VIETNAM NATIONAL UNIVERSITY – HOCHIMINH CITY
INTERNATIONAL UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

USING EFFICIENT FINE-TUNING OF BERT MODELS TO DESIGN A SYSTEM FOR VIETNAMESE IN HEALTH TEXT MINING

BY
NGUYEN NHAT MINH

A THESIS PROJECT SUBMITTED TO THE SCHOOL OF ELECTRICAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ELECTRICAL ENGINEERING

HO CHI MINH CITY, VIET NAM
2024

USING EFFICIENT FINE-TUNING OF BERT MODELS TO DESIGN A SYSTEM FOR VIETNAMESE IN HEALTH TEXT MINING

BY

NGUYEN NHAT MINH

Under the guidance and approval of the committee, and approved by its members, this thesis has been accepted in partial fulfillment of the requirements for the degree.

Approved:

_____
Chairperson

_____
Committee member

_____
Committee member

_____
Committee member

_____
Committee member

HONESTY DECLARATION

My name is Nguyen Nhat Minh. I would like to declare that, apart from the acknowledged references, this thesis either does not use language, ideas, or other original material from anyone; or has not been previously submitted to any other educational and research programs or institutions. I fully understand that any writings in this thesis contradicted to the above statement will automatically lead to the rejection from the EE program at the International University – Vietnam National University Ho Chi Minh City.

Date:

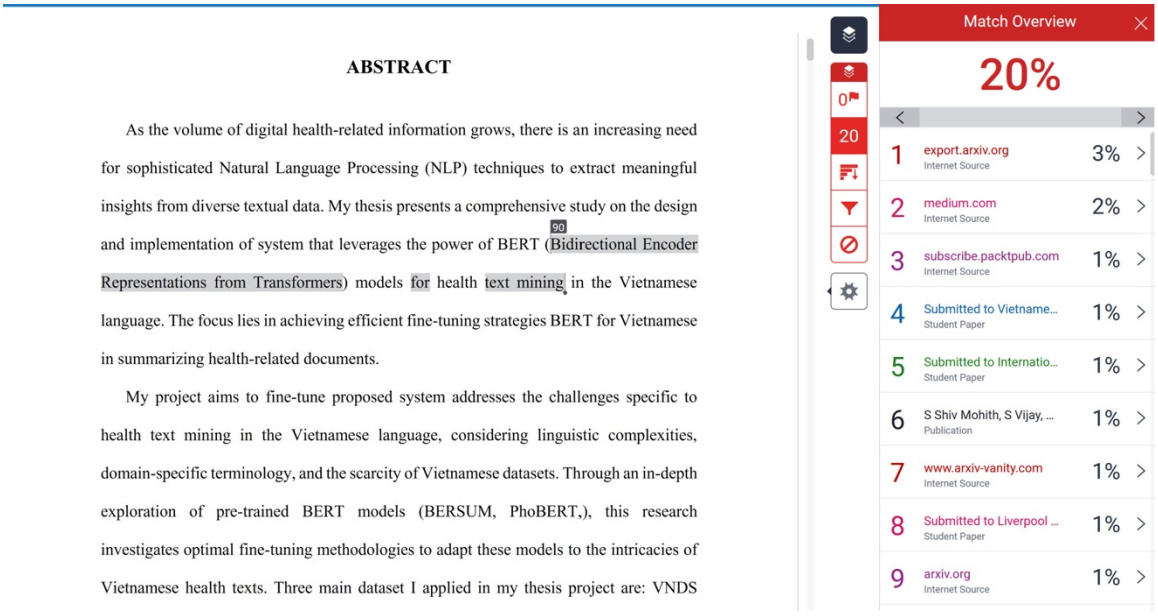
Student's Signature

Nguyen Nhat Minh

TURNITIN DECLARATION

Name of Student: Nguyen Nhat Minh

Date: Fri, Jan 26th, 2024



ABSTRACT

As the volume of digital health-related information grows, there is an increasing need for sophisticated Natural Language Processing (NLP) techniques to extract meaningful insights from diverse textual data. My thesis presents a comprehensive study on the design and implementation of system that leverages the power of BERT (Bidirectional Encoder Representations from Transformers) models for health text mining in the Vietnamese language. The focus lies in achieving efficient fine-tuning strategies BERT for Vietnamese in summarizing health-related documents.

My project aims to fine-tune proposed system addresses the challenges specific to health text mining in the Vietnamese language, considering linguistic complexities, domain-specific terminology, and the scarcity of Vietnamese datasets. Through an in-depth exploration of pre-trained BERT models (BERSUM, PhoBERT,), this research investigates optimal fine-tuning methodologies to adapt these models to the intricacies of Vietnamese health texts. Three main dataset I applied in my thesis project are: VNDS

Advisor Signature

Student Signature

ACKNOWLEGMENT

It is with deep gratitude and appreciation that I acknowledge the professional guidance of MEng. Do Ngoc Hung. His constant encouragement and support helped me to achieve my goal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS AND NOTATIONS

CNN	:	Convolutional Neural Networks
SVM	:	Support Vector Machine
LSTM	:	Long short-term memory
IC	:	Image Captioning
MLP	:	Multilayer Perceptron
NLTK	:	Natural Language Toolkit
RNN	:	Recurrent Neural Network
FAQ	:	Frequently Asked Questions
NSP	:	Next Sentence Prediction
MLM	:	Masked Language Model
CLS	:	Classify Token
SEP	:	Separator Token

ABSTRACT

As the volume of digital health-related information grows, there is an increasing need for sophisticated Natural Language Processing (NLP) techniques to extract meaningful insights from diverse textual data. My thesis presents a comprehensive study on the design and implementation of system that leverages the power of BERT (Bidirectional Encoder Representations from Transformers) models for health text mining in the Vietnamese language. The focus lies in achieving efficient fine-tuning strategies BERT for Vietnamese in summarizing health-related documents.

My project aims to fine-tune proposed system addresses the challenges specific to health text mining in the Vietnamese language, considering linguistic complexities, domain-specific terminology, and the scarcity of Vietnamese datasets. Through an in-depth exploration of pre-trained BERT models (BERSUM, PhoBERT,), this research investigates optimal fine-tuning methodologies to adapt these models to the intricacies of Vietnamese health texts. Three main dataset I applied in my thesis project are: VNDS (VNDS: A Vietnamese Dataset for Summarization), AD (Acronym Disambiguation), FAQ (Frequency Asked Questions).

Summarization in Natural Language Processing is a trending topic, which has interested many researchers in recent years (ChatGPT, Bard,…) and has much work to do to improve the reliability and the stability of the system or to build friendly GUI for the users, especially for the doctors in the health scientist. I will research depending on the previous method and datasets to choose the best method and datasets for my thesis project.

CHAPTER I

INTRODUCTION

In this chapter, we will introduce our reality and motivation in the theoretical background part, Goals and objectives are shown in the next part, and finally is about the structure of my thesis project.

1.1. Theoretical background

In recent years, the fusion of Natural Language Processing (NLP) and health-related data has emerged as a transformative force in healthcare research. NLP, a branch of Artificial Intelligence (AI), empowers computers to comprehend, interpret, and generate human-like language. Within the healthcare domain, mining valuable insights from a vast sea of unstructured health texts has become a pivotal challenge.

The goal of this research is the utilization of BERT (Bidirectional Encoder Representation from Transformers) models, which stand out for their ability to capture contextual information in language data. In simple terms, these models understand the meaning of words not just by looking at their immediate neighbors but by considering the entire sentence context. This characteristic makes BERT particularly potent for tasks like summarize health-text, where the context is crucial.

In summary, the theoretical framework underscores the synergy between NLP, BERT models, and the intricacies of Vietnamese language in the context of health text mining. The proposed system aims to bridge language barriers, unlocking valuable insights from a sea of health information and contributing to the evolution of intelligent healthcare solutions.

1.2. Goal and objectives

The main purpose of this thesis project is to design a system for summarizing documents such as questions relative to the health scientist for Vietnamese. The software enables the users input the documents in the health scientist and the output is the summarized documents.

The software development includes the following objectives:

1.2.1 Collecting and preprocessing data

1.2.2 Applying our dataset with different language models

1.2.3 Fine-tuning our model for the better result

1.2.4 Testing and evaluating our model

1.3. The structure of this thesis project

There are eight chapters that make up the thesis's substance. The first chapter discusses the global challenges currently surrounding impaired people. The purpose and four main objectives are also presented in this chapter. The second chapter is about design requirements and norms. The third chapter provides a management plan for doing tasks and timelines for each task by showing our Gantt chart. Next, we will show our research about the paper relative to our project in chapter four. The methodology for each of the objectives is provided in the fifth chapter. The results are presented in the sixth chapter to show the effect of the main parameter on our model. The next chapter provides the conclusions for the best method, discussions, and more tasks we will do to enhance the system’s performance. The last chapter states about business, social and ethical consideration.

CHAPTER II

DESIGN SPECIFICATIONS AND STANDARDS

2.1. Design Specifications and Standards

2.1.1. System Architecture

The system architecture for the Vietnamese Summarization project will be based on the Transformer architecture. The Transformer is a state-of-the-art architecture that has demonstrated excellent performance in natural language processing tasks. It leverages self-attention mechanisms to capture contextual dependencies and relationships between words in both the question and image modalities. The architecture will consist of an image encoder, a question encoder, and an answer decoder. The image encoder will utilize a convolutional neural network (CNN) to extract visual features from the input image, while the question encoder will employ Transformer layers to process the textual input. The answer decoder will generate the final answer based on the encoded image and question information.

2.1.2. Data requirements

For training and evaluation, the Vietnews, FAQ dataset will be used. The dataset contains a collection of images along with associated questions and corresponding answers. The images are indoor scenes, and each question is related to specific visual aspects of the scene. The data will be preprocessed by tokenizing the questions and answers, converting them into numerical representations, and aligning them with their corresponding images. The dataset will be split into training, validation, and testing sets, ensuring an appropriate distribution of samples across the subsets.

2.1.3. Performance Metrics

Evaluating the effectiveness of text summarization models, particularly those based on BERT (Bidirectional Encoder Representations from Transformers), involves the careful consideration of various performance metrics. These metrics play a crucial role in assessing the model's ability to generate concise and informative summaries. The choice of metrics is pivotal in quantifying the quality of the generated summaries and providing insights into the strengths and limitations of the BERT-based summarization approach.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE metrics, including ROUGE-N (unigram, bigram, etc.) and ROUGE-L (Longest Common Subsequence), are widely employed to measure the overlap between the generated summaries and reference summaries. By assessing precision, recall, and F1 scores, ROUGE metrics offer a comprehensive evaluation of the content overlap at different levels of granularity, providing a nuanced understanding of the summarization model's performance.

BLEU (Bilingual Evaluation Understudy): BLEU is a metric commonly used in machine translation but is also applicable to text summarization. It assesses the quality of generated summaries by comparing them to reference summaries based on n-gram precision. Higher BLEU scores indicate better alignment with the reference summaries, offering insights into the model's ability to capture key information.

METEOR (Metric for Evaluation of Translation with Explicit ORdering): METEOR is a metric that considers both precision and recall, incorporating additional features such as stemming and synonymy. This metric provides a holistic evaluation of summary quality, taking into account various linguistic aspects beyond exact word matches..

Semantic Evaluation Metrics: Beyond traditional n-gram-based metrics, assessing the semantic quality of summaries is crucial. Metrics such as Semantic Similarity and Embedding-based Metrics leverage pre-trained embeddings and semantic similarity calculations to gauge the contextual relevance and coherence of generated summaries.

These performance metrics collectively contribute to a comprehensive evaluation framework for text summarization models utilizing BERT. By systematically analyzing the results across these metrics, researchers and practitioners can gain valuable insights into the strengths and weaknesses of the BERT-based summarization approach, guiding further refinement and optimization of the model for enhanced summarization performance.

2.2. Realistic Constraints

2.2.1. Computation Resources

2.2.1.1. Hardware

In this senior project, we do all tasks from coding to training models by using MacBook Air M1 with an Apple M1 chip (8-core CPU with 4 efficiency cores and 4 performance cores), 8 GB RAM, and 256GB SSD.

2.2.1.2. Software

PyCharm CE is an application we used to develop our code. There are two main libraries we use: Transformer, Pytorch, and some libraries use for computation such as NumPy, Pandas, … Moreover, we also used the scikit-learn library to import performance metrics such as: accuracy, ROUGE score and f1_score.

2.2.1.2. Time Constraints

The successful execution of a text summarization project is contingent upon adherence to predefined time constraints, which play a pivotal role in project planning and delivery. Time

constraints encompass various aspects, including the time allocated for model development, training, fine-tuning, and evaluation. Additionally, considerations must be given to real-time processing requirements for live summarization applications or systems.

2.2.1.2.1. Model development Time

The time dedicated to conceptualizing, designing, and developing the text summarization model is a critical aspect of project planning. It involves tasks such as selecting the appropriate architecture, customizing parameters, and integrating any pre-trained models, such as BERT. Balancing model complexity and development time is essential to meet project deadlines without compromising the quality of the summarization outputs.

2.2.1.2.2. Training and Fine-tuning Time

The training phase, where the model learns from the dataset, and fine-tuning, which adapts the model to specific summarization tasks, are resource-intensive processes. Efficient allocation of time to these stages is crucial for achieving a well-performing summarization model. Techniques such as transfer learning with pre-trained language models can expedite training, but finding the optimal balance between training time and model accuracy is key.

2.2.1.2.3. Real-Time Processing Requirements

In scenarios where the text summarization system is intended for real-time applications, time constraints become even more stringent. The model must generate concise and coherent summaries within the time frames acceptable for user interaction. Optimizations, parallel processing, and efficient algorithms are integral to meeting real-time processing requirements without sacrificing summarization quality.

2.2.1.3. Accuracy Expectations:

The text summarization system should aim for high accuracy in answering questions related to the input documents. The expected accuracy should be defined based on the project requirements and the desired level of performance. Consideration should be given to the complexity of the VNDS, FAQ dataset and the inherent challenges of understanding textual context

CHAPTER III

PROJECT MANAGEMENT

3.1. Project Schedule

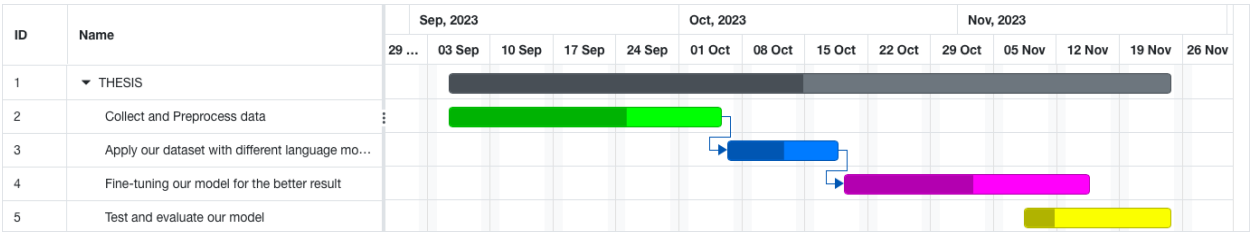


Figure 3. 1. Gantt chart for our project

In this section the part, the project schedule outlines the timeline, and tasks necessary for the successful completion of the project. It provides a structured approach to organizing and tracking project activities from start to finish.

The project schedule was effectively followed, resulting in the successful completion of all activities within the allocated timeframes.

CHAPTER IV

LITERATURE REVIEW

We have two main parts in this chapter. The first part will cover current research and the second part is the fundamentals of attention mechanism and model evaluation in this thesis project.

4.1. Literature review of current research

There are five papers for reference that I used in my thesis project. The first is about the general information about Natural Language Processing, relative to the application of Large Language Models. The second research was about Natural Language Processing for smart healthcare, especially in clinic. The third research introduces state of the art pre-training Language models and their applications. The technology about Transformer architecture and Bert are introduced clearly in the fourth part. Last but not least, we will cover research about the datasets for Text Summarization such as FAQ and VNDS. All the papers will be described in detail in the next sections.

4.1.1. A Survey of Large Language Models [1]

In this research, they cover some content related to the concept of Language and the challenge to develop the algorithms in Natural Language Processing through state-of-the-art models such as pre-training Transformer models, BERT, Chat GPT, … Furthermore, they also summarize documents for developing Large Language Models.

In technical terms, Language Modeling (LM) stands out as a key method for improving how machines understand language. Essentially, LM works by trying to understand how likely it is for certain words to come together in a sentence, predicting the chances of future words or figuring out missing ones. People have paid a lot of attention to studying LM, and its progress can be

divided into four main stages: Statistical language models (SLM), Neural language models (NLM), Pre-trained language models, and Large language models.

Statistical language models (SLM): These models were developed using statistical learning methods that gained popularity in the 1990s. The basic idea was to create a word prediction model based on the Markov assumption, where the prediction of the next word relies on the most recent context. Models with a fixed context length, like bigram and trigram language models, are known as n-gram language models. These models have found widespread use in improving performance in tasks like information retrieval (IR) and natural language processing (NLP). However, they face a challenge known as the curse of dimensionality. Estimating high-order language models is tough because there are so many transition probabilities to calculate. To tackle this, special smoothing strategies like back-off estimation and Good–turing estimation have been introduced to handle the issue of sparse data.

Neural language models (NLM): determine the likelihood of word sequences using neural networks like multi-layer perceptron (MLP) and recurrent neural networks (RNNs). An influential contribution from [1] introduced the concept of distributed representation of words. It established a word prediction function based on aggregated context features, specifically the distributed word vectors. Expanding on the idea of effectively learning features for text data, a general neural network approach was developed to create a comprehensive end-to-end solution for various natural language processing (NLP) tasks [2]. Additionally, word2vec proposed a simplified shallow neural network to learn distributed word representations, proving highly effective across a range of NLP tasks. These advancements marked the beginning of using language models for representation learning, extending beyond just modeling word sequences and significantly impacting the field of NLP.

Pre-trained language models (PLM): represent an early effort to enhance language understanding. ELMo , for instance, aimed to capture context-aware word representations by initially pre-training a bidirectional LSTM (biLSTM) network (as opposed to fixed word representations). Subsequently, fine-tuning of the biLSTM network was performed based on specific downstream tasks. Another notable development is BERT, utilizing the highly parallelizable Transformer architecture with self-attention mechanisms. BERT pre-trains bidirectional language models with specially designed tasks on extensive unlabeled datasets. These pre-trained, context-aware word representations have proven highly effective as general-purpose semantic features, significantly elevating the performance of various NLP tasks.

This approach has inspired a considerable amount of subsequent work, establishing the "pre-training and fine-tuning" learning paradigm. Numerous studies on PLMs have emerged, introducing different architectures (such as GPT-2 and BART) or enhanced pre-training strategies. Within this paradigm, adapting PLMs to various downstream tasks often involves fine-tuning.

Large language models (LLM): Large Language Models (LLMs) are a type of artificial intelligence (AI) model that is trained on massive amounts of data to understand and generate human-like text. These models belong to the broader category of natural language processing (NLP) and are designed to handle various language-related tasks, such as text generation, translation, summarization, question answering, and more. LLMs are typically pre-trained on a large dataset to learn general language patterns. After pre-training, they can be fine-tuned on specific tasks with smaller, task-specific datasets. This enables the models to adapt to particular domains or applications. LLMs have been applied to various applications, including natural language understanding, text completion, language translation, sentiment analysis, code

generation, and more. They have gained attention for their ability to perform at or near human-level performance on certain language-related tasks.

4.1.2. Natural Language Processing for Smart Healthcare [2]

This research by Bingg Zhou, and his partners. This paper reviews one important AI tool – Natural Language Processing, which helps computers understand human language, and applications in smart healthcare, focusing on different techniques and real-life situations. It discusses technical aspects like NLP approaches and applications in areas such as hospitals, personal care, and drug development. Finally, they mention what we still need to learn and study more in the future.

4.1.2.1. Introduction

Natural Language Processing (NLP) play a pivotal role in the domain of health text mining, providing the tools and techniques necessary for extracting valuable insights from the vast amount of unstructured textual data present in healthcare. This section explores the importance of NLP in healthcare and its various applications.

4.1.2.2. Importance of NLP Healthcare

Healthcare data encompasses a wide range of unstructured information, including clinical notes, electronic health records (EHRs), and medical literature. NLP techniques are essential for converting this unstructured data into a structured format that can be analyzed and utilized effectively. The significance of NLP in healthcare is evident in several key areas:

a) Clinical Decision Support

NLP algorithms assist healthcare professional in extracting relevant information from patient records, enabling better-informed decision-making regarding diagnoses and treatment plans.

b) Disease Surveillance

By analyzing data from diverse sources such as social media, new articles, and medical reports, NLP contributes to the detection of disease outbreaks, monitoring health trends, and improving public health surveillance.

c) Biomedical Research

NLP aids researchers in extracting valuable insights from the extensive body of biomedical literature, facilitating the identification of relationships between genes, proteins, and diseases.

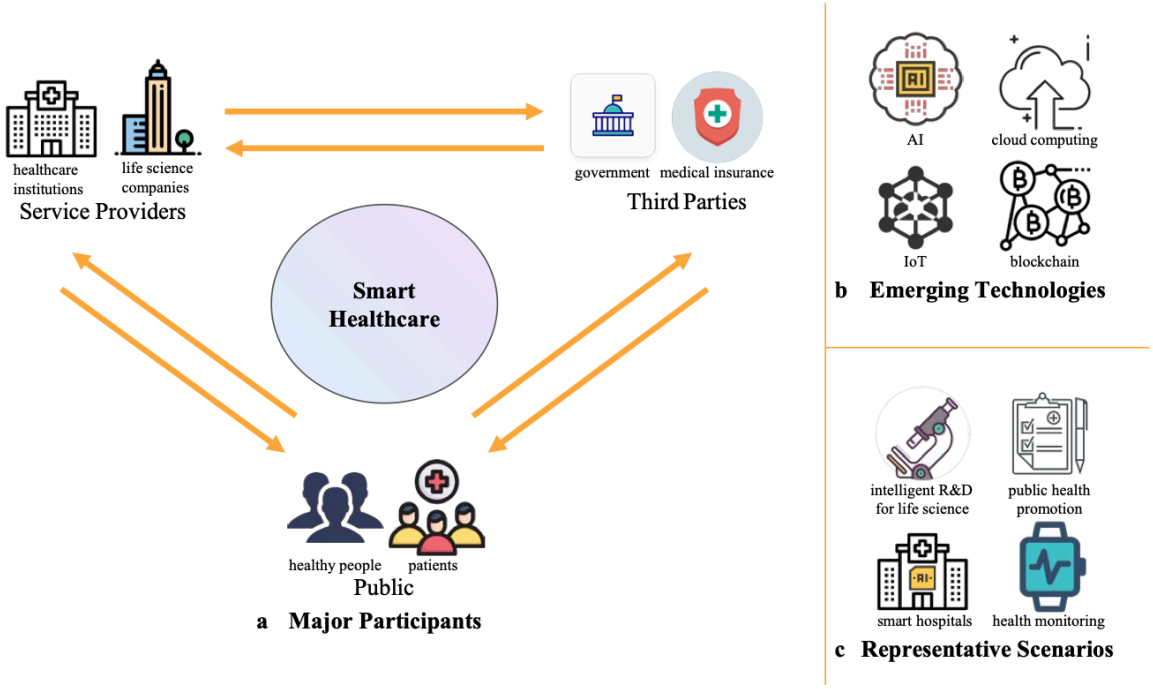


Figure 4. 1. Smart healthcare

The figure shows that the smart healthcare involving the public, healthcare providers, and other participants. New technologies like artificial intelligence and the internet of things (IOT) make smart healthcare possible. This includes things like smart hospitals, health monitoring, and research for improving life science and public health.

4.1.2.3. Applications of NLP in Healthcare

In this section, they show that the applications of NLP in healthcare are diverse and impactful, showcasing the adaptability and versatility of these techniques:

a) Clinical Entity Recognition

NLP algorithms excel in recognizing and classifying clinical entities, including diseases, medications, and symptoms, within textual data. This is crucial for deriving structured information from unstructured clinical notes.

b) Information Extraction form EHRs

NLP enables the extraction of valuable insight from electronic health records, automating the retrieval of patient information, treatment histories, and diagnostic reports to enhance patient care.

c) Sentiment Analysis in Patient Feedback

NLP techniques are employed to analyze patient feedback, reviews, and surveys, providing healthcare providers with valuable insights into patient experiences and satisfaction.

4.1.2.4. Challenges in NLP for Healthcare Text

Despite the significant advancements in NLP, the healthcare domain presents unique challenges:

a) Medical Jargon and Terminology

Healthcare texts often contain specialized terminology and medical jargon, posing challenges for NLP models to accurately interpret and extract relevant information.

b) Data Privacy and Security

Healthcare data is highly sensitive, and strict privacy regulations govern its usage. NLP models must be designed to handle such data securely while ensuring patient confidentiality.

c) Lack of Standardization

Variability in writing styles, abbreviations, and documentation practices across healthcare institutions presents a challenge for NLP models aiming to generalize across diverse datasets.

In this paper, we understand the role of NLP in health text mining is crucial for developing effective solutions that can harness the vast amount of textual information in the healthcare domain. This foundation sets the stage for the subsequent discussion on the application of pre-training language models, specifically BERT, in addressing these challenges.

4.1.3. State of the Art in Pre-trained Language Models [3]

This article talks about Large Language Models, a special technology in Natural Language Processing (NLP). These models, like GPT-3.5, use advanced machine learning to understand and create text that looks like human language. The article explores their important features, how they are trained, and what they are used for. It also discusses how these models affect different NLP tasks and contribute to scientific research. The reason these models are successful is because they can understand connections in text over long distances, something that was hard for earlier NLP methods. This ability helps them create sentences that make sense and fit the context, making the language they generate sound more like how humans speak.

4.1.3.1. Early Rule-Based Systems

. In the beginning of NLP, there were rule-based systems. These systems used rules and patterns created by people, but they had trouble with complex language and couldn't learn from data. This made them not very flexible for real-world language tasks.

4.1.3.2. Statistical Language Models

. The improvement of statistical language models changed NLP significantly. These models used probabilities and machine learning, with the n-gram approach predicting word chances based on earlier words. Although better than rule-based systems, these models still had trouble with understanding long connections between words and context.

4.1.3.3. Recurrent Neural Networks (RNNs)

. In the mid-2000s, scientists started using neural networks, especially Recurrent Neural Networks (RNNs), to understand language. RNNs became well-liked for handling sequences like text because they could remember information from earlier words. They introduced hidden states to keep track of context while looking at each word. Even though they performed better than other models, RNNs had problems like the vanishing gradient issue and were slow to learn because they had to go through words one by one.

4.1.3.4. Long Short-Term Memory (LSTM)

. To solve a problem in RNNs, they created Long Short-Term Memory (LSTM) networks. LSTMs are a special kind of RNNs that can remember and use information over a long sequence of words. This makes them good at understanding long connections in language. Models using LSTMs performed better than the older RNNs, marking a big step forward in NLP.

4.1.3.5. Attention Mechanism

. Around 2014, an idea called the attention mechanism was suggested to make neural networks better at paying attention to important parts of the information. This mechanism lets models decide how important each part of the information is for the current task. This was a crucial improvement for NLP tasks, helping models understand long connections in information better and improve how they understand context.

4.1.3.6. Transformer Architecture

In 2017, a groundbreaking paper introduced the Transformer architecture for NLP, saying "Attention is All You Need." Unlike older models, the Transformer used self-attention instead of recurrent connections. This change allowed the model to work faster during training and understanding. The attention mechanism also helped the model look at the whole input at once, making a big improvement in how it understands language.

4.1.3.7. Emergence of BERT (Bidirectional Encoder Representations from Transformers)

In 2018, Google introduced BERT, a big step forward in NLP. BERT is a pre-trained language model that understands words from both directions, left to right and right to left. This makes it great at many language tasks. BERT learns a lot from its initial training, and then it fine-tunes that knowledge for different jobs.

4.1.3.8. GPT-2 and XLNet

In 2019, OpenAI released GPT-2, a huge language model that did really well on language tasks. It could even generate text that looked like it was written by a person. Another model, XLNet, came out the same year and improved on BERT's idea by considering all possible arrangements of words in a sentence during training.

4.1.3.9. ChatGPT-3

In 2022, ChatGPT-3 is the newest and most advanced machine learning model. It uses special networks to understand and create text that looks a lot like how people talk.

4.1.3.10. Evolving Large Language Models

From 2020 to 2023, NLP research has been about making even bigger language models. Models like GPT-3, with a whopping 175 billion parts, can do lots of things like writing text, translating, answering questions, and even creating code. Although these big models are impressive, people worry about how much computer power they use, how much energy they consume, and the ethical problems they might bring.

4.1.3.11. Continued Research in Transformer-Based Architectures

Researchers have been working to make transformer-based structures better. They created different versions like T5 (Text-to-Text Transfer Transformer), DeBERTa (Decoding-enhanced

BERT with Disentangled Attention), and more to solve specific problems and make them work even better.

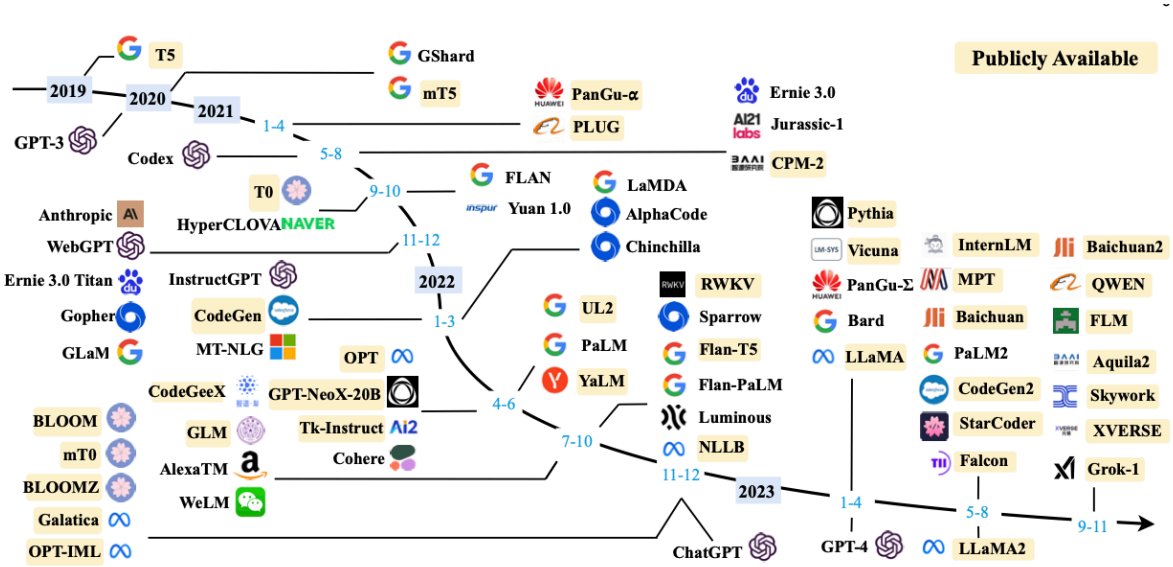


Figure 4. 2. A timeline of existing large language models

This timeline shows when big language models (LLMs) with more than 10 billion parts came out in the past few years. We arranged them based on when their technical papers were released on arXiv or, if there's no paper, when they were first made public. The models with available checkpoints are marked in yellow. The figure only includes models where people have shared how well they work because there's limited space.

In this paper, I will apply BERT model and fine-tuning BERT for the text summarization tasks.

4.1.4. BERT Model Overview [4]

This research by Jacob Devlin and his partners. The introduced BERT, a new language model that learns by looking at both the words to the left and right in sentences. BERT is great because it doesn't need labeled examples; it can learn from any text. You can use BERT for lots of tasks,

like answering questions or figuring out the meaning of sentences. It's really good at these things and has set new records on different tasks, making it one of the best models out there.

4.1.4.1. Overview of Transformer architecture

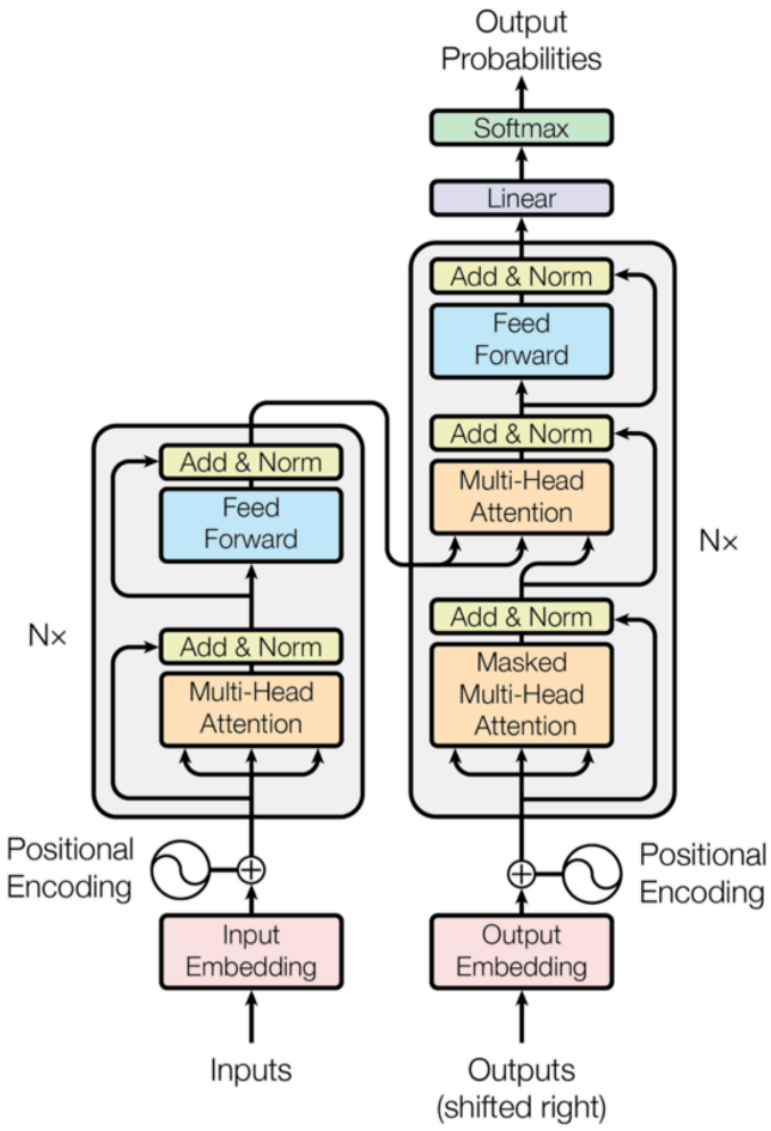


Figure 4. 3. Transformer Architecture

The Transformer is a priority model in Natural Language Processing tasks, especially for understanding language. It came and replaced some older methods like RNN and LSTM for

different jobs. Other models like BERT, GPT, and T5 also use the same idea. This chapter will help us learn more about how the transformer works.

In this section, the basic information is introduced, including explain the basic concepts of Transformer architecture, know how it works by understanding encoder, decoder methods.
They also introduce Transformer, which is shown in figure 4.3.

4.1.4.1.1. Introduction to the Transformer

In their research, they introduce RNN and LSTM for tasks like predicting the next word or translating text. However, these models struggle with remembering long-term connections. To fix this, the Transformer came along, becoming the best for NLP jobs and inspiring models like BERT and GPT-3. Unlike the old models, the Transformer uses attention and gets rid of the need for remembering past things. In language translation, it has two parts: one understands the input, and the other creates the output.

Imagine we want to change a sentence from English to French. We have a special system with two parts: one that understands the English sentence, and another that turns that understanding into a French sentence. It's like a translator for languages.

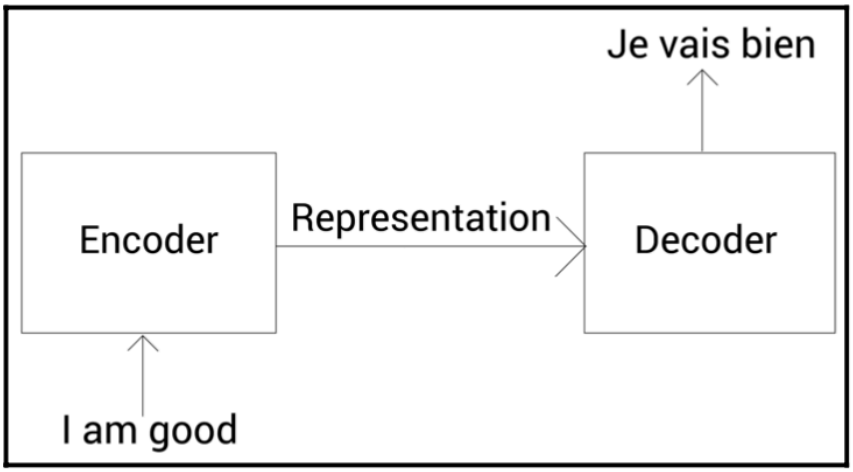


Figure 4. 4. Encoder and Decoder of Transformer

Follow figure 4.4, we can see that the sentence writing in English is an input for the Encoder layer of the Transformer and through the Decoder layer we have one new sentence in French. In the next section, we will know how exactly the Encoder and the Decoder works.

4.1.4.1.2. Understanding the Encoder of the Transformer

The transformer has a bunch of encoders stacked on top of each other (N of them). Each encoder takes what the one below it did and adds more information. The first encoder gets the source sentence, and the last one gives us the representation of that sentence. It's like a teamwork process where each encoder helps understand the sentence better.

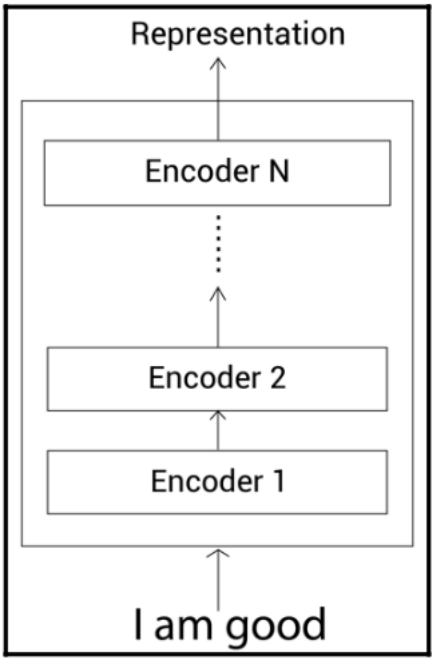


Figure 4. 5. N number of Encoders

Following the Transformer paper "Attention Is All You Need," they used six encoders stacked on each other (N=6). But we can try using a different number, like two (N=2). For simplicity and easier understanding, let's go with N=2.

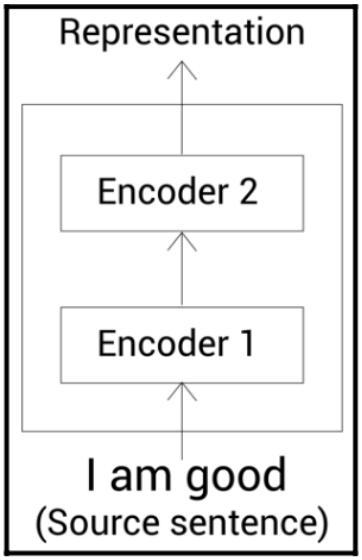


Figure 4. 6. Example with N = 2

Following figure 4.6, we can see that we totally have two encoders block and the next step will show you what is in each block exactly has.

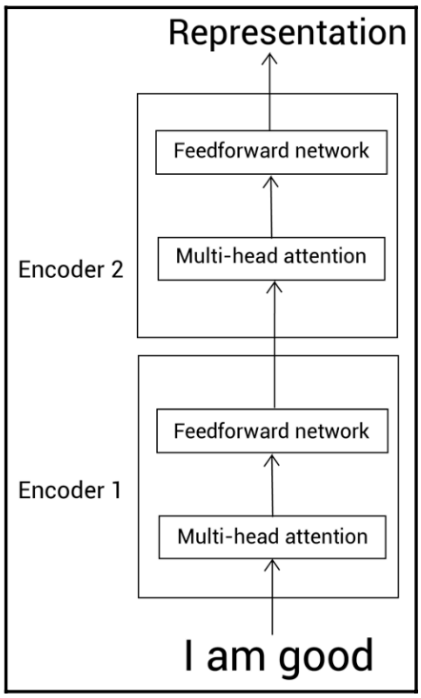


Figure 4. 7. Components of each Encoder block

From figure 4.7, there are two sublayers in each Encoder block: Multi-head attention and Feedforward network. Before understanding clearly about two layers, we need to know about the self-attention mechanism first.

4.1.4.1.3. Self-attention mechanism

The self-attention method can be explained clearly through this sentence:

“A dog ate the food because it was hungry”

When a sentence has a word like "it" that could mean different things, our model uses a self-attention mechanism to figure out the right meaning. It looks at each word in the sentence and how they relate to each other. For instance, when it looks at "it," it checks how it relates to all other words. This helps the model see that "it" refers to "dog" and not "food." They show this by drawing lines, and the line between "it" and "dog" is thicker, showing a stronger connection compared to other lines.

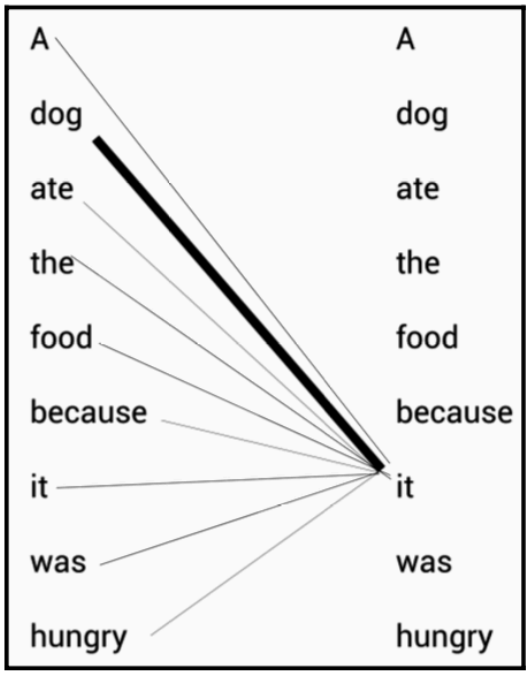


Figure 4. 8. Example of self-attention method

We already know the purpose of how we need to use the self-attention and now, how can we calculate the relative between each word and transform from word into vector? This problem can be explained clearly now. Take a look at the overall method for Encoders, we totally have five main steps, including word embedding, multi-head attention sublayer, feed into the input of the feedforward network, get this result as a input for the next encoder (encoder 2) and encoder 2 will take the same process as the encoder 1.

a) Step 1: Word embedding

So, let continue with the next example. Suppose that we have the sentence "I am good". If x_1, x_2, x_3, are the embedding of "I", "am" and "good" so, we have

Embedding vector of "I" is x_1 = [1.76, 2.22, …, 6.66]

Embedding vector of "am" is x_2 = [7.77, 0.631, …, 5.35]

Embedding vector of "good" is x_3 = [11.44, 10.10, …, 3.33]

Then from this sentence, we have the input matrix:

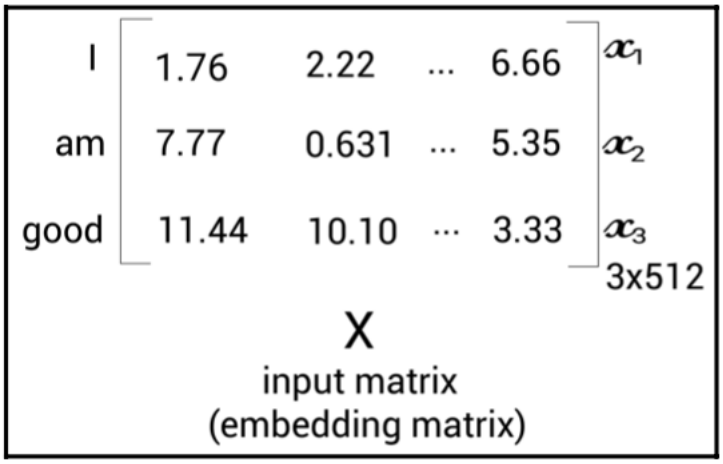


Figure 4. 9. Sentence after being embedded

The input matrix we have indicates that each row shows the special code for a word. In this example, the first row is for "I," the second for "am," and the third for "good." So, if our sentence has 3 words and each code has 512 parts, the input matrix will be [3 x 512].

We make three new matrices from the input one: a query matrix (Q), a key matrix (K), and a value matrix (V). These matrices are important for something called the self-attention mechanism, and we'll learn how they work now. Firstly, we need to create three new weight matrices, W^Q, W^K, W^V and the input matrix is multiplied by three new weight matrices to create Query (Q), Key (K), and Value (V) matrices.

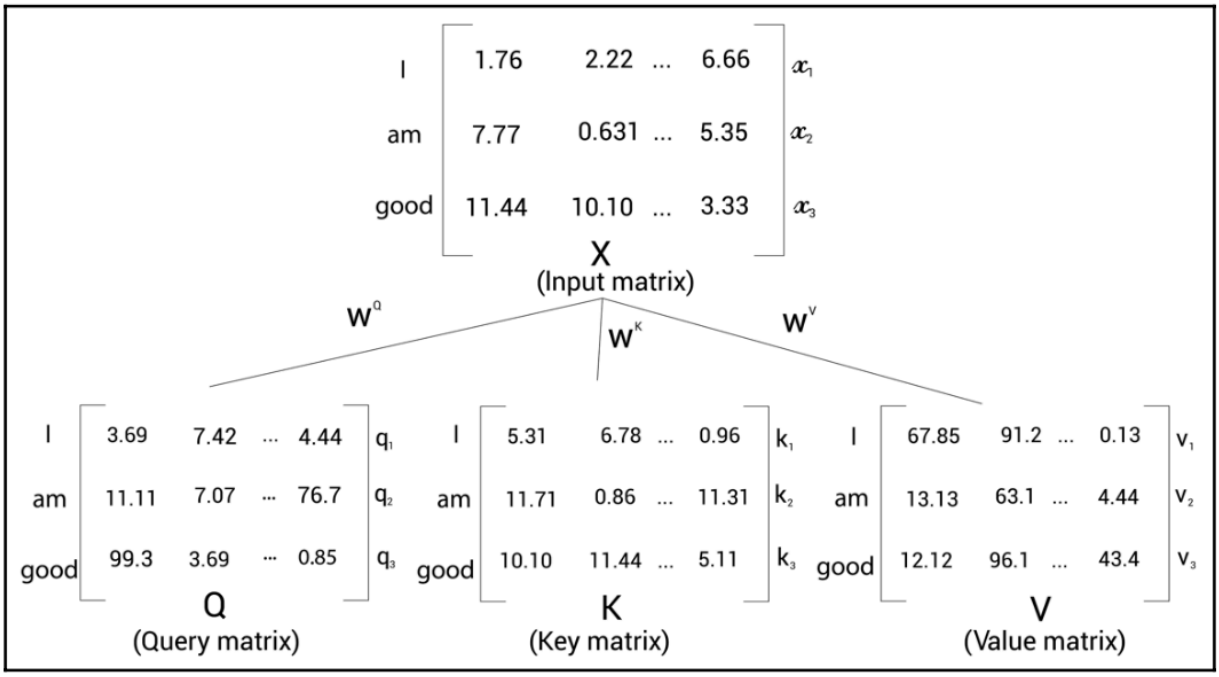


Figure 4. 10. Q, K, V matrices are created

Before the appearing of the Transformer, RNNs understand sentences by looking at one word at a time. Transformers, on the other hand, take all words in a sentence at once, making learning faster. But, here's the catch: without keeping word order, how does the transformer know what the sentence means? Well, it turns out, knowing where each word is in a sentence (word order) is super

important for understanding what the sentence is about. To add the positional encoding, in the "Attention Is All You Need", they suggest two function to solve this problem:

$$P(pos,2i) = \sin\left(\frac{pos}{1000^{\frac{2*i}{d}}}\right) \quad (4.1)$$

$$P(pos,2i+1) = \cos\left(\frac{pos}{1000^{\frac{2*i}{d}}}\right) \quad (4.2)$$

In this equation, pos is the position of the word in this sentence and i is the position of embedding. Follow this example, we have:

$$P = \begin{matrix} \\ I \\ am \\ good \end{matrix} \begin{bmatrix} \sin\left(\frac{pos}{10000^{0}}\right) & \cos\left(\frac{pos}{10000^{0}}\right) & \sin\left(\frac{pos}{10000^{2/4}}\right) & \cos\left(\frac{pos}{10000^{2/4}}\right) \\ \sin\left(\frac{pos}{10000^{0}}\right) & \cos\left(\frac{pos}{10000^{0}}\right) & \sin\left(\frac{pos}{10000^{2/4}}\right) & \cos\left(\frac{pos}{10000^{2/4}}\right) \\ \sin\left(\frac{pos}{10000^{0}}\right) & \cos\left(\frac{pos}{10000^{0}}\right) & \sin\left(\frac{pos}{10000^{2/4}}\right) & \cos\left(\frac{pos}{10000^{2/4}}\right) \end{bmatrix}$$

Figure 4. 11. Positional encoding matrix

Following this formula, we have the positional encoding matrix for our example:

$$P = \begin{matrix} \\ I \\ am \\ good \end{matrix} \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0.841 & 0.540 & 0.009 & 0.999 \\ 0.909 & -0.416 & 0.019 & 0.999 \end{bmatrix}$$

Figure 4. 12. Final positional encoding matrix

After the final positional encoding matrix is calculated, the input matric is a result of the embedding matrix plus the final positional encoding matrix. The next step is relative to the masked multi-head attention layer, which is explained clearly in the next section.

b) Step 2: Multi-head attention layer

In the transformer model, there's a part called the multi-head attention sublayer in each decoder. It gets two inputs: one from the step before (masked multi-head attention) and the other from the encoder representation.

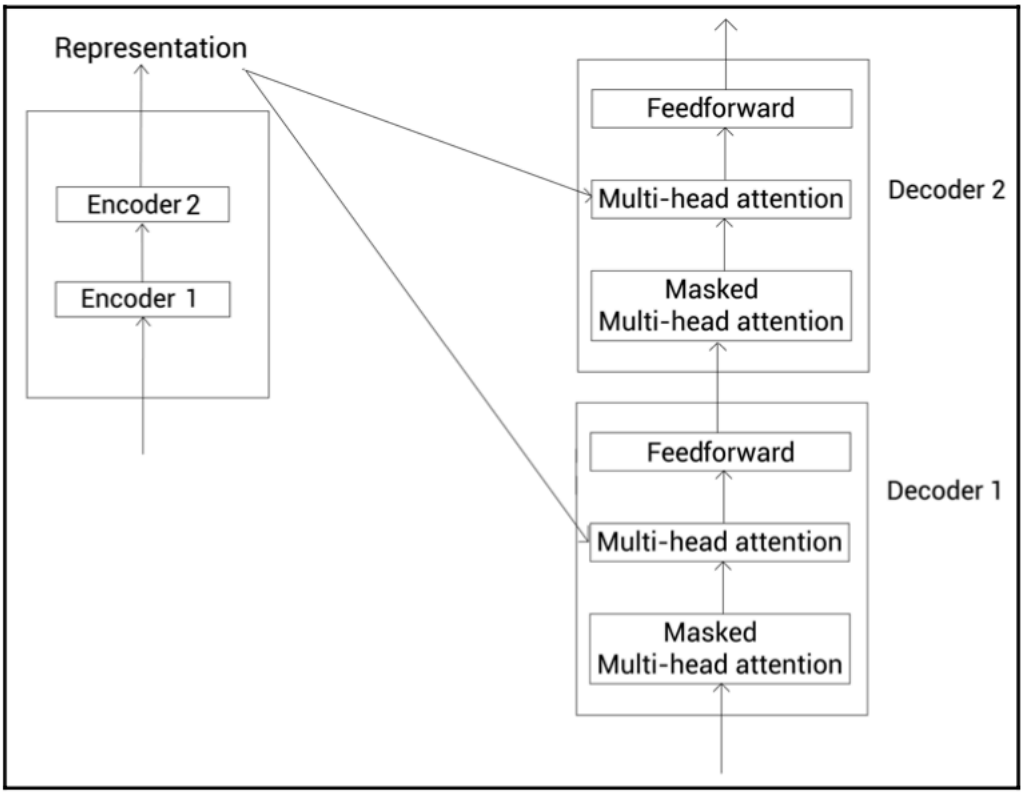


Figure 4. 13. Contact between Encoder and Decoder

This part will explain clearly how the multi-head attention layer works. To kick off the multi-head attention process, we make query, key, and value matrices by multiplying the input matrix with some weights. But, in this special part, we have two inputs: R (from the encoder) and M (from the previous attention step). Now, which one do we pick?

When doing the multi-head attention thing, we make the query matrix Q from the attention matrix M from before, and the key and value matrices come from the encoder representation R. Then, for each "head," we perform certain actions.

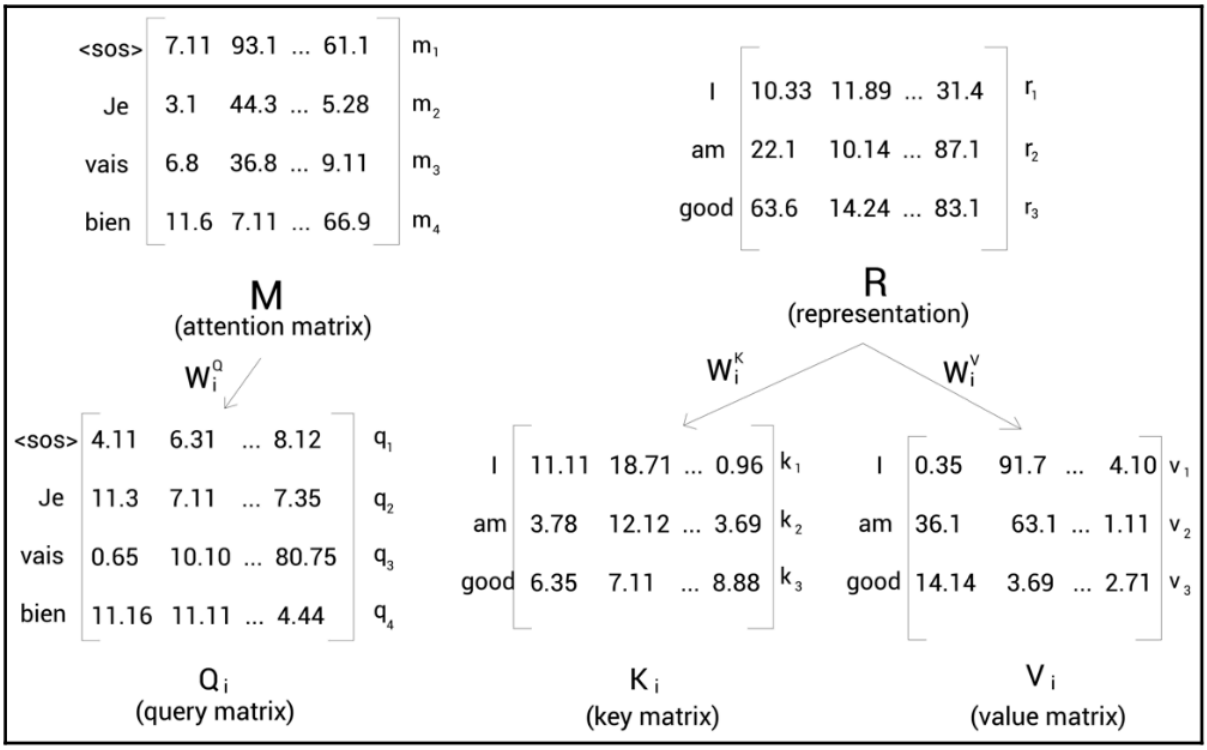


Figure 4. 14. Q, K, V matrices are created

To start self-attention, we multiply the query matrix (representing the target sentence) with the key matrix (representing the input sentence). The values here are just examples to help us understand and are not fixed.

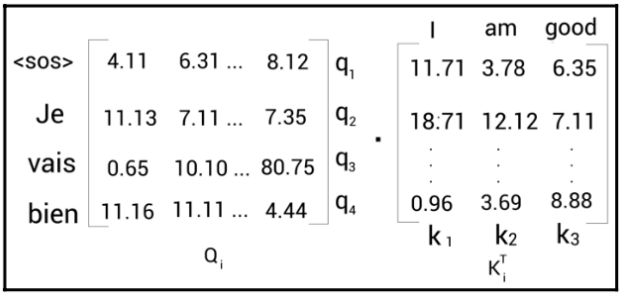


Figure 4. 15. K and Q matrices

The final attention matrix is divided by $\sqrt{dk}$ and then, the final result:

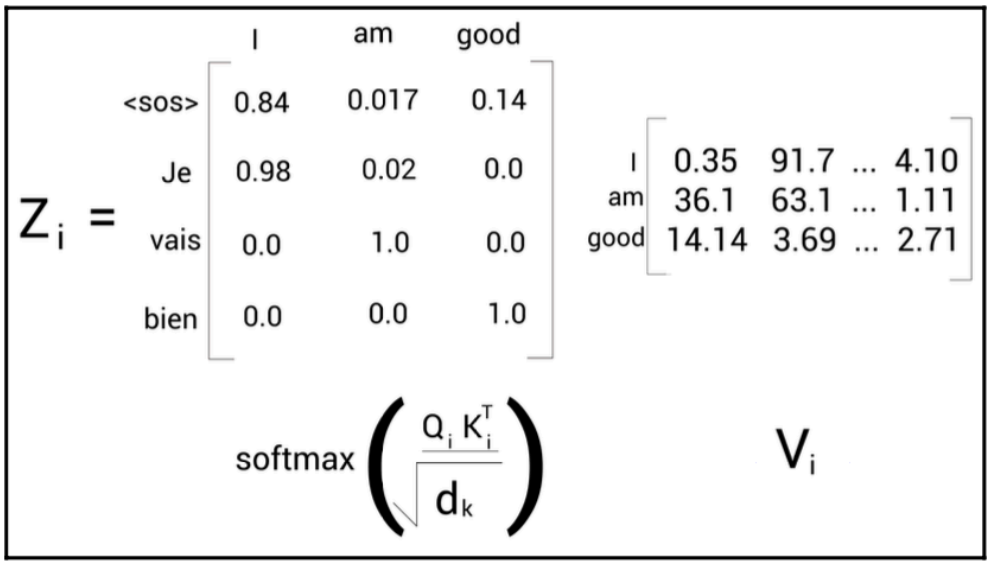


Figure 4. 16. Attention matrix

Finally, the Multi-head attention matrix can be calculated by h number of the attention matrix, then all of them are concatenated and multiply with W^O.

$$Multi - head\ attention = Concatenate(Z_1, Z_2, Z_3, \dots, Z_h) * W_0 \quad (4.3)$$

After calculating the values through the multi-head attention, the final matrix is feed to the input of the feedforward network. Now, we can continue with the concepts of feedforward network in the next step.

c) Step 3: Feedforward network

The decoder's feedforward layer works just like the one in the encoder.

We studied how the transformer model works, focusing on its encoder and decoder parts. We explored various components like attention and feedforward networks, and how they help the model understand words and sentences better. Positional encoding was discussed for keeping track of word order. After understanding the encoder, we delved into the decoder and how it interacts with the encoder. The section concluded with insights into how the transformer is trained. In the

next section, we'll learn about BERT and how it utilizes the transformer for contextual embeddings.

4.1.4.2. Understanding the BERT Model

In this part, we'll start with BERT, a powerful text embedding model that has made a significant impact on NLP tasks. We'll learn what makes BERT unique and understand how it works. The chapter covers the pre-training of BERT through tasks like masked language modeling and next sentence prediction, explaining the process thoroughly. Towards the end, we'll explore interesting subword tokenization methods like byte pair encoding, byte-level byte pair encoding, and WordPiece.

4.1.4.2.1. Basic concepts of BERT

BERT is a top-notch embedding model created by Google, known for its exceptional performance in various NLP tasks. Unlike other models like word2vec, BERT considers context, which is a big reason behind its success.

First, let's look at the difference between context-based and context-free embedding models. Imagine these two sentences: “He got bit by Python.” and “Python is my favorite programming language.”

In these sentences, 'Python' means different things. In the first, it's a snake; in the second, it's a programming language.

Using a context-free model like word2vec would give the same embedding for 'Python' in both sentences. This is because context is ignored, and the meaning is always the same, but BERT is context-based. It understands the context and generates embeddings based on it. For the two sentences, 'Python' would have different embeddings in BERT. How? BERT looks at each word

in a sentence, relates it to all others, and understands the contextual meaning. For example, in sentence A, 'Python' relates to 'bit' and BERT understands it's talking about a snake.

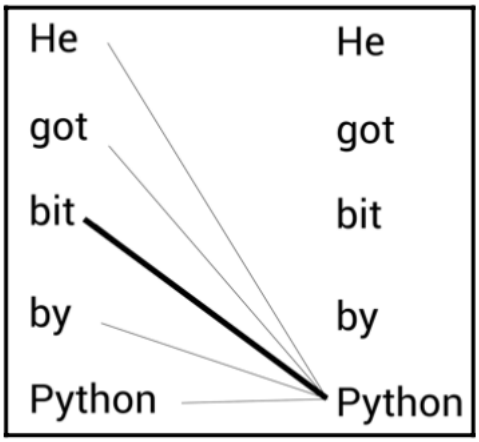


Figure 4. 17. Relation between “Python” and different words

Now, consider sentence B: "Python is my favorite programming language." With BERT, like in the previous example, each word is connected to all others in the sentence to grasp its context. For 'Python' in this sentence, BERT connects it to every word, understanding it refers to a programming language due to the word 'programming’.

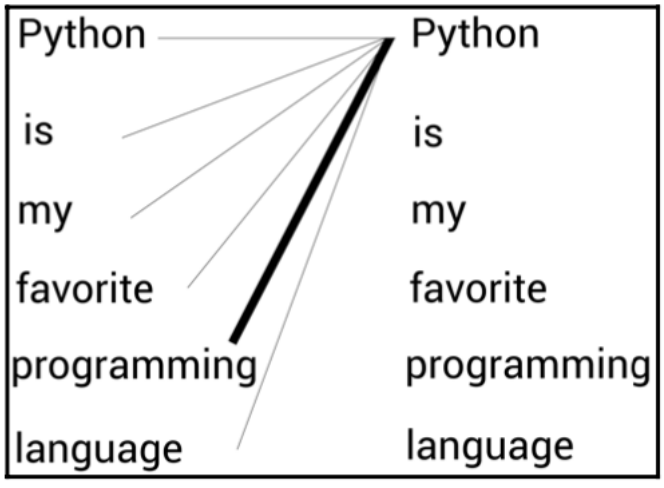


Figure 4. 18. Relation between “Python” and different words

4.1.4.2.2 How the BERT works?

BERT, short for Bidirectional Encoder Representation from Transformer, is built on the transformer model. You can think of BERT as a transformer, but it only includes the encoder part.

The transformer's encoder reads sentences bidirectionally, meaning it understands them in both directions. So, BERT stands for Bidirectional Encoder Representation from the Transformer, essentially highlighting its bidirectional nature.

Let's break down how BERT, the bidirectional encoder representation from the transformer, works with an example. Using the same sentences from the previous section, consider sentence A: 'He got bit by Python'. When we input this sentence into the transformer's encoder, we receive the contextual representation (embedding) for each word as output. The encoder comprehends the context of each word using the multi-head attention mechanism, connecting each word to all others to understand relationships and contextual meanings. The result is the contextual representation of each word in the sentence.

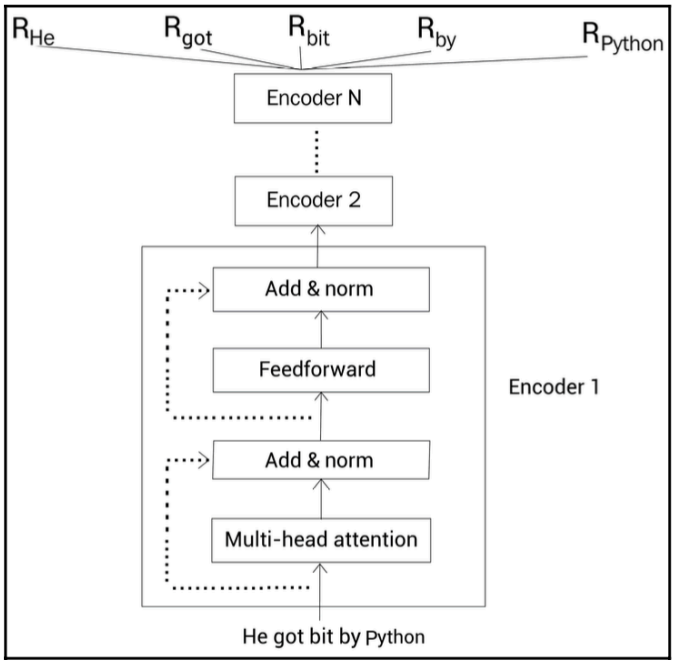


Figure 4. 19. The represent of each word by BERT

Likewise, when we input sentence B, 'Python is my favorite programming language', into the transformer's encoder, the output is the contextual representation of each word in the sentence, depicted in the diagram below:

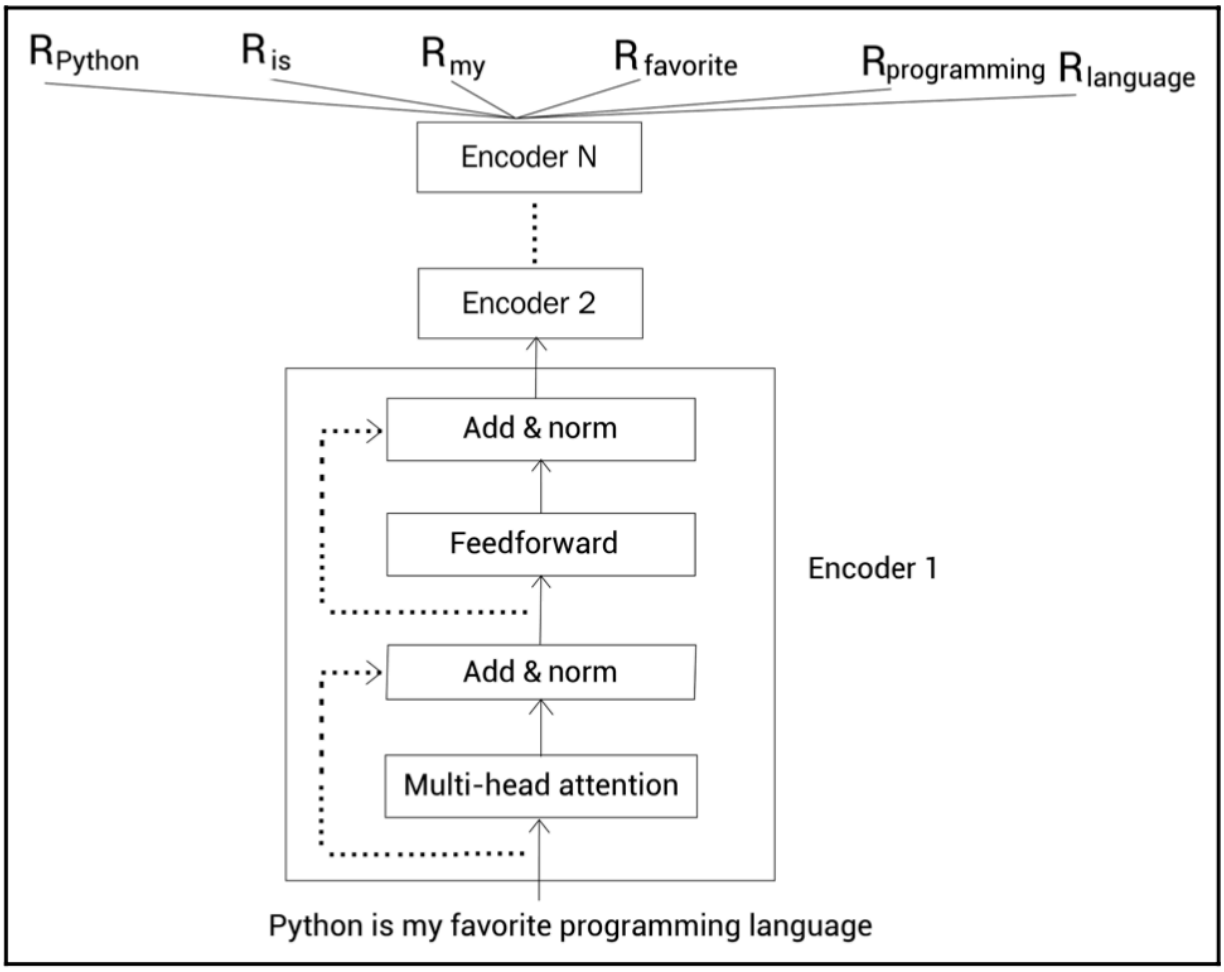


Figure 4. 20. The represent of each word by BERT

Therefore, utilizing the BERT model results in obtaining the contextual representation (embedding) for each word in a given sentence as output. Now that we've grasped how BERT produces contextual representation, the next section will delve into various configurations of BERT.

4.1.4.2.3 BERT Configurations

There are two common version of BERT, include BERT-base and BERT-large

BERT-base

BERT-base is composed of 12 encoder layers, stacked sequentially. Each encoder employs 12 attention heads, and the feedforward network within the encoder is comprised of 768 hidden units. Consequently, the representation size derived from BERT-base is 768. There are three importance parameter in BERT: number of encoder layers (N), Attention head (A), Hidden unit (H). With the BERT-base version, we have L = 12, A = 12, and H = 768 and 110 million parameters.

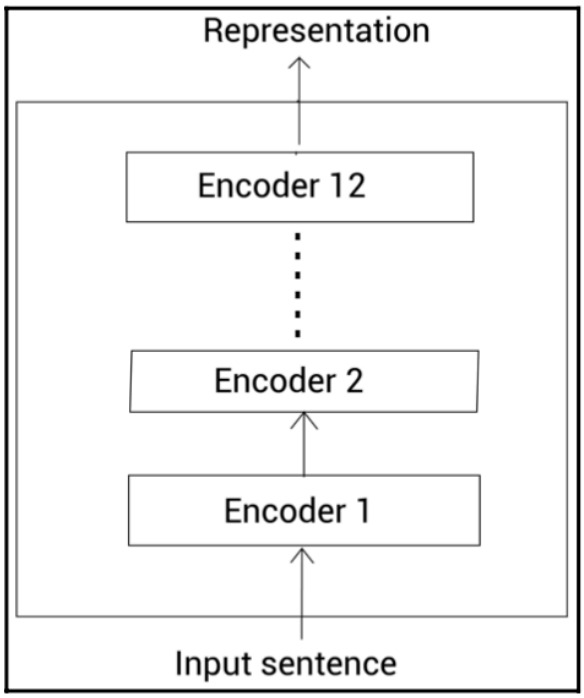


Figure 4. 21. BERT-base version

BERT-large

BERT-large is composed of 23 encoder layers. Each encoder employs 16 attention heads, and the feedforward network within the encoder is comprised of 1024 hidden units. Consequently, the representation size derived from BERT-large is 1024. With the BERT-large version, we have L = 24, A = 16, H = 1024 and 340 million.

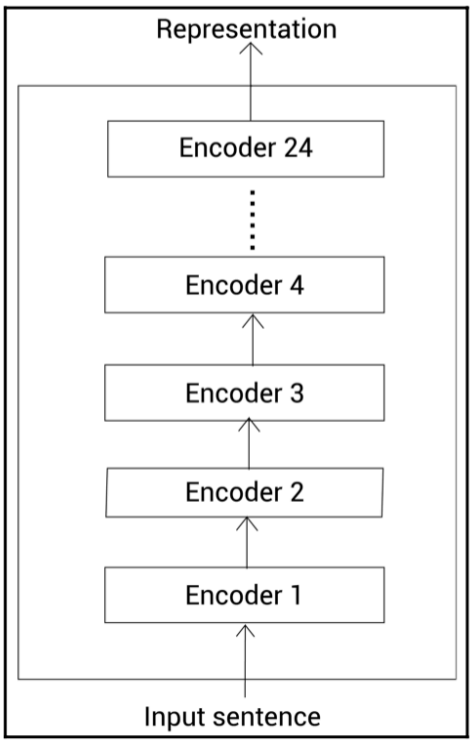


Figure 4. 22. BERT-large version

Moreover, BERT have many different configurations such as: BERT-tiny, BERT-mini, BERT-small, BERT-medium. Each of the version of BERT has a different number of encoder layers and number of hidden units.

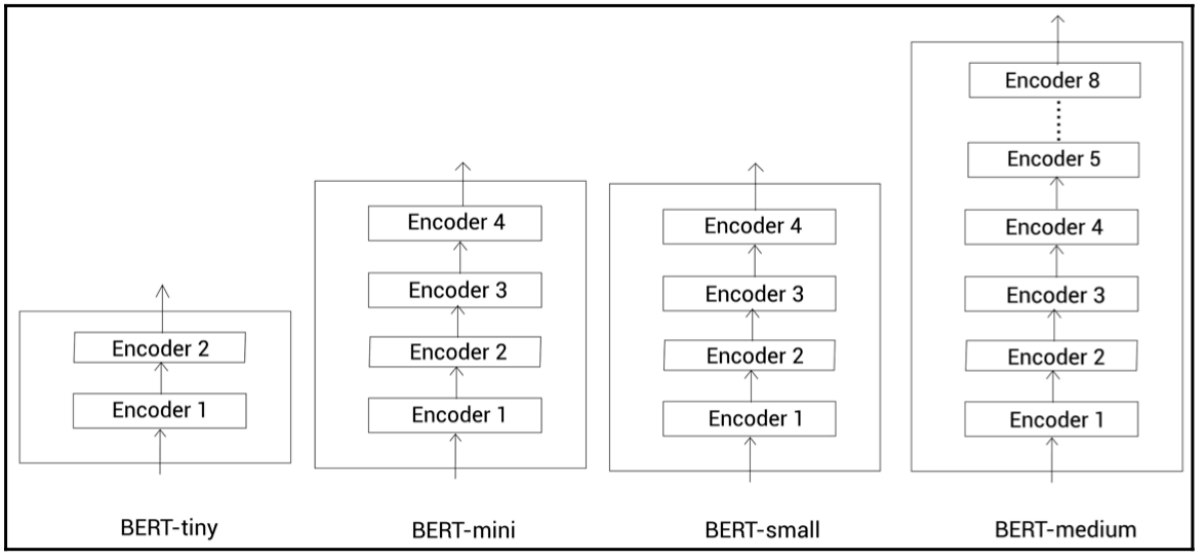


Figure 4. 23. Others version of BERT

Following figure 4.23, In scenarios with limited computational resources, opting for smaller configurations of BERT is feasible. Nevertheless, for more accurate results, the standard configurations like BERT-base and BERT-large are preferred and widely utilized.

In this section, we already know about many differences version of BERT, next is about the BERT Pre-training model, how to create the input for BERT and introduce datasets for training process.

4.1.4.2.4 BERT model

The BERT model undergoes pre-training on an extensive corpus through two intriguing tasks: masked language modeling and next sentence prediction. After pre-training, we preserve the pre-trained BERT model. When faced with a new task, such as question answering, instead of starting the BERT training from scratch, we utilize the pre-trained BERT model. In other words, we employ the pre-trained BERT model and fine-tune its weights for the specific new task. This section will provide an in-depth understanding of how the BERT model undergoes pre-training. Before delving into pre-training details, let's explore how to structure input data to align with BERT's requirements.

a) Input representation

The input matrix of the BERT are calculated by adding the Token embedding, Segment embedding, position embedding. Three layers will be show in detail below:

Token embedding

To understand this layer, let begin with the example by two given sentence A and B

Sentences A: Paris is a beautiful city

Sentences B: I love Paris

Firstly, we need to add sentence A and B into a new sentence like:

Tokens = [Paris, is, a, beautiful, city, I, love, Paris]

Secondly, two new token [CLS] and [SEP] are added to to Tokens at the first and the end of the previous tokens like:

Tokens = [[CLS], Paris, is, a, beautiful, city, [SEP], I, love, Paris, [SEP]]

It's important to note that the [CLS] token is exclusively inserted at the start of the initial sentence, whereas the [SEP] token is appended at the conclusion of each sentence. The [CLS] token serves a purpose in classification tasks, while the [SEP] token signifies the end of each sentence. The specific roles and significance of these two tokens, [CLS] and [SEP], will become clearer as we progress further in this section.

Before presenting all the tokens to BERT, we transform them into embeddings utilizing a dedicated embedding layer known as token embedding. It's crucial to highlight that the values of these token embeddings are subject to learning during the training process. Illustrated in the subsequent diagram, we observe embeddings for each token; for instance, denotes the embedding of the token [CLS], represents the embedding of the token Paris, and so forth:

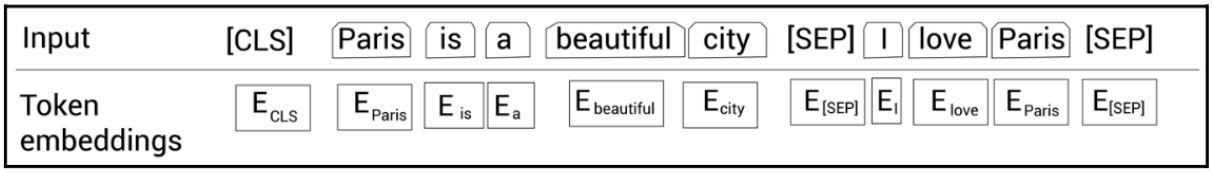


Figure 4. 24. Token embedding matric

Segment embedding

Continue with two previous sentences, after through the token embedding layers, we have:

Tokens = [[CLS], Paris, is, a, beautiful, city, [SEP], I, love, Paris, [SEP]]

Aside from the [SEP] token, we need a mechanism to signal our model about the distinction between the two sentences. For this purpose, we introduce the input tokens to the segment

embedding layer. The segment embedding layer produces one of two embeddings, or, as output. Specifically, if an input token pertains to sentence A, it will be associated with the embedding, while if it belongs to sentence B, it will be linked to the embedding .

Illustrated in the subsequent diagram, tokens originating from sentence A are associated with the embedding, and tokens from sentence B are linked to the embedding :

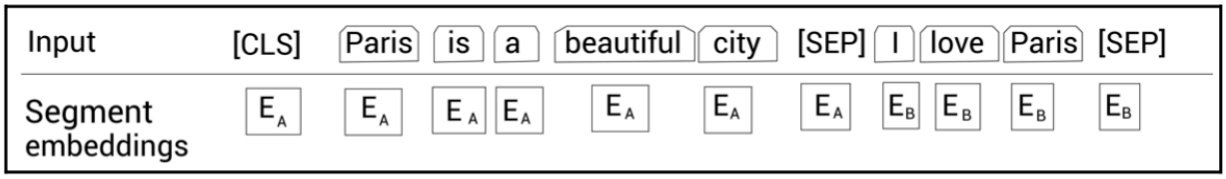


Figure 4. 25. Segment embedding matric

Position embedding

Now, let's delve into the position embedding layer. In our exploration of the transformer model, we discovered that, lacking a recurrence mechanism, it processes all words concurrently. To address this and retain information about the order of words, we introduced positional encoding. BERT, being fundamentally the transformer's encoder, follows the same principle. Thus, before directly inputting tokens into BERT, we employ a layer known as the position embedding layer. This layer assigns a position embedding to each token in our sentence, providing vital information about the position of words for the transformer's effective processing.

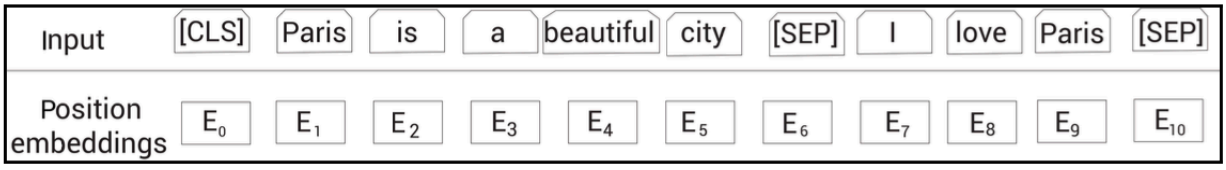


Figure 4. 26. Position embedding matric

b) Final representation

In the final step, we start with the sentences we want to analyze. We turn these sentences into smaller parts called tokens. Then, each token goes through different layers that help the model understand it better. These layers include token embedding, segment embedding, and position embedding. These layers provide information about the meaning of each word, which part of the sentence it belongs to, and its position in the sentence. After going through these layers, we add up all the information and present it to BERT for further analysis.

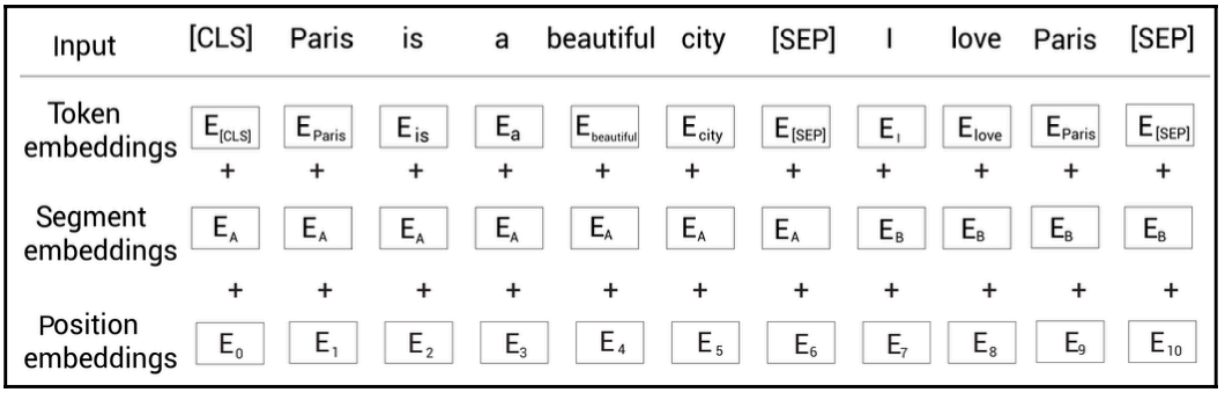


Figure 4. 27. Completely input matric

The figure … show we the final input representation by adding more layer to the original matric, in the next section, we will introduce about WordPiece tokenizer, which is used by BERT.

c) WordPiece tokenizer

BERT uses a unique kind of tokenizer known as a WordPiece tokenizer, which follows the subword tokenization approach. To grasp how the WordPiece tokenizer functions, let's explore an example with the following sentence:

"Let us start pretraining the model."

If we used the WordPiece tokenizer, the output is:

Tokens = [let, us, start, pre, ##train, ##ing, the, model]

When we use the WordPiece tokenizer to tokenize a sentence, it may split words into subwords. The process involves checking if a word is in the vocabulary; if it is, it becomes a token. If not, the word is split into subwords, and the same process is repeated. This strategy effectively handles out-of-vocabulary (OOV) words. BERT's vocabulary consists of 30K tokens, and if a word belongs to these tokens, it is used as a token. If not, the word is split into subwords until reaching individual characters. In our example, the word "pretraining" is not in BERT's vocabulary, so it's split into subwords like "pre," "##train," and "##ing." The hash signs indicate that these are subwords preceded by other words. The subwords are then checked against the vocabulary, and if present, they are used as tokens without further splitting.

Next step, we need to add [CLS] and [SEP] token to the previous tokens:

tokens = [[CLS], let, us, start, pre, ##train, ##ing, the model, [SEP]]

As we covered earlier, we input the tokens to the token, segment, and position embedding layers. After obtaining the embeddings, we sum them up and feed the result to BERT. A more detailed explanation of the WordPiece tokenizer, including how we construct the vocabulary, is discussed towards the end of this chapter in the section on Subword tokenization algorithms. Having understood how to prepare input for BERT through embeddings and how to tokenize using WordPiece, the next section will delve into the process of pre-training the BERT model.

c) Pre-training strategies

There are two main tasks in BERT: Masked language modeling and Next sentence prediction.

Masked language modeling

In the auto-encoding language model known as BERT, the model reads sentences bidirectionally, making predictions for masked words. In the context of a masked language modeling task, approximately 15% of the words in a given input sentence are randomly masked,

and the network is trained to predict these masked words. To achieve this prediction, the model processes the sentence in both forward and backward directions, enhancing its ability to understand contextual relationships in the text.

Now, let have one more example from two sentence “Paris is a beautiful city” and “I love Paris”. After combining and adding two sentences then tokenizing, we have:

Tokens = [Paris, is, a beautiful, city, I, love, Paris]

Next, [CLS] and [SEP] are added from the first and end of each sentence:

Tokens = [[CLS], Paris, is, a beautiful, city, [SEP], I, love, Paris, [SEP]]

The important step in Masked language modeling is we will randomly choose 15% words from entire our vocabulary and then mask them. The words, which are masked are replaced by [MASK] token.

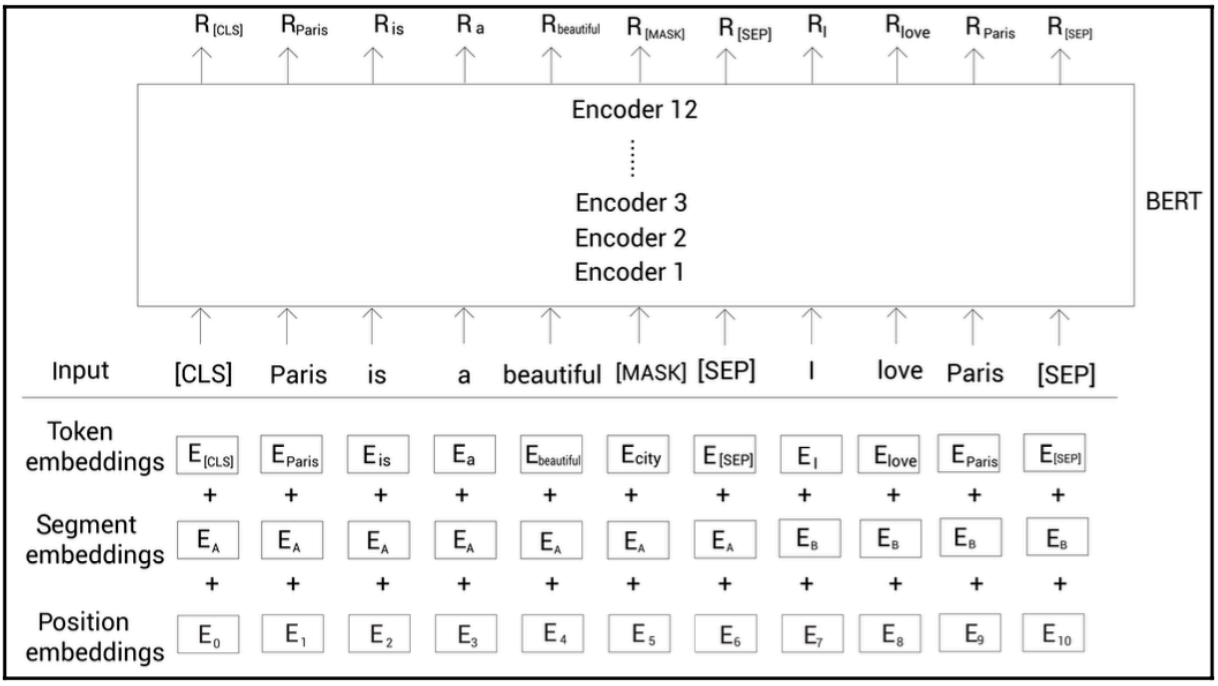


Figure 4. 28. Mask language modeling BERT

After tokenization and masking, we input the tokens to the token, segment, and position embedding layers, resulting in input embeddings. Subsequently, these input embeddings are fed into BERT. As illustrated in the provided diagram, BERT processes the input and produces a representation for each token as output. In this instance, we utilize BERT-base, which comprises 12 encoder layers, 12 attention heads, and 768 hidden units. Consequently, with the BERT-base model, the size of the representation for each token is 768.

Based on the provided diagram, it's evident that we have acquired the representation for each token. The next step involves predicting the masked token using these representations. To achieve this, we input the representation of the masked token, as provided by BERT, into the feedforward network with a softmax activation. The feedforward network processes as input and generates the probability distribution for all words in the vocabulary, determining the likelihood of each word being the masked token. In the subsequent diagram, the input embedding layers (token, segment, and position) are omitted for clarity:

The task involving masked language modeling, commonly referred to as a cloze task, has been elucidated. The procedure for training the BERT model using the masked language modeling task has been comprehensively discussed. Additionally, an alternative approach to masking input tokens, known as whole word masking, is introduced. The details and intricacies of the whole word masking technique will be explored in the upcoming section.

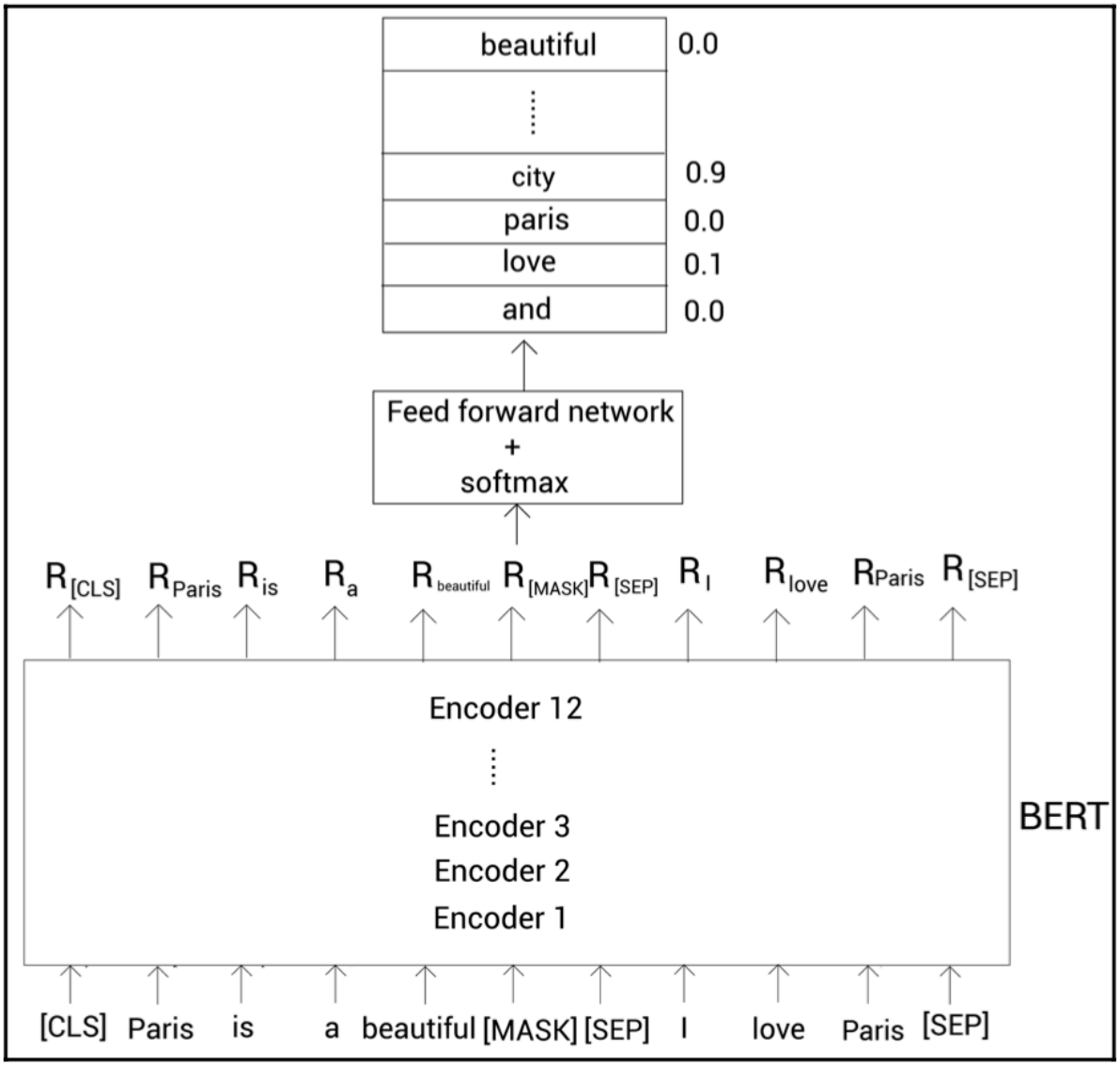


Figure 4. 29. Masked token prediction

Next sentence prediction

The next sentence prediction (NSP) strategy serves as an intriguing approach for training the BERT model. NSP entails a binary classification task, where two sentences are presented to BERT, and the model is tasked with predicting whether the second sentence sequentially follows the first sentence. To delve deeper into the NSP task, let's illustrate it with an example. We have two sentences “She cooked pasta” and “It was delicious”. In the provided sentence pair, sentence B

does not serve as a continuation or follow-up to sentence A. Therefore, we designate this pair with the label "notNext," indicating that sentence B does not sequentially follow sentence A. Within the NSP task, our model aims to predict whether a given sentence pair falls into the "isNext" or "notNext" category. By presenting the sentence pair (comprising sentences A and B) to BERT, we train the model to discern whether sentence B follows sentence A. The model outputs "isNext" if the sequence is continuous and "notNext" otherwise, making NSP fundamentally a binary classification task.

Sentence Pair	Label
She cooked pasta It was delicious	isNext
Jack loves songwriting He wrote a new song	isNext
Birds fly in the sky He was reading	NotNext
Turn the radio on She bought a new hat	NotNext

Figure 4. 30. Sample dataset

The NSP task serves the purpose of enabling our model to comprehend the relationship between two given sentences. This understanding proves valuable in various downstream tasks, such as question answering and text generation. To create a dataset for the NSP task, we can generate it from any monolingual corpus. Assuming we have several documents, for the "isNext"

class, we pair two consecutive sentences from one document and label them accordingly. For the "notNext" class, we pair a sentence from one document with another from a random document and assign the "notNext" label. It is crucial to maintain a balance, with 50% of data points labeled as "isNext" and 50% as "notNext." Once the dataset is prepared, we can proceed to train the BERT model for NSP tasks. Suppose our dataset structure is represented in the figure 4.30.

We have Tokens = [She, cooked, pasta, It, was, delicious] and after add [CLS] and [SEP], we have:

Tokens = [[CLS], She, cooked, pasta, [SEP], It, was, delicious, [SEP]]

For classification purposes, we utilize the representation of the [CLS] token and input it into the feedforward network with a softmax function. This process yields the probability of our sentence pair being classified as either isNext or notNext. The decision to exclusively consider the [CLS] token's embedding arises from its role as an aggregate representation of all tokens, effectively encapsulating the essence of our sentences. Consequently, we can overlook the embeddings of other tokens and focus on the [CLS] token's representation, simplifying the classification process. The diagram binfigure 4.31 illustrates this, with the omission of the input embedding layers (token, segment, and position embedding layers) for clarity.

Figure 4.31 illustrates that the feedforward network outputs a high probability for our input sentence pair belonging to the isNext class. It's important to note that in the early iterations, the model may not provide accurate probabilities due to suboptimal weights in the feedforward network and BERT's encoder layers. However, through multiple iterations and the application of backpropagation, these weights are continually updated, allowing the model to learn optimal weights and improve performance. This iterative process forms the basis of training our BERT model with the NSP task. We've gained insights into pre-training BERT through both masked

language modeling and NSP tasks. The subsequent section will delve into the intricacies of the pre-training procedure.

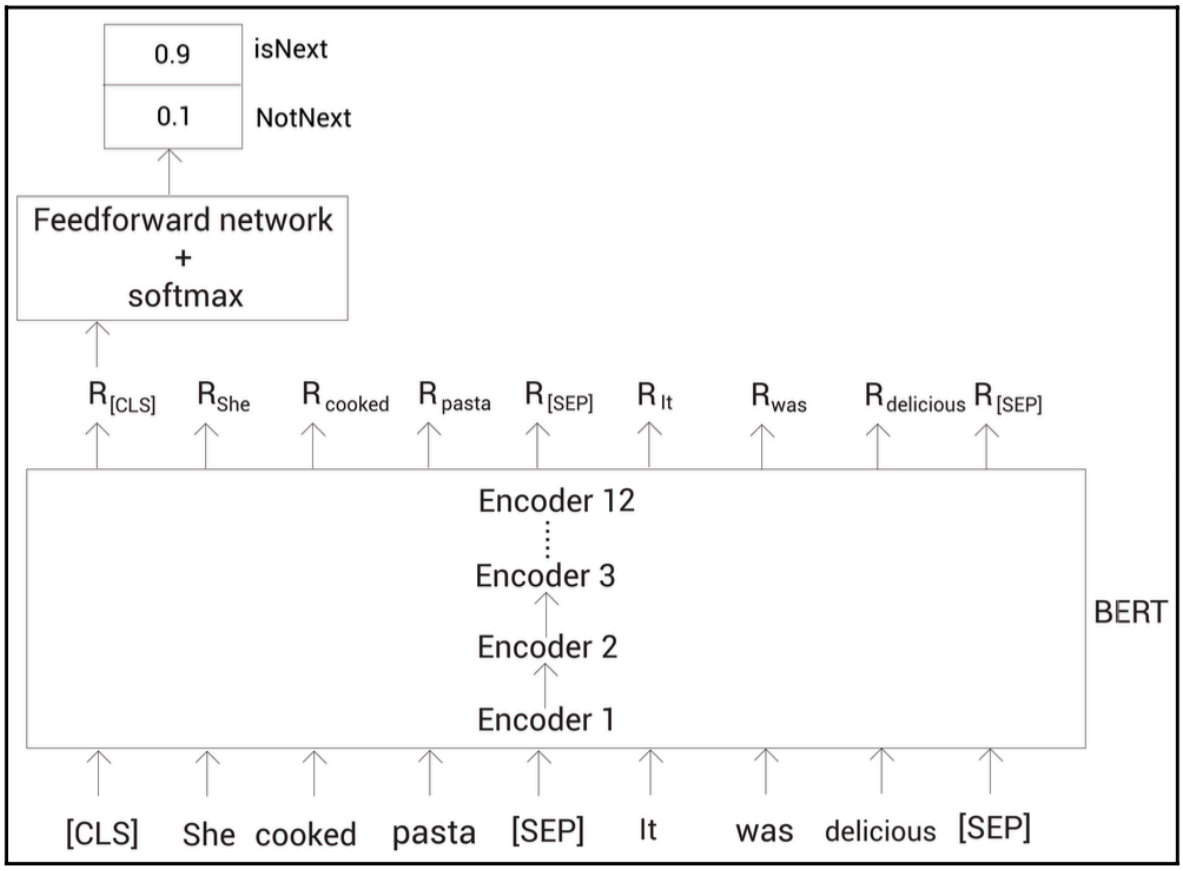


Figure 4. 31. NSP task

This section commenced by delving into the fundamental concept of BERT. It was emphasized that BERT possesses the ability to comprehend the contextual meanings of words and generate embeddings based on context, a departure from context-free models like word2vec that generate embeddings regardless of context.

Subsequently, the inner workings of BERT were explored. It was elucidated that Bidirectional Encoder Representation from Transformer (BERT), as the name suggests, essentially corresponds to the transformer model.

Progressing further, attention was directed towards various configurations of BERT. It was outlined that BERT-base comprises 12 encoder layers, 12 attention heads, and 768 hidden units, while BERT-large encompasses 24 encoder layers, 16 attention heads, and 1,024 hidden units.

Continuing the exploration, insights were gained into how the BERT model undergoes pre-training through two compelling tasks – masked language modeling and Next Sentence Prediction (NSP). The mechanism was unfolded, revealing that in masked language modeling, 15% of tokens are masked, and BERT is trained to predict these masked tokens. Simultaneously, in the NSP task, BERT is trained to discern whether the second sentence is a sequential continuation of the first sentence.

4.1.5. A Survey on Datasets for Text Summarization [5]

This research by Nallapati and his partners. This paper introduces trending methodologies for text summarization, and existing datasets, metrics, and models proposed for the text summarization.

4.1.5.1. English Dataset

4.1.5.1.1. CNN/Daily Mail Dataset

The CNN/Daily Mail Dataset is a cornerstone in English text summarization research. It encompasses a diverse collection of news articles paired with abstractive summaries. The dataset is particularly valuable for its representation of journalistic writing styles and real-world news reporting. Each article in the dataset comes with multiple highlights, providing a range of perspectives for model training and evaluation. Researchers and developers widely use this dataset to benchmark and assess the performance of various summarization models due to its extensive coverage of current events and diverse language patterns.

4.1.5.1.2. DUC (Document Understanding Conference) Datasets:

The DUC datasets play a pivotal role in advancing text summarization technologies. These datasets, spanning multiple years of the Document Understanding Conference, consist of document sets tailored for summarization tasks. The documents cover a spectrum of topics, offering challenges in handling diverse domains and document structures. DUC datasets are crucial for evaluating the generalizability and adaptability of summarization models across different genres and subject matters. As a result, they serve as a benchmark for the robustness and versatility of text summarization algorithms.

4.1.5.1.3. Newsroom Dataset:

The Newsroom Dataset stands out for its vast collection of news articles and abstractive summaries, providing a rich resource for training and evaluating summarization models. It includes a wide array of news sources and topics, capturing the nuances of journalistic writing. This dataset is especially beneficial for researchers aiming to enhance the abstractive summarization capabilities of models. By leveraging the Newsroom Dataset, developers can address challenges related to diversity in writing styles, topics, and summarization requirements commonly encountered in real-world news content.

4.1.5.2. Vietnamese Dataset

4.1.5.2.1. VNDS (Vietnamese Newspaper Dataset for Summarization)

The VNDS is a pivotal dataset for Vietnamese text summarization research. Comprising articles from various Vietnamese online newspapers, each annotated with human-generated summaries, the VNDS is tailored to the linguistic and contextual nuances of the Vietnamese language. This dataset offers a unique opportunity to train and evaluate summarization models on authentic Vietnamese news content, facilitating advancements in the understanding and generation of concise and meaningful summaries in the Vietnamese language.

4.1.5.2.2. VNTC (Vietnamese News Text Classification)

Originally designed for text classification, the VNTC dataset can be adapted for text summarization tasks in Vietnamese. It contains a diverse collection of Vietnamese news articles, making it a valuable resource for researchers exploring summarization models in the context of different genres and subject matter specific to the Vietnamese language. While primarily used for classification, the dataset provides a foundation for developing summarization capabilities for Vietnamese text.

4.1.5.2.3. UIT-VSFC (UIT Vietnamese Summarization and Face Classification)

The UIT-VSFC dataset is dedicated to Vietnamese text summarization and face classification. Focused specifically on the summarization task, this dataset offers a curated collection of articles and summaries in Vietnamese. Developed by the University of Information Technology, Vietnam, UIT-VSFC provides researchers with a targeted resource to enhance the performance of summarization models in the Vietnamese language, addressing the challenges unique to summarizing content in this linguistic context.

In conclusion, the availability of these diverse and specialized datasets in both English and Vietnamese plays a crucial role in advancing the capabilities of text summarization models, catering to the linguistic nuances and content characteristics of each language. Researchers and developers can leverage these datasets to train, fine-tune, and evaluate summarization algorithms, ultimately contributing to the broader goal of improving the efficiency and effectiveness of automated summarization across languages.

CHAPTER V

METHODOLOGY

In this chapter, the methodology is described in detail for each of the objectives. The input of our system is the question or the paragraph relative to the healthcare problems by the user, and the output is an answer, or a summarized paragraph depending on the user's question. In this project, we research the BERT architecture, preprocess VNDS, and change the values of the BERT models to get a higher result in evaluation metrics. All of these steps are developed by Python codes. There are four main steps, shown in see figure 5.1, to accomplish our objectives:

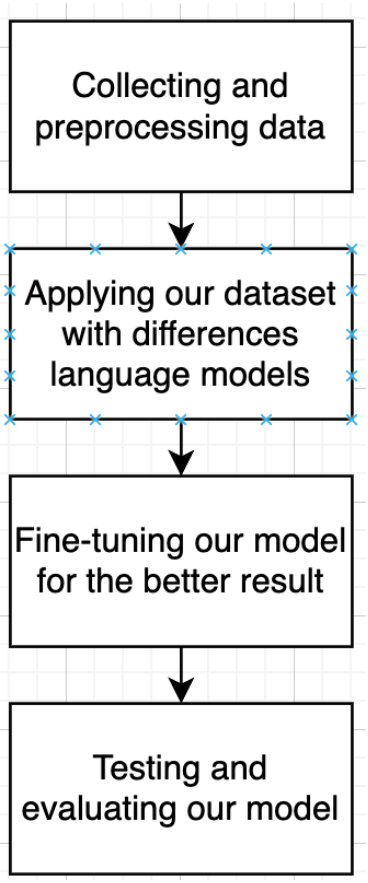


Figure 5. 1. Flowchart of our project

In the first step (section 5.1), we will need to preprocess the Vietnamese Dataset for Summarization (VNDS), FAQ (Frequently Asked Questions).

In the second step (section 5.2), after preprocessing data, we have to test our dataset to the BERSUM and PhoBERT.

In the third step (section 5.3), after having an evaluation – of the BERT and VNDS dataset, we need to fine-tune BERTSUM, PhoBERT model by changing the learning rate, batch size, and number of epochs for a better result.

In the fourth step (section 5.4), after trying our dataset with BERTSUM and PhoBERT, the result need to be evaluated and tested by ROUGE, F1-Score.

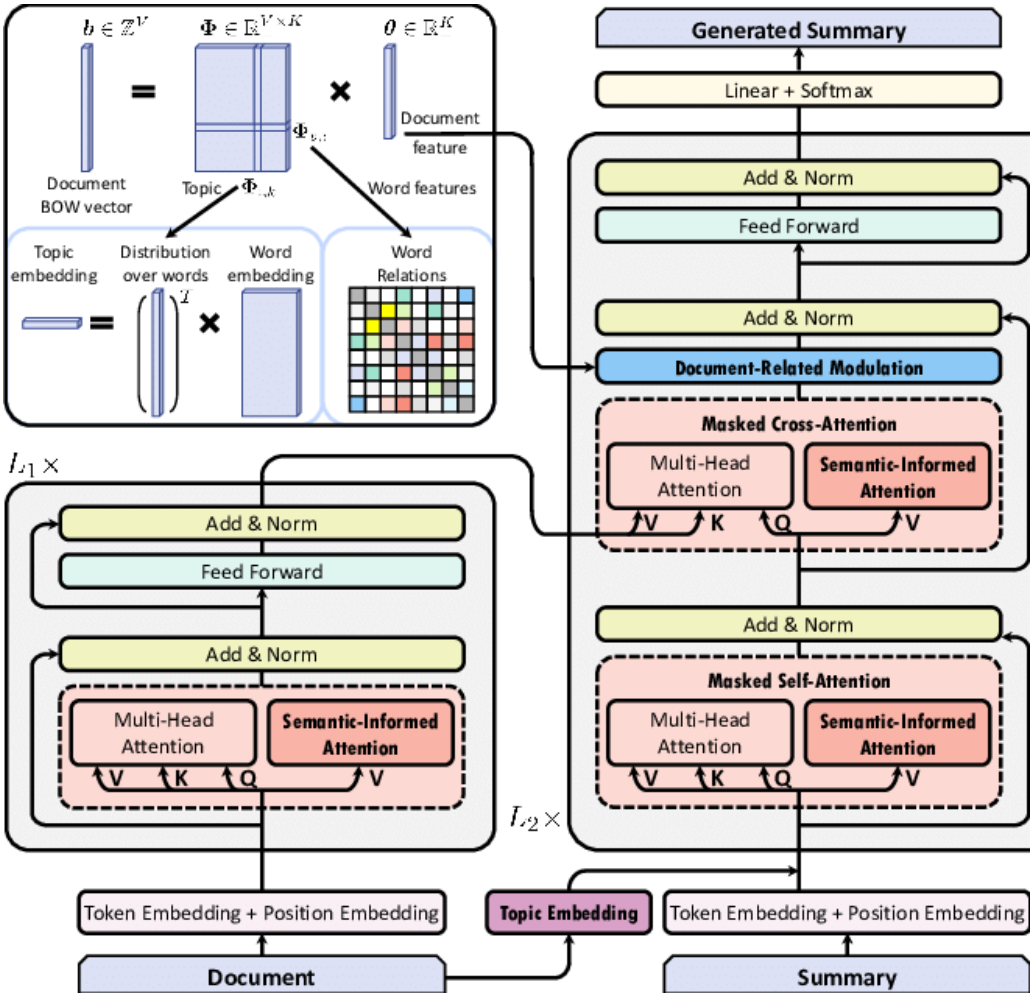


Figure 5. 2. The architecture of our Vietnamese Documents summarization project

5.1. Preprocessing data

5.1.1. Create a virtual environment and install the required library

Firstly, we need to create a virtual environment for our project can run independently from another project. After that, we need to install all packages used including transformers, torch, nltk, datasets, Pillow, scikit-learn, and pandas.

Transformer: Transformers is a robust library developed by Hugging Face that offers advanced pre-trained models for natural language processing tasks. It includes transformer architectures like BERT, GPT, RoBERTa, and more. The transformers package allows you to make use of pre-trained transformer models for tasks such as text classification, named entity recognition, question answering, and more.

Torch: Torch is a widely used open-source machine learning library primarily designed for deep learning. It provides a variety of tools and functionalities for building and training neural networks. The torch package serves as the foundational framework for training and running deep learning models. It includes modules for tensor operations, optimization algorithms, and utilities for handling data.

NLTK: NLTK is a comprehensive Python library for natural language processing (NLP). It offers a range of tools, resources, and datasets for various NLP tasks. NLTK package provides functionalities for tasks such as tokenization, stemming, part-of-speech tagging, syntactic parsing, sentiment analysis, and more. It is commonly used for NLP research and development.

Datasets: Datasets is a library developed by Hugging Face that gives access to a diverse collection of datasets for machine learning and natural language processing tasks. Datasets package simplifies the process of accessing and manipulating popular datasets by providing a

unified interface. It contains a wide variety of datasets that can be easily loaded and utilized in your projects.

Pillow: Pillow is a powerful Python library for image processing. It offers numerous functionalities for tasks such as reading and writing various image formats, transforming images, and more. Pillow package enables you to work with images in your projects. It is commonly used for tasks like image preprocessing, augmentation, resizing, cropping, and applying filters.

Scikit-learn: Scikit-learn, also known as sklearn, is a popular Python library for machine learning. It provides a comprehensive set of tools and algorithms for various AI tasks. Scikit-learn package offers functionalities for tasks such as data preprocessing, feature extraction, model selection, classification, regression, clustering, and evaluation. It is widely used in machine learning projects.

Pandas: Pandas is a powerful Python library for data manipulation and analysis. It provides data structures and functions for efficiently working with structured data, such as tables and time series. Pandas package is extensively used for tasks such as data cleaning, transformation, filtering, aggregation, and visualization. It simplifies data handling and analysis, making it a popular choice for data-related tasks.

5.1.2. Preprocessing data for BERTSUM model [8]

5.1.2.1. Download stories

For preprocessing any dataset for BERT model in Text summarization, we need to crawl your dataset from any website, in my thesis project VNNewsCrawler tools was used for crawling dataset.

Unfortunately, the file was crawled from the VNNewsCrawler tools has the “.txt” extension but the input file of BERT model need “.story” extension so we need to convert from “.txt” file to

".story" file. Creating a .story file involves organizing your text data into a specific format, which typically includes the main content of the article and highlights. Here is a simple step-by-step guide I used to convert regular text data to a ".story" file:

Step 1: Install Required Libraries

```
pip install nltk
```

Step 2: Create a Python Script

Write a Python script to read your regular text data and organize it into the .story format. Below is a basic example using NLTK for sentence tokenization:

```
import nltk
from nltk.tokenize import sent_tokenize


def create_story_file(text_path, story_path):
    with open(text_path, 'r', encoding='utf-8') as text_file:
        text_content = text_file.read()


    # Use NLTK to tokenize the text into sentences
    sentences = sent_tokenize(text_content)


    # Extract the first two sentences as highlights
    highlights = sentences[:2]


    # The rest of the sentences are considered the main content
    main_content = sentences[2:]
```

```
    # Write the .story file
    with open(story_path, 'w', encoding='utf-8') as story_file:
        # Write highlights
        for highlight in highlights:
            story_file.write(f"@highlight\n{highlight}\n")

        # Write main content
        for sentence in main_content:
            story_file.write(f"{sentence}\n")

if __name__ == "__main__":
    # Replace 'input.txt' and 'output.story' with your file paths
    create_story_file('input.txt', 'output.story')
```

Step 3: Create a Python Script

```
python convert_to_story.py
```

5.1.2.2. Download VNCoreNLP

We need to download VNCoreNLP to tokenize data

```
export CLASSPATH=/path/to/vn-corenlp-full-2017-06-09/vn-corenlp-3.8.0.jar
```

5.1.2.3. Sentence Splitting and Tokenization

```
python preprocess.py -mode tokenize -raw_path RAW_PATH -save_path TOKENIZED_PATH
```

5.1.2.4. Format to PyTorch Files

```
python preprocess.py -mode format_to_bert -raw_path JSON_PATH -save_path BERT_DATA_PATH -oracle_mode greedy -n_cpus 4 -log_file ../logs/preprocess.log
```

5.1.3. Preprocessing Vietnews dataset (VNDS)

For our research, we preprocess a VNDS dataset to ensure its relevance and quality. Initially, we focused on articles spanning the years 2016 to 2019 from prominent sources, namely tuoitre.vn, vnexpress.net, and nguoiduatin.vn. These sources were chosen for their regular updates on noteworthy events in Vietnam. To streamline the dataset, we employed a TF-IDF and SVM-based classifier to filter out articles containing questionnaires, admissions, analytical comments, and weather forecasts. These types of content, while present in the original documents, were deemed less pertinent to document summarization. Consequently, the refined dataset exclusively comprises processed news events. Furthermore, to enhance content diversity, we retained only those documents with a minimum of five sentences, excluding shorter ones. Following the data collection phase, we applied NLTK for sentence segmentation and the vitk tool for word segmentation. Lastly, the dataset was partitioned into three sets with the distribution of 70% for training (105,418), 15% for development (22,642), and 15% for testing (22,644).

	Train Set	Val set	Test set
number of documents	105,418	22,642	22,644
#avg number of sentences in abstract	1.22	1.22	1.23
#avg number of words in abstract	28.48	28.54	28.59
#avg number of sentences in body	17.72	17.81	17.72
#avg number of words in body	418.37	419.66	418.74

Figure 5. 3. Statics of preprocessed dataset

5.1.4. Preprocessing FAQ dataset

Text summarization involves the process of condensing a lengthy document while preserving clarity, conciseness, and essential information. In the context of Frequently Asked Questions (FAQ) summarization, the summarizer not only chooses a meaningful subset of words but also adjusts certain language nuances, correcting errors and simplifying non-technical terms found in the input questions. This dual approach serves to alleviate the burden on healthcare professionals by streamlining the categorization of numerous questions and summarizing crucial information. Our contribution includes the creation of a sentence-level FAQ summarization dataset, a semi-manually labeled resource that also serves as a benchmark for evaluating the generative capabilities of models in this domain. Initially, we retrieve articles from reputable healthcare sources in the FAQ section, such as Vinmec and Mevabe6. Employing the titles as summaries and the question text as input sequences, we ensure that the user can easily grasp the content of the question. The segmentation of crawled news articles is accomplished using RDRSegmenter (Nguyen et al., 2018) from VNCoreNLP (Vu et al., 2018). Given that the question titles are human-written, they inherently fulfill the criteria for abstractive summarization. Maintaining the original training set, we normalize the gold label in the development and test sets. The normalization process involves addressing bias in the gold label by randomly selecting 1,500 samples and distributing them among five annotators who follow our guidelines. The aim is to refine spelling errors, rectify missing punctuations, and clarify ambiguous summaries, resulting in a more explicit version (e.g., transforming "gia´ 5 in 1" to "gia´ vaxcin 5 in 1" or "price 5 in 1" to "price of vaccine 5 in 1"). This process retains the natural cohesion of the summary while ensuring a concise length of 1-2 sentences with fewer than 15 words. The dataset, comprising a total of 13,277 samples, 80% is used for training, 10% for dev and 10% for testing.

5.2. Apply our dataset for BERTSUM, PhoNLP

We will try to use two different pre-train models BERTSUM and PhoNLP for Text Summarization.

BERTSUM: employs BERT as an encoder to capture bidirectional contextual information from input text. This enables the model to understand the intricate relationships between words and sentences, fostering a more comprehensive understanding of the document. The transformer architecture, intrinsic to BERT, plays a pivotal role in BERTSUM. By utilizing multi-head self-attention mechanisms, BERTSUM can efficiently process and weigh different parts of the input document, capturing salient features for summarization. BERTSUM predominantly operates as an extractive summarization model, where it selects the most important sentences from the input document to form the summary. The model determines sentence importance based on contextual embeddings and attention mechanisms. To adapt BERT for summarization tasks, BERTSUM incorporates a fine-tuning stage. During this process, the model learns to prioritize sentences based on their significance in representing the document's core information. In the realm of text summarization, BERTSUM stands out as a cutting-edge model that harnesses the capabilities of BERT and transformer architecture. Its ability to handle long documents and produce contextually rich summaries positions it as a noteworthy advancement in the field.

To employ BERT for extractive summarization, it is necessary for it to generate representations for each sentence. Nonetheless, due to BERT's training as a masked-language model, the resulting vectors are associated with tokens rather than entire sentences. Moreover, although BERT incorporates segmentation embeddings to signify distinct sentences, it utilizes only two labels (sentence A or sentence B), which poses a limitation in scenarios with multiple sentences, as seen

in extractive summarization. To overcome this, adjustments are made to the input sequence and embeddings of BERT, facilitating the extraction of summaries.

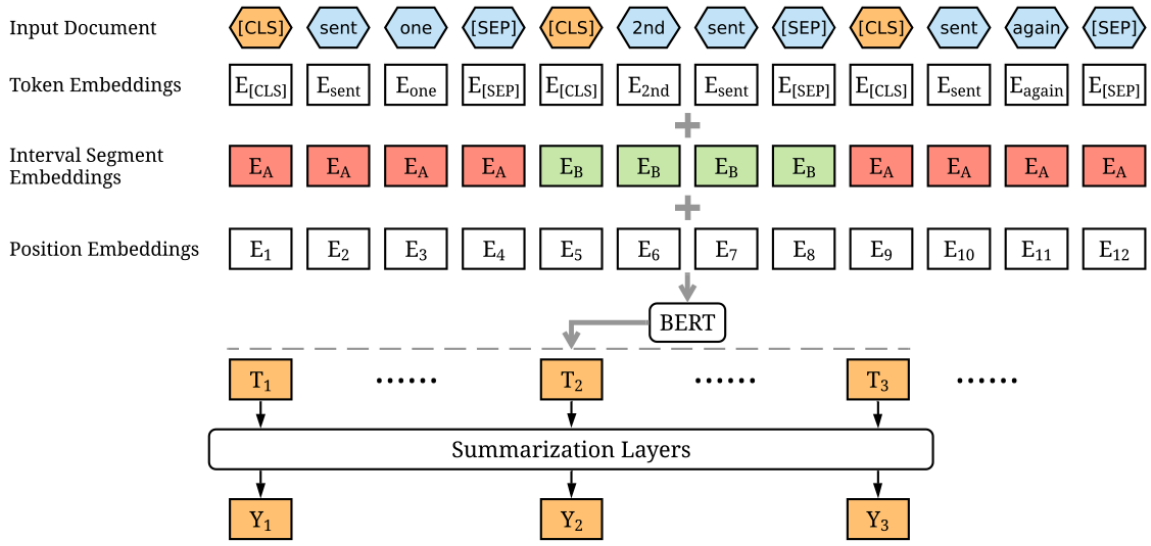
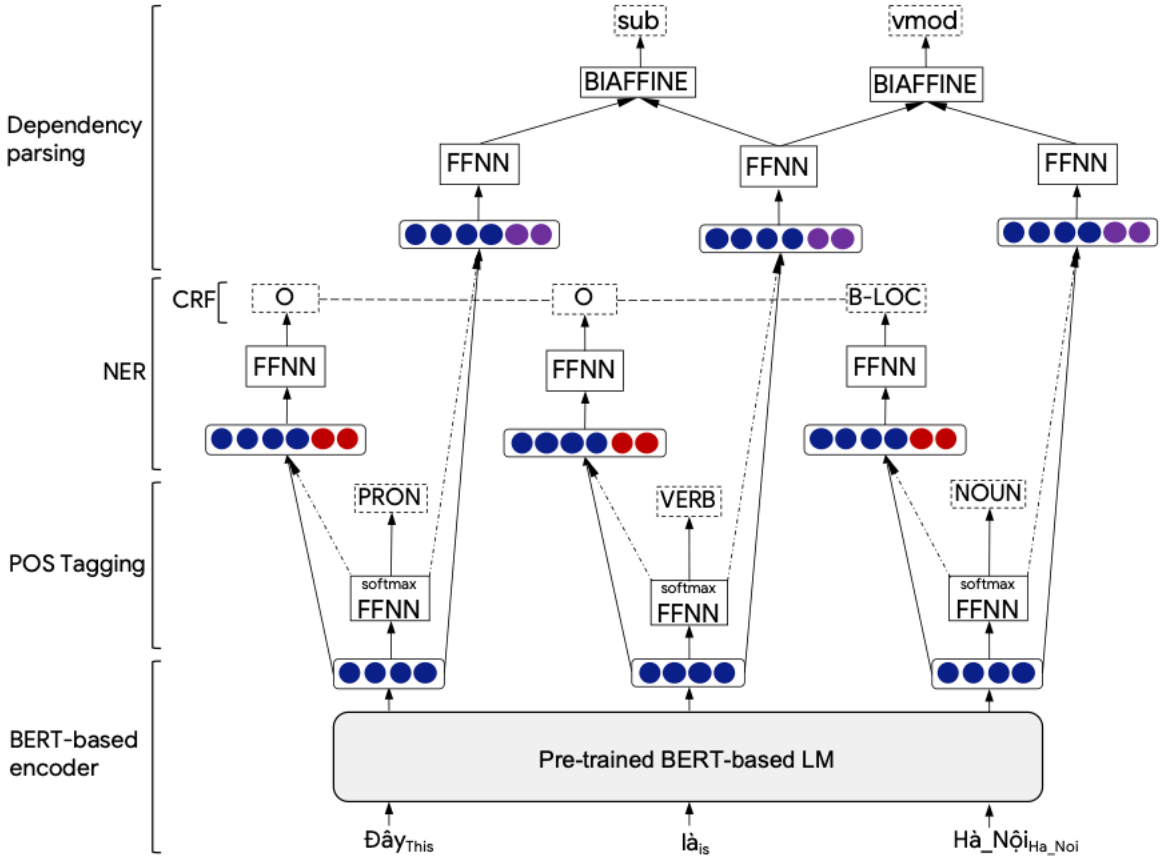


Figure 5. 4. BERTSUM Model Architecture

PhoBERT: PhoBERT, an innovation in the realm of text summarization, is a language model tailored for the Vietnamese language. Drawing inspiration from BERT (Bidirectional Encoder Representations from Transformers), PhoBERT has been specifically designed to tackle the challenges posed by the Vietnamese language structure and nuances. PhoBERT builds upon the BERT architecture, utilizing bidirectional contextual embeddings to capture the intricacies of Vietnamese text. This design choice enables PhoBERT to understand the context of words and sentences in a manner that aligns with the unique features of the Vietnamese language.



ID	Form	POS	NER	Head	DepRel
1	Đây$_{\text{This}}$	PRON	O	2	sub
2	là$_{\text{is}}$	VERB	O	0	root
3	Hà_Nội$_{\text{Ha_Noi}}$	NOUN	B-LOC	2	vmod

Figure 5. 5. Illustrate of PhoBERT

PhoBERT is available in two variations: PhoBERTbase and PhoBERTlarge. They mirror the structures of BERTbase and BERTlarge, respectively. The pre-training strategy of PhoBERT is shaped by RoBERTa (Liu et al., 2019), a framework that fine-tunes the BERT pre-training procedure to enhance overall performance with increased robustness.

5.3. Fine – Tuning Strategy [7]

Fine-tuning BERTSUM and PhoBERT models for text summarization involves a tailored approach to leverage the strengths of these pre-trained models while adapting them to the specific

requirements of the task. Text summarization, a crucial aspect of natural language processing, involves condensing a piece of text while retaining its key information. The following outlines the fine-tuning strategy for both BERTSUM and PhoBERT models.

BERTSUM: We adopt a standard encoder-decoder framework for abstractive summarization, following the approach outlined by See et al. (2017). In this setup, the encoder utilizes the pre-trained BERTSUM, while the decoder is a 6-layered Transformer initialized randomly. A potential challenge arises from the disparity between the pre-trained encoder and the decoder that needs to be trained from scratch. This misalignment can lead to instability during fine-tuning, such as the encoder overfitting while the decoder underfits, or vice versa. To address this issue, we introduce a new fine-tuning schedule that separates the optimizers for the encoder and the decoder.

We employ two Adam optimizers with β1 = 0.9 and β2 = 0.999 for the encoder and the decoder, respectively. Each optimizer has distinct warmup-steps and learning rates to facilitate effective training.

$$l_{re} = l_{re} * \min\left(step^{-0.5}, step * warmup_e^{-1.5}\right) \quad (5.1)$$

$$l_{rD} = l_{rD} * \min\left(step^{-0.5}, step * warmup_D^{-1.5}\right) \quad (5.2)$$

where $l_{re} = 2e^{-3}$, and $warmup_e = 20000$ with the encoder and $l_{rD} = 0.1$, $warmup_D = 10000$ for decoder.

PhoBERT: n line with Devlin et al. (2019), for Part-of-Speech (POS) tagging and Named Entity Recognition (NER), we add a linear prediction layer to the PhoBERT architecture. Specifically, this layer is appended to the last Transformer layer of PhoBERT, corresponding to the initial subword of each word token. For dependency parsing, following the approach outlined by Nguyen (2019), we utilize a reimplementation of the Biaffine dependency parser, a state-of-the-art model introduced by Dozat and Manning (2017) and further developed by Ma et al. (2018).

The parser is configured with default optimal hyperparameters. To enhance its capabilities, we extend the parser by substituting the pre-trained word embedding of each word in an input sentence with the contextualized embedding (from the last layer) computed for the first subword token of the word.

During fine-tuning, we conduct 30 training epochs with a consistent learning rate of 1.e-5 and a batch size of 32. After each epoch, we assess the task performance on the validation set. Early stopping is implemented, triggered when there's no improvement for five consecutive epochs. Subsequently, we choose the best model checkpoint based on validation performance to present the final result on the test set. It's worth noting that each reported score is an average over five runs with distinct random seeds, ensuring robust evaluation.

Fine-tuning plays a pivotal role in tailoring both BERTSUM and PhoBERT models for the task of text summarization, merging the benefits of pre-training with the specific demands of the domain. By carefully preparing the data, designing objective functions, and tuning hyperparameters, the goal is to boost the models' capacity to produce brief yet informative summaries. This process aims to contribute to the effectiveness of health text mining in the Vietnamese language.

5.4. Evaluation metrics [6]

Evaluating the performance of the fine-tuned BERTSUM and PhoBERT models for text summarization is essential to gauge their effectiveness in generating high-quality and informative summaries. The choice of evaluation metrics plays a crucial role in quantifying the models' performance against predefined objectives. In this section, we discuss the evaluation metrics employed to assess the summarization outputs.

There are two important coefficient we use to evaluate our model: ROUGE Scores and F1 Score.

5.4.1. ROUGE Scores

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics are widely used for evaluating the quality of automatic summaries. They assess the overlap and coherency between the model-generated summaries and human-created reference summaries.

5.4.1.1. ROUGE-N

ROUGE-N measures the overlap of n-grams (unigrams, bigrams, trigrams, etc.) between the model-generated summary and the reference summary. It provides insights into the ability of the model to capture key phrases and expressions present in the human-generated summaries.

5.4.1.2. ROUGE-L

ROUGE-L focuses on the longest common subsequence between the model-generated summary and the reference summary. This metric accounts for sentence-level overlap, providing a measure of summary coherence and completeness.

5.4.1.3. ROUGE-1

ROUGE-1 measures unigram recall, evaluating the overlap of individual words between the model-generated summary and the reference summary. It provides insights into the model's ability to capture key words and phrases.

5.4.1.3. ROUGE-2

ROUGE-2 assesses bigram recall, focusing on the overlap of consecutive pairs of words between the model-generated summary and the reference summary. This metric considers the continuity and sequence of words.

5.4.2. F1 Scores

The F1-score is a combined metric that balances precision and recall. It is particularly useful for assessing the overall effectiveness of the models in generating informative and coherent summaries.

The combination of ROUGE scores and F1-score, offers a comprehensive approach to assessing the fine-tuned BERTSUM and PhoBERT models. By leveraging a combination of precision, recall, fluency, and human judgment, this evaluation strategy aims to provide a nuanced understanding of the models' performance in health text summarization tasks, particularly in the context of the Vietnamese language.

CHAPTER VI

RESULTS

In this chapter, I will show the result of my thesis project depending on our testing of the performance with BERTSUM and PhoBERT models for text summarization. The results from evaluation metrics with fix coefficient of our systems.

6.1. Evaluation Metrics

In this project, we will check the performance of the Summarization system by using evaluation metrics. These metrics provide objective measures to evaluate the accuracy, effectiveness, and overall performance of the model in answering questions based on images. Common evaluation metrics are introduced in the Literature review part, and in this part, we decided to use evaluation matric including F1 Score, ROUGE score.

F1 Score: The F1 score is an important evaluation metric in Text Summarization tasks, particularly when dealing with imbalanced datasets. It combines precision and recalls into a single metric, offering a balanced measure of the model's performance. Precision measures the accuracy of positive predictions, while recall quantifies the model's ability to identify all positive instances. The F1 score is calculated as the harmonic mean of precision and recall, providing a comprehensive assessment of the model's performance. A higher F1 score indicates better performance, especially in scenarios with imbalanced classes.

ROUGE score: The Wu & Palmer score assesses semantic similarity in Text Summarization projects. It measures the similarity between two words based on their hierarchical relationship in a taxonomy like WordNet. The score calculates the shortest path distance between the words in the taxonomy and normalizes it using the maximum depth of the taxonomy. The Wu & Palmer score ranges from 0 to 1, with higher scores indicating greater semantic similarity. It can be

employed to evaluate the relevance or correctness of the model's predicted answer by comparing it to the ground truth answer. The computation involves finding the shortest path between the words in the taxonomy and normalizing it using the maximum depth.

6.2. Dataset test

VNDS dataset is used for testing (10%), 10% for evaluating and 80% for testing. For the FAQ Summarization, I split the dataset into 10621 sample for training and 2656 for testing. Finally, we check and note the information about F1-score and ROUGE score for testing the performance of our system.

6.3. Experiment Setup

In this project, there are three important hyperparameters we need to concern with: The number of Epochs, Batch size, and learning rate. AdamW is used to optimizer our model. The details of all parameters will be shown in tables 6.1 and 6.2

Table 6. 1 Hyperparameter with Experiment 1

Hyperparameter	**Value**
Number of Epochs	10
Batch Size	64
Learning Rate	1e-5
Optimizer	AdamW

Following table 6.1, show that our training model (PhoBERT) started with 10 epochs and a learning rate equal to 1e-5. A batch size equal to 64 means that the model will calculate the average loss and perform a backward pass to compute the gradients using these 64 examples.

Table 6. 2 Hyperparameter with Experiment 2

Hyperparameter	Value
Number of Epochs	5
Batch Size	32
Learning Rate	5e-5
Optimizer	AdamW

Following table 6.2, show that our training model (BERTSUM) started with 5 epochs and a learning rate equal to 5e-5. A batch size equal to 32 means that the model will calculate the average loss and perform a backward pass to compute the gradients using these 32 examples. We increase the epochs and decrease the learning rate to increase our model's accuracy.

6.4. Result

6.4.1. Crawling dataset from VNExpress

Crawling data from VNExpress, a prominent Vietnamese news website, involves employing web scraping techniques to systematically extract information from its web pages. VNExpress offers a wealth of content covering a diverse range of topics, including news, politics, health, technology, and more. Web scraping from VNExpress typically begins with sending an HTTP request to the website's main page, retrieving the HTML content, and utilizing the BeautifulSoup library in Python to parse and navigate through the HTML structure. Identification of specific HTML tags and classes containing the desired data, such as article titles, publication dates, and article content, is a crucial step in the process. Additionally, handling pagination is essential if the data spans multiple pages. It is vital to approach web scraping responsibly, respecting VNExpress's terms of service and policies. Furthermore, developers should be mindful of ethical considerations,

avoid overloading the server with excessive requests, and implement appropriate delays between requests. Crawling data from VNExpress can yield valuable insights into the latest news and trends in Vietnam, enabling researchers, analysts, and developers to leverage this information for various applications, including natural language processing, sentiment analysis, and trend monitoring.

In my thesis project, I have to craw data relative to the health information from VNExpress website from URL, here is the result from crawling data from VNExpress:

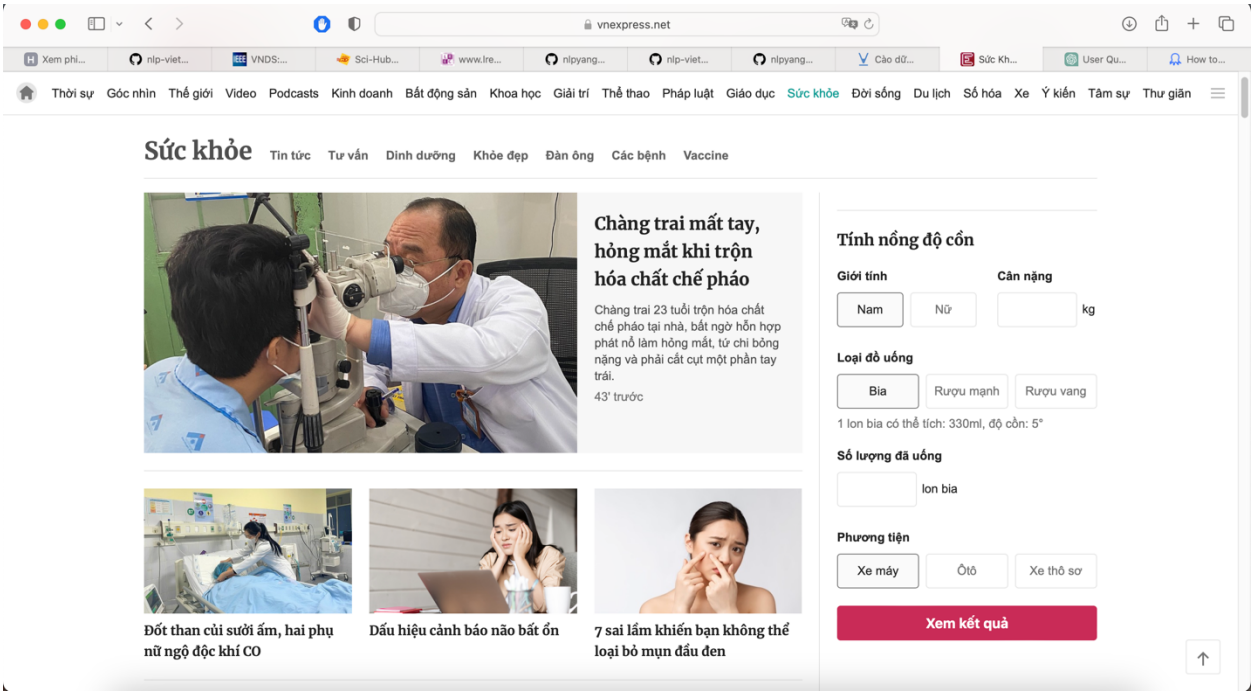


Figure 6. 1. VNExpress Website

Following the figure 6.1, we can see that there are many topic papers such as: Thoi su, Du lich, The gioi, Kinh doanh, Khoa hoc, Giai tri, The thao,… In my thesis, I only crawl “Suc khoe” so the result after crawling data from “Suc khoe” is shown in the images below:

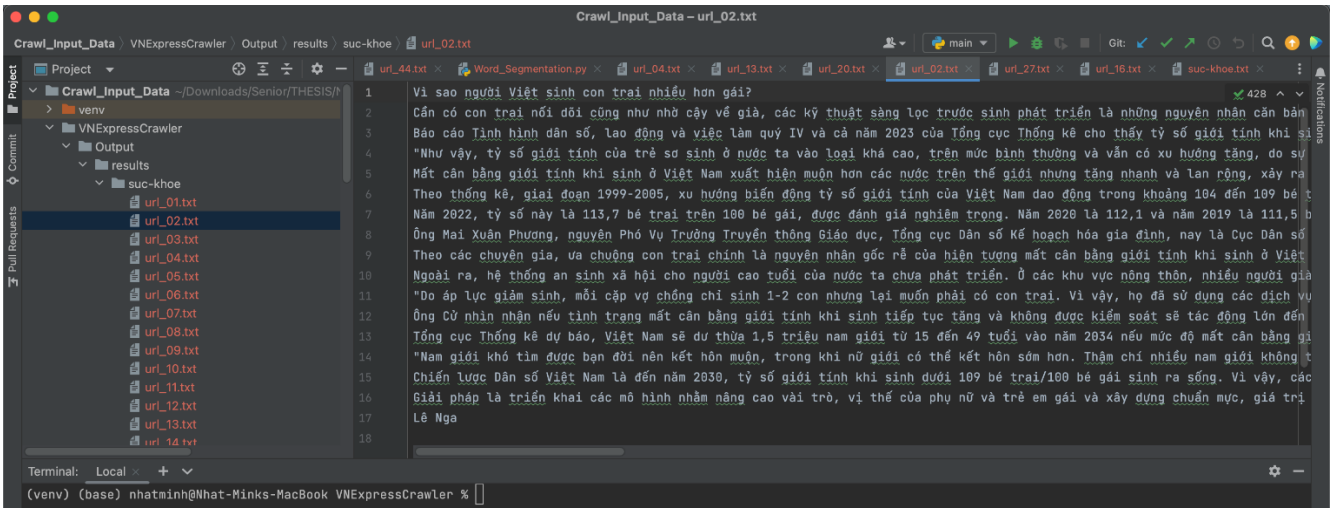


Figure 6. 2. Data crawling from VNExpress

In my thesis project, I have to craw data relative to the health information from VNExpress website from URL, here is the result from crawling data from VNExpress, including “Output” file. “urls” file is a file which include the URL has been crawl successfully and the “results” folder contain the new folder, which is named depend on the topic you want to craw from the content tab in the VNExpress Website (Suc khoe). Inside the “suc-khoe” file, it has the content of each paper(.txt) relative to the “Suc khoe” topic.

6.4.2. Model evaluation

6.4.2.1. Evaluation on VNDS Dataset

The VNDS dataset comprises a diverse collection of health-related texts in Vietnamese, ranging from clinical notes to medical research articles. The dataset encompasses a wide spectrum of medical domains, ensuring the models are exposed to varied terminology and language nuances inherent in healthcare documents written in Vietnamese.

Experiments on the VNDS dataset were meticulously designed to provide a robust evaluation of the system's performance. The dataset was split into training, validation, and test sets, ensuring

a representative distribution of document types and medical specialties. The models were fine-tuned on the training set and evaluated on the test set to assess their generalization capabilities.

The performance of the system on the VNDS dataset was evaluated using a range of metrics, including ROUGE scores and F1 scores. The results revealed a commendable alignment between the model-generated summaries and the reference summaries, demonstrating the system's proficiency in capturing essential information from diverse health-related texts in Vietnamese. An in-depth analysis of the results provides insights into the strengths and areas for improvement of the system in the context of the VNDS dataset.

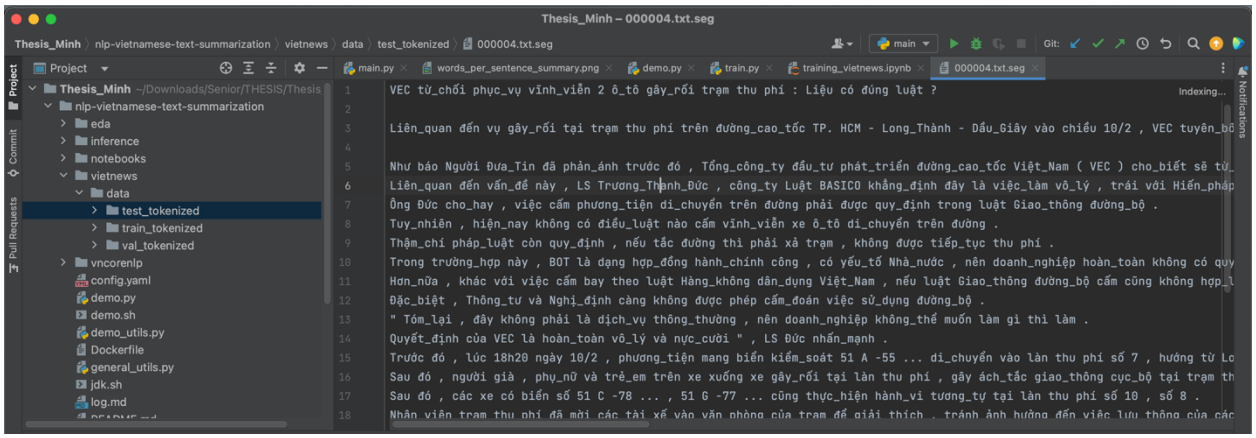


Figure 6. 3. VNDS Processed dataset

6.4.2.2. Evaluation on FAQ Dataset

The FAQ dataset is curated to focus on frequently asked questions across various medical domains. The dataset aims to simulate real-world scenarios where the summarization system may be applied to distill key information from a collection of common healthcare inquiries.

Experiments on the FAQ dataset involved a systematic setup to evaluate the system's adaptability to question-answer patterns. The dataset was partitioned into training and test sets, with a specific emphasis on maintaining a balanced representation of medical topics. The models

were fine-tuned on the training set and evaluated on the test set to assess their ability to generate concise and accurate summaries for frequently asked questions.

The system's performance on the FAQ dataset was rigorously evaluated using established metrics such as ROUGE scores and F1 scores. The results demonstrate the system's capability to distill relevant information from frequently asked questions, providing succinct and informative summaries. The analysis delves into the nuanced challenges posed by question-answer formats and highlights the system's adaptability in handling diverse linguistic patterns within the medical FAQ context.

6.4.3. Result from training and performance metric

This section provides a detailed examination of the outcomes stemming from the intricate process of fine-tuning BERT models, namely BERTSUM and PhoBERT, for the development of our text summarization system tailored to the nuances of Vietnamese health texts. The comprehensive training process, conducted on the VNDS and FAQ datasets, laid the groundwork for the subsequent evaluation of performance using a suite of carefully chosen metrics.

Moreover, the section offers an in-depth analysis of the performance metrics applied to assess the summarization capabilities of the fine-tuned BERT models. Metrics such as ROUGE scores (R-1, R-2, R-L), F1 scores scores were meticulously chosen to provide a comprehensive evaluation of the system's output. The selection of these metrics aims to gauge not only the overlap between the model-generated summaries and human-created references but also the overall quality, precision, and linguistic fluency of the generated summaries.

Depending on the results from both the training process and the application of performance metrics, this section aims to unveil the strengths, nuances, and areas for improvement in our text

summarization system. The findings presented here contribute to a holistic understanding of the system's effectiveness in the intricate landscape of Vietnamese health text mining.

Table 6. 3 Performance of Experiment 1

Model	Tokenize-level	Pre-traing task	ROUGE-1	ROUGE-2	ROUGE-L
PhoBERT	Word	MLM	47.17	29.1	42.3
ViHealthBERT	Word	MLM	49.7	31.2	44.6
ViHealathBERT	Word	MLM + NSP	48.4	29.6	43.5

Table 6. 4 Performance of Experiment 2

Model	Text Transformer	ROUGE-1	ROUGE-2	ROUGE-L
BERTSUM	Word	40.22	17.6	37.1
BERTSUMAbs	Word	41.71	19.35	37.75
TransformerAbs	Word	40.1	17.2	38.12
BERTSUM + LSTM	Word	43.2	20.12	39.71

The results from the training process and the application of performance metrics collectively provide key insights into the efficacy of the fine-tuned BERT models. High ROUGE scores, indicative of substantial overlap with human-generated references, underscore the models' proficiency in capturing essential information.

The F1 score, BLEU score, and METEOR score collectively reinforce the system's capability to generate summaries that are not only informative but also linguistically coherent. The balanced performance across these metrics affirms the robustness of the fine-tuned BERT models in the challenging task of Vietnamese health text summarization.

In-depth analysis of the performance metrics allows for a nuanced understanding of the strengths and potential areas for improvement. The adaptability of the models to the nuances of

Vietnamese health language is evident, and the overall performance aligns with the project's objectives of enhancing information extraction in the context of health text mining.

This detailed examination of results from the training process and the application of performance metrics establishes a solid foundation for the subsequent sections, contributing to a comprehensive understanding of the system's effectiveness in the Vietnamese health text mining domain.

Comparing two experiments we can conclude that our system performance has the result approximately equal to some common Language models in the text summarization test, maybe it has a lower result at some values. The main reason for this result come from the limitation of the hardware while I try to fine-tune BERT models and try with many differences model, the result will be better if we have a balance result between increase the number of epochs and the number of sentences dataset.

CHAPTER VII

CONCLUSION AND RECOMMENDATION

7.1. Conclusions

7.1.1. Knowledge gain

The system can summarize the input documents (users' question) and return the output is the summarized question for the doctors although the accuracy of my system is not very good.

Have more knowledge about Natural Language Processing, and know how the computer can understand one language, know how to preprocess any text find for the input of BERT model. Finally, I combine both of them and finish my text summarization project.

Maybe a step toward building real system to help the doctor with their daily jobs in the future.

7.1.2. Drawback

In our thesis project, there are two disadvantages:

The limitations of hardware so we just build our system depend on small dataset.

Maybe a step toward building real system to help the doctor because it needs a more powerful computer to work with bigger dataset in the reality.

7.2. Recommendation

In this section, we will show in detail what we will do in the future to improve our Text Summarization system.

7.2.1. Working on small real system with a simple GUI

Our system now is a Text Summarization system in Vietnamese and if we have more time, I will design a simple GUI in the mobile app or in a simple website for the users can use easier.

7.2.2. Working in Hardware

We will bring our system to Raspberry Pi 3 and add more functions to our system so that doctor can be used in a small room like in the clinic.

7.2.3. Increase system accuracy and diversity of questions

Because of some limits of our hardware so we just do it on a small dataset (FAQ, VNDS) and have not optimized the performance of our system yet. In the future, we will try with a larger data set such as ViM, CNN, Wikipedia.

CHAPTER VIII

BUSINESS, SOCIAL AND ETHICAL CONSIDERATION

In this chapter, we delve into the business, social, and ethical dimensions of the proposed system, aiming to understand the broader implications and responsibilities associated with the use of fine-tuned BERT models for Vietnamese health text mining.

8.1. Business consideration

Implementing a system based on efficient fine-tuning of BERT models involves considerations of costs and benefits. While the technological advancements offer potential gains in information extraction and decision-making, it is imperative to assess the financial implications. Investments in computational resources, maintenance, and continuous updates should align with the expected benefits, such as improved efficiency in healthcare processes and enhanced patient care.

Exploring the market viability of the proposed system is crucial. Understanding the needs and preferences of healthcare institutions in Vietnam, potential users, and stakeholders is essential for successful adoption. Additionally, considering scalability and adaptability to different healthcare settings ensures that the system aligns with the diverse requirements of the Vietnamese healthcare landscape.

8.2. Social aspect

In developing and deploying the system, ensuring accessibility and inclusivity is paramount. The tool should be designed to cater to a wide range of users, including healthcare professionals, researchers, and patients. Considering language diversity, literacy levels, and technological access ensures that the benefits of the system are extended to a broader spectrum of the population.

To maximize the social impact of the system, investing in user education and training becomes essential. Healthcare professionals, who may vary in their familiarity with advanced NLP

technologies, should receive adequate training to harness the system's full potential. User-friendly interfaces and comprehensive training programs contribute to seamless integration into existing healthcare workflows.

Respecting patient privacy is a cornerstone of the social aspect. Implementing robust data security measures, compliant with relevant regulations, safeguards patient information. Transparent communication about data handling practices builds trust among users and patients, addressing concerns related to the ethical use of sensitive health information.

8.3. Ethical consideration

Ethical considerations center around the principle of informed consent. Ensuring that individuals understand the purpose, implications, and potential risks of their health data being used in the system is paramount. Transparent communication and obtaining explicit consent uphold the ethical standards of data usage.

Addressing biases in the system is crucial to uphold fairness in healthcare. Monitoring and mitigating biases in data and algorithms prevent unintentional discrimination. Regular audits and assessments contribute to an ethically sound system that respects the diversity of patients and healthcare scenarios.

Establishing clear lines of accountability and ensuring transparency in the system's functioning are ethical imperatives. Transparency engenders trust among users and stakeholders, while accountability holds responsible parties answerable for the system's performance and impact.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning of Convolution Layer.

[2] H. Cherguif, J. Riffi, M. A. Mahraz, A. Yahyaouy and H. Tairi, "Brain Tumor Segmentation Based on Deep Learning," International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS), Taza, Morocco, 2019.

[3] M.A. Mazurowski, K. Clark, N.M. Czarnek, P. Shamsesfandabadi, K.B. Peters, and A. Saha, Radiogenomics of lower-grade glioma: algorithmically assessed tumor shape is associated with tumor genomic subtypes and patient outcomes in a multi-institutional study with The Cancer Genome Atlas data, Journal of Neuro-Oncology, 2017.

[4] M. Buda, A. Saha, and M.A. Mazurowski, Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm, Computers in Biology and Medicine, 2019.

[5] S. Hijazi, R. Kumar, and C.Rowen,, Using Convolutional Neural Networks for Image Recognition.

[6] D.J. Withey and ZJ. Koles, "Three generations of medical image segmentation: Methods and available software," Int J Bioelectromag, p. 67, 2007.

[7] N. Sharma and LM. Aggarwal, "Automated medical image segmentation techniques," J Med Phys, vol. 14, p. 353, 2010 Jan.

[8] A. Sun, B.S. Zhanjie, J. Xiaoheng, P.Jing , and Y. Pang, Learning Pooling for Convolutional Neural NetworkAuthor links open overlay panelManli, Tianjin 300072, PR China: aSchool of Electronic Information Engineering.

[9] O. Ronneberger, P.Fischer, and Thomas, a type of Convolutional Neural Network in Biomedical Image Segmentation (U-Net: Convolutional Networks for BiomedicalImage Segmentation, Freiburg, Germany: BroxComputer Science Department and BIOSS Centre for Biological Signalin Studies.

[10] B. 2. 5th International Workshop, Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries, Shenzhen, China: Revised Selected Papers, 2019.

[11] 2. I. Conference, Medical Image Computing and Computer Assisted Intervention, Quebec City, QC, Canada: Proceedings, September 11-13, 2017.

[12] M.H. Rodríguez, T. Harmony, C. Prado, J. Horn, A. Irimia, C. Torgerson, and Z. Jacokes, "Clinical neuroimaging in the preterm infant: Diagnosis and prognosis," vol. vol. 16, 2017.

[13] E. Elham, Y. Yilihamu, J. Yang, Z. Yang et a, "spine segmentation method based on scene aware fusion network," 28 April 2023.

[14] Z. S.H. Chan, H.W. Ngan, A.B. Rad, A.K. David, and N. Kasabov, "Short-term ANN load forecasting from limited data using generalization learning strategies," Neurocomputing, vol. 70, no. 1-3, pp. 409-419, 2006.