**Vietnam General Confederation of Labor**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL REPORT**

**DEEp learning**

*Instructor*: **Mr. LE ANH CUONG**

*Student:* **Nguyen Gia My – 521H0272**

**Le Tran Nhat Quang - 521H0413**

**Nguyen Dinh Viet Hoang – 522H0120**

*Course* : **503077**

*Year* : **25**

**HO CHI MINH CITY, 2024**

Vietnam General Confederation of Labor

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL REPORT**

**DEep learning**

*Instructor*: **Mr. LE ANH CUONG**

*Student* **Nguyen Gia My – 521H0272**

**Le Tran Nhat Quang - 521H0413**

**Nguyen Dinh Viet Hoang – 522H0120**

*Course* : **503077**

*Year* : **25**

**HO CHI MINH CITY, 2024**

ACKNOWLEDGEMENT

The first sincere thanks we want to give to Mr. Le Anh Cuong, who enthusiastically taught and worked tirelessly to give us enough tools and skills to complete this report. He played an important role in improving our mathematical logic and knowledge. The second thanks we would like to give to the teachers of the Department of Information Technology of Ton Duc Thang University for giving us the opportunity to do this report. Nevertheless, our report will have some errors, we are open to receive any feedbacks from teachers so that we can improve our report.

Finally, I wish you good health and success in your noble career.

*Ho Chi Minh city, 20th May, 2024*

*Author*

*Nguyễn Gia Mỹ*

*Lê Trần Nhật Quang*

*Nguyễn Đình Việt Hoàng*

**THIS PROJECT WAS COMPLETED AT**

**TON DUC THANG UNIVERSIY**

We fully declare that this is our own project and is guided by Mr. Le Anh Cuong. The research contents and results in this topic are honest and have not been published in any form before.

**Should any frauds be found, we will take full responsibility for the content of our report.** Ton Duc Thang University is not related to copyright and copyright violations caused by me during the implementation process (if any).

*Ho Chi Minh city, 20th May, 2024*

*Author*

*Nguyễn Gia Mỹ*

*Lê Trần Nhật Quang*

*Nguyễn Đình Việt Hoàng*

CONFIRMATION AND ASSESSMENT SECTION

**Instructor confirmation section**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh, , 2024*

*(Sign and write full name)*

**Evaluation section for grading instructor**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh , 2024*

*(Sign and write full name)*

SUMMARY

This report assumes general theory and model theory of question answering task.

INDEX

[ACKNOWLEDGEMENT i](#_Toc167141369)

[CONFIRMATION AND ASSESSMENT SECTION iii](#_Toc167141370)

[SUMMARY iv](#_Toc167141371)

[INDEX 1](#_Toc167141372)

[1. General theory: 2](#_Toc167141373)

[1.1. Overview: 2](#_Toc167141374)

[**1.2.** **What approaches can we use to deal with the problem:** 3](#_Toc167141375)

[2. Model theory used: 9](#_Toc167141376)

[2.1. Background and Related Work 9](#_Toc167141377)

[**2.2. Encoder-Decoder Models** 10](#_Toc167141378)

[3. REFERENCES 12](#_Toc167141379)

1. General theory:

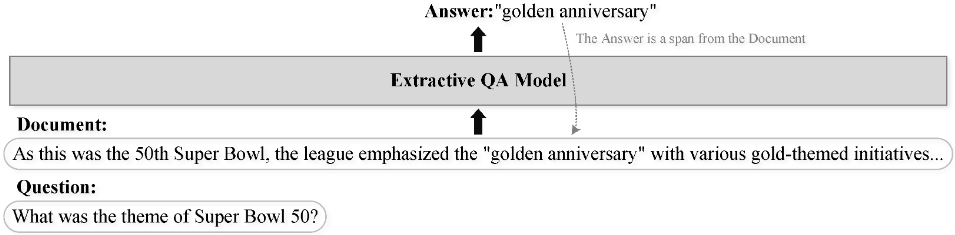
1.1. Overview:

Introduction: Extractive Question Answering, also known as computer literacy, can be used to assess a user's level of language comprehension. This is a valuable topic with many applications, limited to chatbots and personal support.

End-to-end neural network-based models have achieved outstanding performance in these tasks.

The next most frequently used way to extract answers using neural networks is to predict the starting and ending positions of the answer in the document, either independently or together.

How does it work:

We are given 2 datapoints: a question and a context, for example full wikipedia article that may contain an answer to the question. ****

Our goal is to produce an answer to the question based on the context. In the case of extractive question answering we assume that the answer is a subset of the context, so we can define it as span prediction, e.g. a range of characters or tokens.

We can also formulate this as an abstractive task, where we just want to obtain the answer, which may be phrased differently from our context.

**1.2.** **What approaches can we use to deal with the problem:**

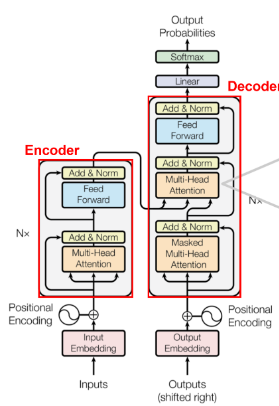
**1.2.1. Seq2Seq**

In a sequence to sequence approach, which can be associated with abstractive question answering, we use an auto-regressive language model to generate an answer.

We can use either question only, or a concatenation of context and question as the prompt. We train a model so that it learns to produce an answer as a continuation of given prompt.

During inference, we generate the answer step by step, using the language model to predict the most likely token until it comes up with the token representing end of sequence.

**Encoder:**

**Input**: is an input data string. The first part of Transformer's architecture is embedding, with two components are **Word Embedding và Positional Encoding.**

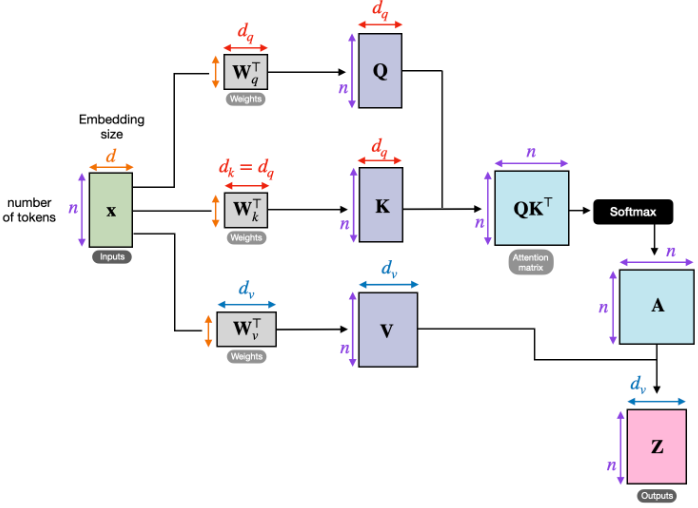
+ **Word Embedding**: is a method of representing words in sentences into feature vectors through a reasonable mapping of vocabulary words into vector space, so that these vectors represent the relationship between words. (text -> word tokenized -> word vectors)

+ **Positional Encoding**: Because the Transformer model does not sequentially calculate the order of words in a sentence, the Positional Encoding method provides information about the position of words in a sentence to the model.

The information obtained from **Positional Encoding** will be **added** to the **Word embedding vectors** of the words in the source and target language sentences. After going through Word Embedding and Positional Encoding, **we get an X matrix**

**Encoder**: is a combination of Word Embedding, Positional Encoding and a series of N consecutive Encoder Layers. In particular, the Encoder Layer includes many components such as Multi-Head Attention, Feed Forward, skip connection and Layer Normalization.

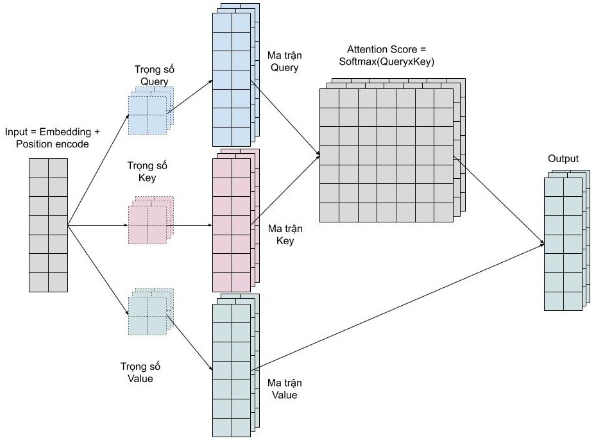
+ **Multi Head Attention** is essentially self-attention, but in order for the model to be able to pay attention to many different patterns, it is simply by using many self-attentions. **Self-Attention has 4 steps:**



1. **Create a set of 3 vectors from the input vectors of the encoder:** At the first encoder, the input vector is the word embedding of the word. So for each word, we will have 3 vectors Query, Key and Value. These vectors are created by matrix multiplication between the input vector and 3 weight matrices corresponding to Query, Key and Value that we use during training. These three vectors play different roles and are all important to attention.
2. **Calculate points:** For each word, we need to calculate the score of other words in the sentence relative to this word. This value helps decide which words should be given attention and how much attention when encoding a word. The score is calculated by the dot product between the Query vector of the word under consideration and the Key vectors of the words in the sentence. For example, when we calculate self-attention on a word with position 1, its score with itself is q1.k1, its score with the second word is q1.k2, etc.
3. **Normalize scores:** In the original article, the score is divided by 8 (square root of 64 – the dimension of the Key vector). This makes the slope more stable. Next, this value is passed through the softmax function to ensure that the point values ​​are all positive and the sum does not exceed 1**.**
4. **Multiply the Value vector** with each point value calculated above and then add them together. The idea is to preserve the vector value of words that need attention and eliminate the vector of irrelevant words (by multiplying it by a very small number, such as 0.001)**.**

+ Multi head attention allows the model to pay attention to easily observable patterns at the same time as follows:

* Pay attention to the preceding word of a word
* Pay attention to the next word of a word
* Pay attention to related words of a word



*Multi Head Attention*

**+ Feed Forward Network** (FFNin Transformer uses 2 fully connected layers with number of units and activation function ReLU in the first layer. The input of FFN is also a matrix.

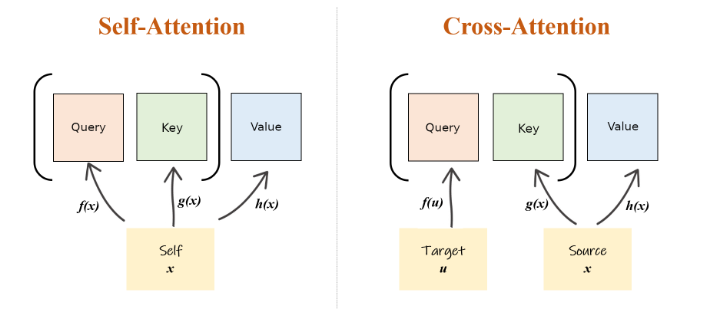
+ The encoder output will be a matrix with the same shape as the input matrix X of the Encoder. Thus, we can imagine that the Encoder is doing the task of adding important features to the original embedding vector of words in the sentence.

**Decoder**: also has a similar organization to Encoder:

1. Start with **Word Embedding and Positional Encoding.**

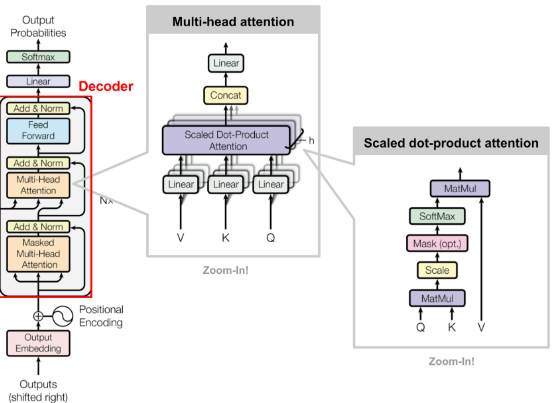
**2.** Then there is a **series** of **N Decoder Layers** **consecutively**.

In the Decoder Layer, we will perform both Self-Attention and Cross-Attention calculations. That:



**- Self-Attention** is performed first to carry out Attention to positions in front of the current position in the sentence, meaning we are using Masked Multi-Head Attention.

- Then, **Cross-Attention** is performed with two components 𝐾 and 𝑉 calculated from the Encoder output. At this time, we are performing Attention to all positions in the Encoder output, so Multi-Head Attention is used..



**Note:**

**-** Before going to the final fully connected layer of the Decoder to perform classification, the output matrix we receive will also = the output matrix of the Encoder Layer.

- The dimensionality of the output vector of the final fully connected layer will be equal to the size of the dictionary of the target language.

- **Add&Norm layer:** This layer simply normalizes the output of multi-head attention, effectively improving convergence.

**3.**  The last part of the Decoder is a fully connected layer with a softmax function

- The linear layer maps the context vector to a space of the same dimension as the vocabulary size. The softmax function then converts the resulting values ​​into probabilities, representing the likelihood that each token in the vocabulary is the next token.

- Finally, the predicted word is the word with the highest probability in the probability distribution.

**-** The predicted token is added to the input string and this process is repeated for the next encoding step **until the desired length or stopping condition is reached**.

**1.2.2. Start/End Positions**

This approach is associated with extractive question answering because we are trying to identify the span - start and end token indices - which represents the answer to our question.

To do this, we concatenate question and answer together, and pass it through a model that is typically a transformer with 2 linear heads corresponding to START and END logits.

For each token, we obtain probability that it is the START of the answer span (START logits) and that it is the END of the answer span (END logits).

Based on those probabilities, we can select the most likely span and use it as our answer prediction

2. Model theory used:

2.1. Background and Related Work

Extractive Question Answering: also known as span prediction or machine reading comprehension:

An extractive QA sample contains a question Q, a document D, and an answer A to question Q. 𝑄={𝑞1,𝑞2,…,𝑞|𝑄|} , 𝐷={𝑑1,𝑑2,…,𝑑|𝐷|}, and 𝐴={𝑎1,𝑎2,…,𝑎|𝐴|}. These are represented as word sequences where 𝑞𝑖, 𝑑𝑖, and 𝑎𝑖 denote the words, and |𝑄|, |𝐷|, and |𝐴| are the number of words in each sequence. The word sequence 𝐴={𝑎1,𝑎2,…,𝑞|𝐴|}

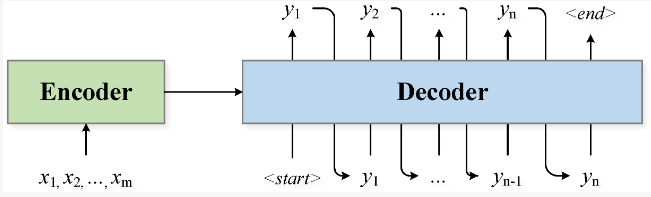
is a substring (subsequence occupies consecutive positions) in D, i.e., s and e exist and satisfy

{𝑎1,𝑎2,…,𝑎|𝐴|} = {𝑑𝑠,𝑑𝑠+1,…,𝑑𝑒}, where 0 < 𝑠 ≤ 𝑒 < |𝐷|.

We can solve this task by using an end-to-end model that takes Q and D as the input and A as the output. The model predicts the probability of a span S being the correct answer 𝑃(𝑆|𝐷,𝐴) (We refer to 𝑃(𝑆|𝐷,𝐴) as 𝑃(𝑆) for brevity). The span 𝑆={𝑑𝑠,𝑑𝑠+1,…,𝑑𝑒} denotes a substring in D.

**2.2. Encoder-Decoder Models**

Encoder-decoder models are also known as sequence-to-sequence models, which can generate word sequences according to the input word sequences



Generally, an encoder-decoder model 𝜋𝜃 estimates the conditional probability 𝑃(𝑌∣𝑋) with 𝜋𝜃(𝑌∣𝑋). The number of output candidates is exponential, and it is impractical to enumerate them to find the one with the highest probability. Thus, the 𝑃(𝑌∣𝑋) is conditionally factorized for the decoder to handle it in an autoregressive manner:

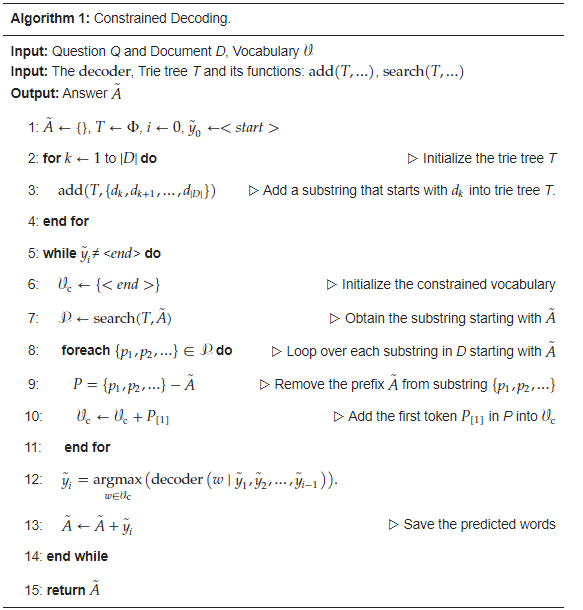
**𝑃(𝑌∣𝑋) = 𝑃(𝑦1∣𝑋)𝑃(𝑦2∣𝑋,𝑦1)𝑃(𝑦3∣𝑋,𝑦1,𝑦2)⋯𝑃(𝑦𝑛∣𝑋,𝑦1,𝑦2,…,𝑦𝑛−1).**

The decoder can generate the output sequence in a word-by-word greedy manner. The generation of the i-th word depends upon the already generated words:

**𝑦˜𝑖 = argmax𝑤∈𝒱(decoder(𝑤∣𝑦˜1,𝑦˜2,…,𝑦˜𝑖−1)).**

𝒱 represents the vocabulary containing all of the candidate words. The process starts with a special “start of the sequence” token representing an empty generated sequence and stops if gets a special “end of the sequence” token.

**Constrained Decoding:** the extractive QA ensures that the output answer is a substring in the document, so we need to limit the output space of the decoder, which is usually referred to as constrained decoding



3. REFERENCES

1. S. Li, C. Sun, B. Liu, Y. Liu, and Z. Ji, “Modeling Extractive Question Answering Using Encoder-Decoder Models with Constrained Decoding and Evaluation-Based Reinforcement Learning,” *Mathematics*, vol. 11, no. 7, p. 1624, Mar. 2023, doi: 10.3390/math11071624.