Overall statistics

The statistics of the training (Train), development (Dev) and test (Test) sets of UIT-ViQuAD dataset are described in this below table. The number of questions of UIT-ViQuAD is 23,074 in JSON format. In the table, the numbers of articles and passages, the average lengths of questions and answers, and vocabulary sizes are also presented.

	Train	Dev	Test	All
Number of articles	138	18	18	174
Number of passages	4,101	515	493	5,109
Number of questions	18,579	2,285	2,210	**23,074**
Average passage length	153.9	147.9	155.0	153.4
Average question length	12.2	11.9	12.2	12.2
Average answer length	8.1	8.4	8.9	8.2
Vocabulary size	36,174	9,184	9,792	41,773

Figure 3.2: Overview statistics of the UIT-ViQuAD dataset

3.1.2 Type-based analysis

To gain more in-depth insights into the evaluation of the machine models and humans in Vietnamese, the authors analyze their performances in terms of different linguistic aspects such as length-based (question length, answer length, and passage length) and type based (question type, answer type, and reasoning type). Because Vietnamese is a subject-verb-object language similar to Chinese (Nguyen et al.,2018), Vietnamese question types in UIT-ViQuAD follow a manner in CMRC (Cuietal.,2019b). Thus, we also divide the questions into seven types: Who, What, When, Where, Why, How, and Others.

3.2 XLM-RoBERTa

In this study, we employ XLM-RoBERTa as the foundational model. To clarify, XLM-RoBERTa, a Transformers model, is utilized for our specific objectives.

Rather than training the model from scratch, we about to leverage the pre-trained parameters of XLM-RoBERTa and fine-tune them for the Vietnamese language in the context of a Question-Answering (QA) task. Although the selected base model was not originally designed for QA tasks, its architecture, rooted in transformers (as detailed in the Background), allows XLM to effectively meet our defined objectives. Particularly noteworthy is its adept handling of the Vietnamese language.

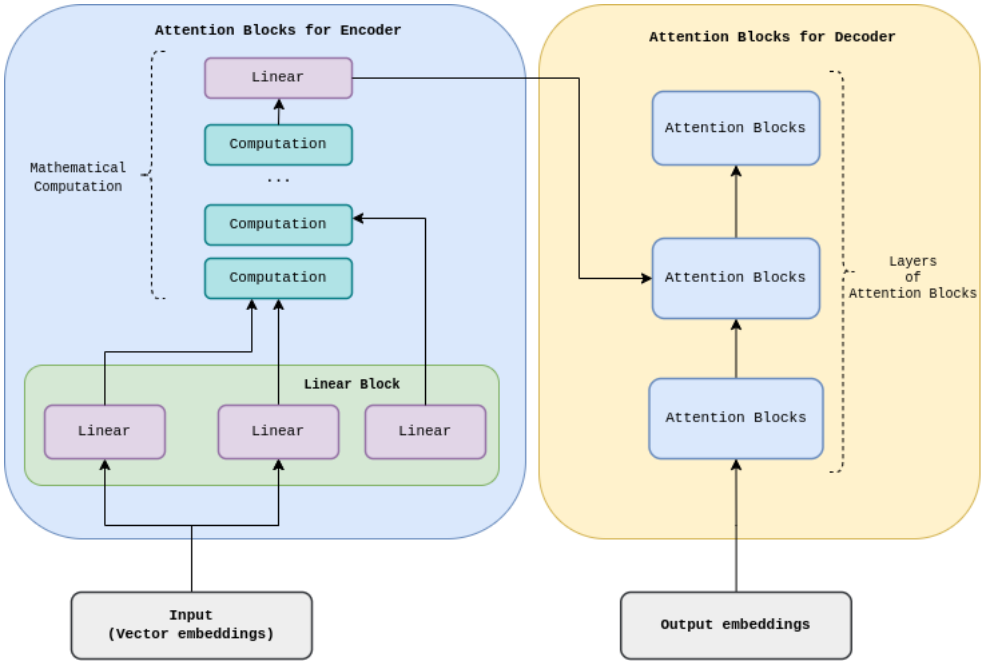

Figure 3.3: Architecture of XLM-RoBERTa based on Transformer Architecture

3.2.1 Finetuning XLM-RoBERTa for QA task

We use a tokenizer language model compatible with a pre-trained language model from XLM-R. Then, we proceed to process the data to made necessary features for the model with pre-written functions from the Hugging face for the SQuAD 2.0 dataset because the structure of the UIT-ViQuAD 2.0 dataset is the same as the SQuAD 2.0 dataset.

The model we use to train the MRC task is RoBERTa because it is a model that can take advantage of the parameters from PrLM XLM-R. We feed the training dataset for model and train. The model after being trained to determine if the question has an answer within the passage.

For answerable questions, the resulting span extracted from the context is calculated by taking the answer's start position and end position values with the highest probability among the positions that the model predicts. Cases where the model returns null text, include: The answer is within the question. The answer has a larger starting position than the ending position and a probability less than the null threshold. [6]

Training models for Vietnamese poses unique challenges due to language differences from English. Adjusting hyperparameters like epochs, batch size, and learning rate requires careful tuning to accommodate these distinctions. Our iterative approach involves testing to find the best parameters for optimal performance. Initially, we organize the training data into clusters: Context (C) with C1, . . . ,Cn, questions (Q) involving Q1, . . . , Qn, and answers (A) including A1, . . . , An. The test dataset is created with input clusters containing context and questions. The main task is training the model on the training dataset to recognize correct answers to questions, involving responses like null text or spans from the associated paragraph.

3.2.2 Model Description

First, we load the data from the train set into clusters: Context C = C1,..., Cn, question Q = Q1,..., Qn and answer A = A1,..., An. The test dataset with input includes the clusters: context and question. Our task is to train the model on the training dataset so that the model can find the correct answer to the question.

For the model to be able to understand human language, we need to convert the passages, questions, and answers from text to numbers.

We use a tokenizer language model compatible with a pre-trained language model from XLM-R. Then, we proceed to process the data to extract the necessary features for the model with pre-written functions from the Hugging face for the SQuAD 2.0 dataset because the structure of the UIT-ViQuAD 2.0 dataset is the same as the SQuAD 2.0 dataset.

The model we use to train the MRC task is RoBERTa because it is a model that can take advantage of the parameters from PrLM XLM-R. We feed the features extracted from the training dataset for model and train. The model after being trained to determine if the question has an answer within the passage. For answerable questions, the resulting span extracted from the context is calculated by taking the answer's start position and end position values with the highest probability among the positions that the model predicts.

Cases where the model returns null text, include: The answer is within the question. The answer has a larger starting position than the ending position and a probability less than the null threshold. [6]

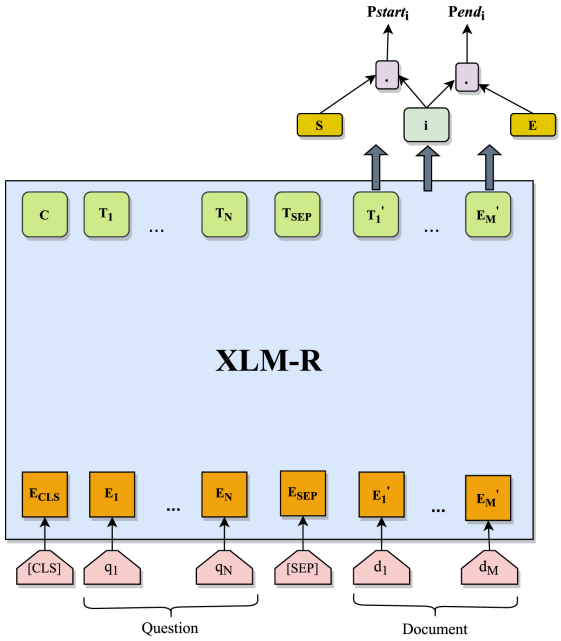

Figure 3.4: The answer-extractor component of the MRC4MRC model is built based on the XLM-Roberta model. [5]

The figure shows how the XLM-R pre-trained language model takes input and gives an answer. The model will receive the question and the context (document), tokenize them and separate the question and the context with [SEP]. After processing and calculating, the model finds the starting and ending positions of the answer. If the model predicts the question has no answer, it returns null text; otherwise, it will return multiple start and end positions. The selected answer is a span in the context with the start position value multiplied by the end position with the highest result.

3.3 Fine-tuning XLM-RoBERTa with LoRA

During the training process, due to hardware limitations, we encountered challenges related to insufficient memory and slow training speeds. Processing the Vietnamese language requires significantly more resources compared to other languages. Therefore, we applied two techniques to optimize the model: LoRA and QLoRA.

In this part, we would like to inform LoRA technique.

Low-Rank Adaptation, or LoRA is a method that freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks. Many sought to mitigate this by adapting only some parameters or learning external modules for new tasks. This way, we only need to store and load a small number of task-specific parameters in addition to the pre-trained model for each task.[3]

The LoRA paper's key insight is that during adaptation, weight matrices can have a low intrinsic rank. This means smaller matrices W_a and W_b can represent the same information as the original W. The rank of a matrix is not necessarily equal to its dimensions, but rather the number of linearly independent rows or columns.

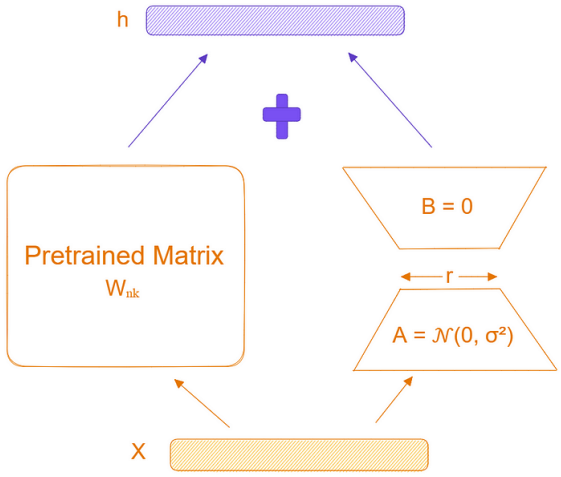

Figure 3.5: Modified forward pass using low-rank decomposition.

3.3.1 Rank and Decomposition

The rank of a matrix $(m \times n)$ is the number of linearly independent rows/columns. Matrix decomposition represents a large matrix as a product of smaller matrices. In LoRA, weight updates ΔW are decomposed as:

$$\Delta W = W_a W_b^T$$

where the rank of W_a and W_b is significantly smaller than m and n.

3.3.2 LoRA in Transformers

LoRA can be applied to different attention weights based on resources. The authors applied LoRA to all 96 layers of a model. The low intrinsic rank of weight matrices suggests LoRA's effectiveness. LoRA performs better or equally well as prefix-based approaches with the same parameters.

3.3.3 Benefits of LoRA

Integrating LoRA during fine-tuning offers several potential benefits:

- **Reduced Training Costs:** Compared to traditional fine-tuning, LoRA requires significantly fewer trainable parameters. This translates to smaller training data requirements, faster training times, and lower computational resources.
- **Improved Generalization:** By incorporating linguistic knowledge, the model may exhibit enhanced generalization to various linguistic patterns and structures.
- **Enhanced Task-Specific Performance:** LoRA may lead to better performance on downstream tasks by guiding the model to focus on relevant linguistic features.
- **Reduced Overfitting:** The inclusion of linguistic constraints may contribute to preventing overfitting, especially in scenarios with limited task-specific training data.

Fine-tuning XLM-RoBERTa with LoRA provides a method to leverage linguistic knowledge and improve the model's effectiveness on downstream tasks. This integration enhances the model's ability to understand and capture subtle linguistic nuances, contributing to its overall performance in various natural language processing applications.

3.4 Fine-tuning XLM-Roberta using QLoRA

This section explores fine-tuning XLM-Roberta using QLoRA, a quantization technique that enables efficient deployment on low-memory devices while preserving accuracy.[2]

Finetuning large language models requires a significant amount of GPU memory, which is expensive and complicated to set up. To fine-tune a large model like Llama 2, it requires 840 GB of GPU memory. Low-rank adapters can be used to adapt only a small portion of the neural network, reducing the memory requirement to 150 GB. QLoRA (Quantized LLMs with Low-Rank Adapters) is a method that quantizes the base model to 4 bits and adds low-rank adapters on top. With QLoRA, the memory requirement is reduced to 46 GB, making it feasible to fit on a single low-grade data center GPU.

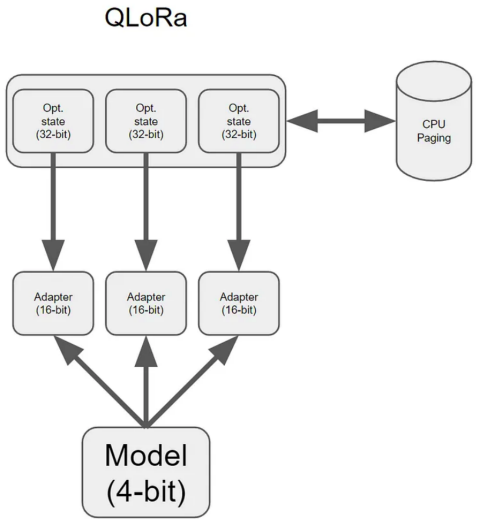

Figure 3.6: QLoRa works by first quantizing the LLM to 4-bit precision.

3.4.1 QLoRA for Efficient Fine-tuning

QLORA allows for finetuning a 65B parameter model on a single 48GB GPU while preserving full 16-bit finetuning task performance. The approach backpropagates gradients through a frozen, 4-bit quantized pretrained language model into Low Rank Adapters. QLoRA addresses the memory challenge by:

- ***Quantizing activations and weights to 4-bit precision:*** This significantly reduces memory usage compared to the standard 32-bit representation.
- ***Low-rank approximation of weight matrices:*** QLoRA decomposes weight matrices into smaller matrices, further reducing memory consumption and potentially accelerating computations.

In addition to reducing the size and improving the efficiency of neural networks, quantization can also help to improve their accuracy and robustness. By reducing the precision of the weights and activations, quantization can help mitigate the effects of noise and other imperfections in the data, leading to better generalization and performance on real-world tasks.

3.4.2 Benefits and Performance

Fine-tuning XLM-Roberta with QLoRA offers several advantages:

- **Reduced memory footprint:** Enables deployment on low-memory devices.
- **Potentially faster inference:** Lower precision and smaller weight matrices can lead to faster computations. Minimal accuracy loss: QLoRA's joint optimization maintains high performance on downstream tasks.

Fine-tuning XLM-Roberta with QLoRA provides a promising approach for deploying LLMs on low-memory devices. The ability to achieve significant compression while maintaining accuracy opens new possibilities for applying LLMs in resource-constrained environments.

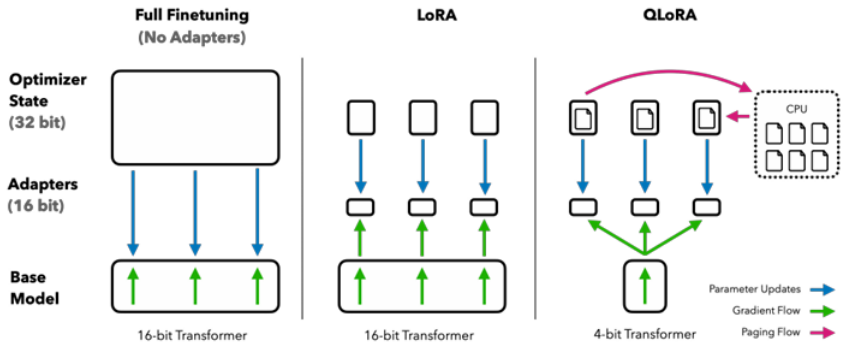

Figure 3.7: Different finetuning methods and their memory requirements. QLORA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Chapter 4

EXPERIMENTAL AND RESULTS

In this chapter, we will present the results and experiments conducted on our XLM-RoBERTa model to serve the QA task for Vietnamese. The evaluation will cover both the finetuned models, one without the application of QLoRA and the other with QLoRA applied.

4.1 Experimental Settings

We use a single NVIDIA Tesla K80 via Google Collaboratory to train our model. The baseline configuration provided by HuggingFace1 is used to fine-tune the machine reading comprehension components of the XLM-Roberta QLoRa model.

4.2 Result

We would show our results here.

4.2.1 Raw Training

Below are the results of the model with no optimize approach on it.

The XLM model operates with input pairs consisting of a question and an answer. The output includes Start Score and End Score, evaluated by calculating scores based on the co-occurrence of words in the input question and answer. These scores are then compared to the question-answer pairs in the dataset. A higher score indicates a better match for the given input question and answer pair.

With the Google Colab hardware we utilize, the XLM model, without the application of QLoRA, can only be trained for a maximum of 4 epochs. Additionally, it is not feasible to run additional inferences on the server due to Google Colab's time restrictions for GPU usage. Consequently, the model can be trained for a maximum of 3 epochs.

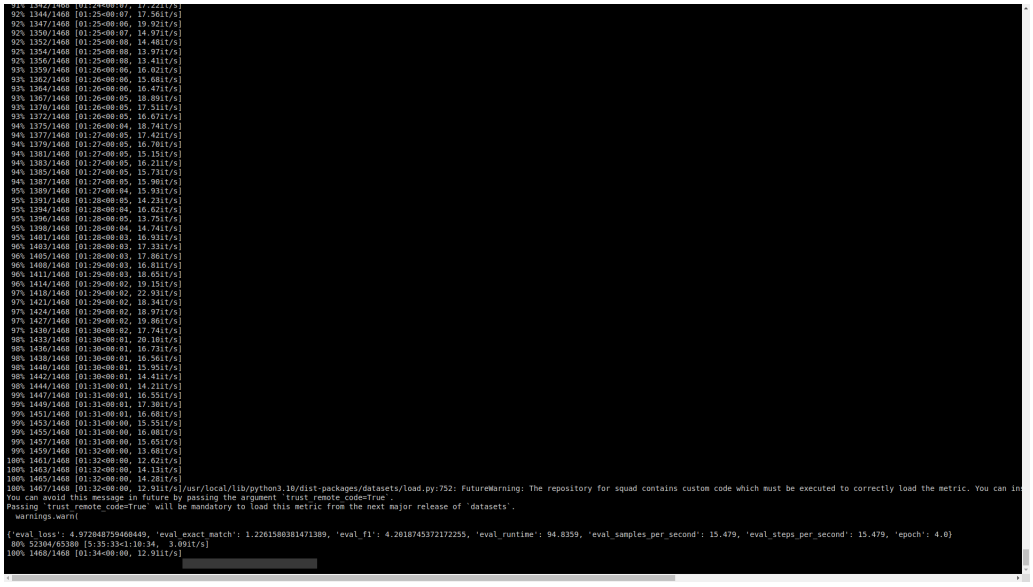

Figure 4.1: Finetune XLM with no QLoRA - 4 epochs

Below is the result after training XLM-RoBERTa model for 2 epochs. As we can see, the result is quite poor due to the low epochs training.

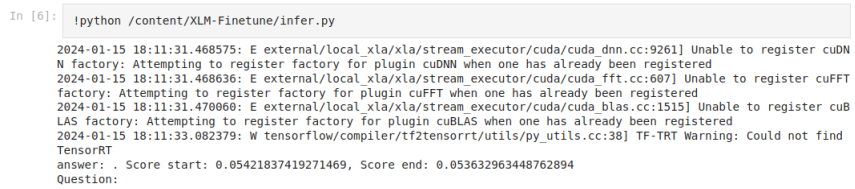

Figure 4.2: Raw finetuned XLM - Result

4.2.2 Applied QLoRA

Below are the results of the model after applying QLoRA.

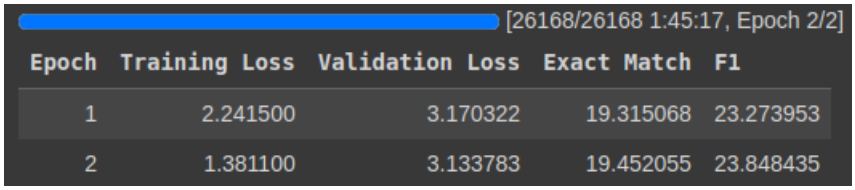

Figure 4.3: Finetune XLM with QLoRA - 2 epochs

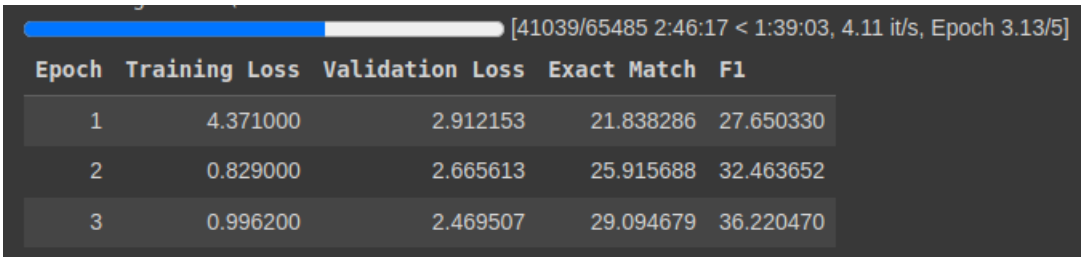

Figure 4.4: Finetune XLM with QLoRA - 3 epochs

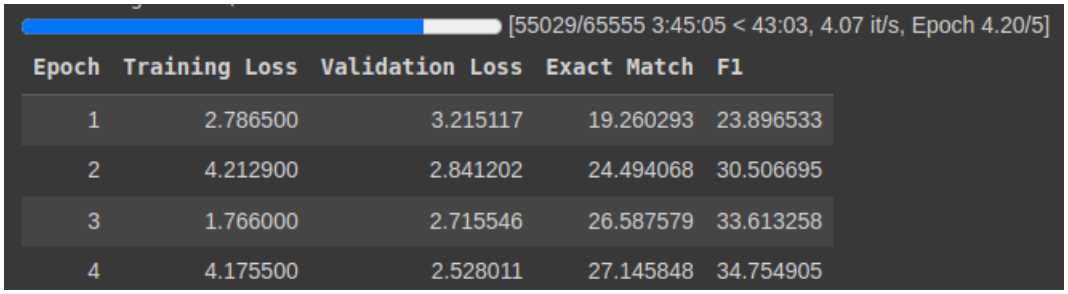

Figure 4.5: Finetune XLM with QLoRA - 4 epochs

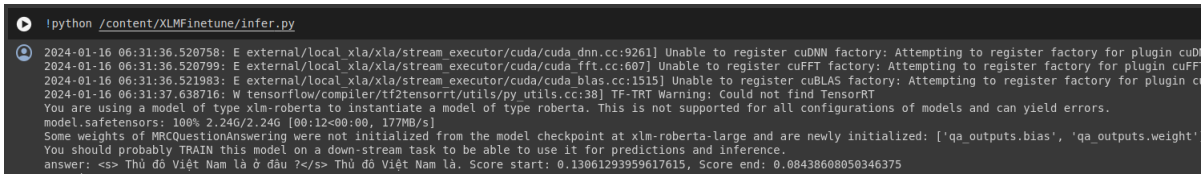

Figure 4.6: Finetuned XLM with QLoRA - Result

Chapter 5

CONCLUSION

In the context of our report, the outcomes of fine-tuning the XLM model to establish a Vietnamese QA model are discernible. However, it's essential to note that the achieved results are not optimal, primarily due to limitations in hardware infrastructure.

Turning our attention to the Optimized model with QLoRA, a closer examination, as depicted in the visual representation below, reveals a notable acceleration in the training process—approximately 1.5 times faster compared to conventional model training. Furthermore, the results obtained with an equivalent number of epochs showcase improved performance, signifying the effective application of QLoRA in optimizing the training workflow.

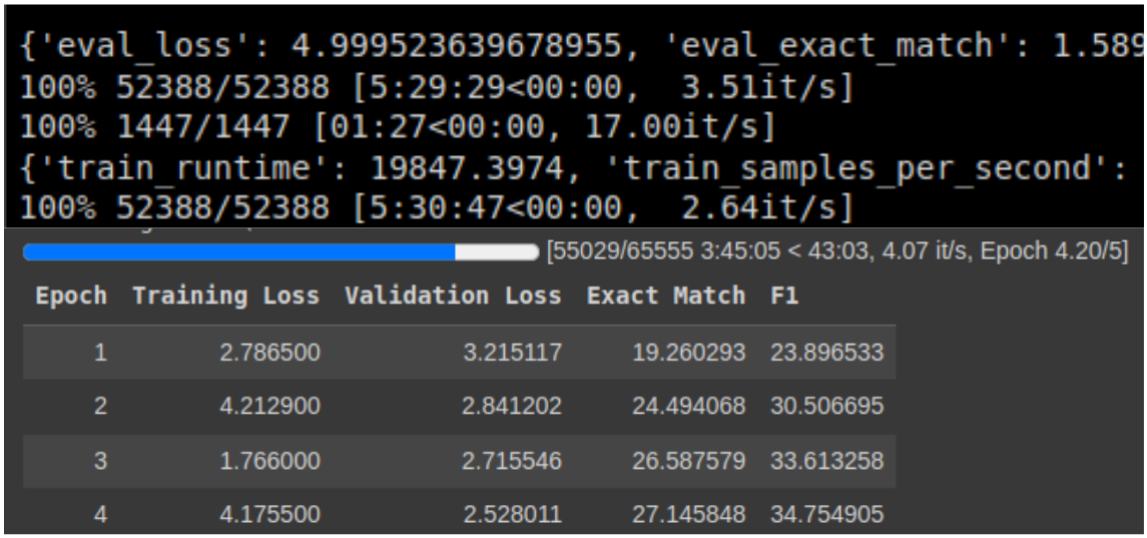

Figure 5.1: Combination of Figure 4.5 and Figure 4.2

Looking forward, the research team envisions the future trajectory of the project. One crucial aspect involves continued model training with more robust hardware to potentially unlock further enhancements in the current results. Additionally, recognizing the unique demands of the Vietnamese language task, the team proposes an

innovative two-step fine-tuning strategy. After the initial fine-tuning that establishes the Vietnamese QA model, if the model demonstrates satisfactory performance, a subsequent fine-tuning iteration with a new dataset using a similar methodology will be undertaken. This iterative refinement aims to elevate the model's efficacy even further.

In summary, our report not only sheds light on the present findings but also outlines a strategic roadmap for future endeavors, emphasizing continual improvement and innovation in the pursuit of an optimized Vietnamese QA model.

All information about the project, including the paper and code files, can be retrieved from the following GitHub link: https://github.com/phuonganhcorn/XLM_Finetune_QLora

For people who want to try our pre-trained model, can run the code snippet down below to download our model from HuggingFace

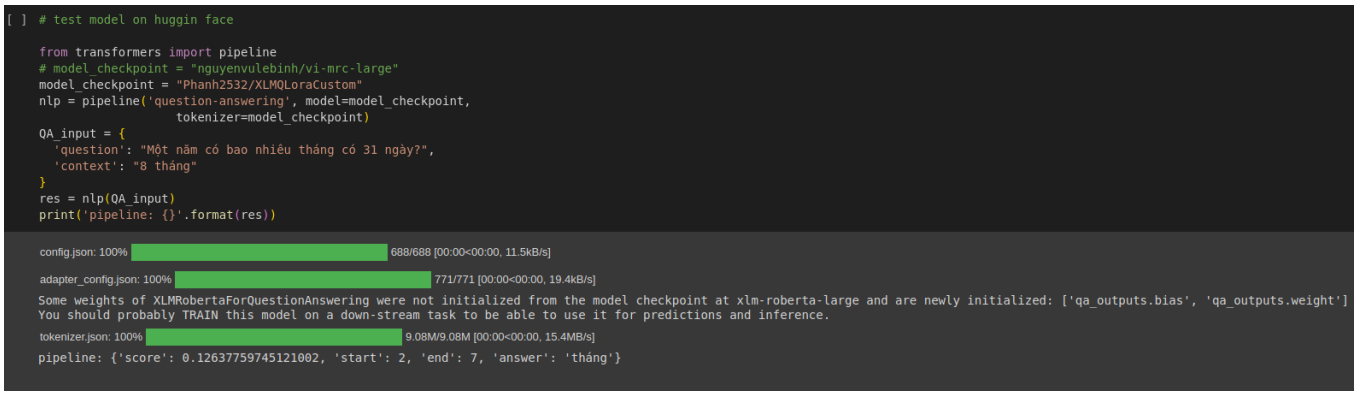

Figure 5.2: Combination of Figure 4.5 and Figure 4.2

Bibliography

[1] Zahra Abbasiantaeb and Saeedeh Momtazi. "Text-based question answering from information retrieval and deep neural network perspectives: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11.6 (2021), e1412.

[2] Tim Dettmers et al. "Qlora: Efficient finetuning of quantized llms". In: *arXiv preprint arXiv:2305.14314* (2023).

[3] Edward J Hu et al. "Lora: Low-rank adaptation of large language models". In: *arXiv preprint arXiv:2106.09685* (2021).

[4] Kiet Nguyen et al. "A Vietnamese Dataset for Evaluating Machine Reading Comprehension". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Ed. by Donia Scott, Nuria Bel, and Chengqing Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 2595–2605. DOI: 10.18653/v1/2020.coling-main.233. URL: https://aclanthology.org/2020.coling-main.233.

[5] Kiet Nguyen et al. "Vireader: A wikipedia-based vietnamese reading comprehension system using transfer learning". In: *Journal of Intelligent and Fuzzy Systems* 41 (2021), pp. 1–19.

[6] Văn Nhân Đặng and Minh Lê Nguyễn. "ViMRC-VLSP 2021: Using XLM-RoBERTa and Filter Output for Vietnamese Machine Reading Comprehension". In: *VNU Journal of Science: Computer Science and Communication Engineering* (2022).