VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**PROJECT**

**BUILDING VIETNAMESE QUESTION-ANSWERING SYSTEM WITH DEEP LEARNING METHODS**

**Project 10**

*Student*: **NGUYEN TRUNG NAM – 519H0321**

Class **: 19H50302**

Year  **: 23**

*Supervisor*:  **Dr. BUI THANH HUNG**

**HO CHI MINH CITY, 2022**

VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**PROJECT**

**BUILDING VIETNAMESE QUESTION-ANSWERING SYSTEM WITH DEEP LEARNING METHODS**

**PROJECT 10**

*Student*: **Nguyễn Trung Nam – 519H0321**

Class **: 19H50302**

Year  **: 23**

*Supervisor*:  **Dr. BUI THANH HUNG**

**HO CHI MINH CITY, 2022**

# THANK YOU

I sincerely thank my guide Dr. Bui Thanh Hung, who enthusiastically imparted and taught me useful knowledge about machine learning, big data and deep learning, thank you for enthusiastically guiding me throughout my process. research, develop and complete this thesis.

**PROJECT COMPLETED**

**AT TON DUC THONG UNIVERSITY**

I hereby declare that this is my/our own project and is under the guidance of Dr. Bui Thanh Hung. The research contents and results in this topic are honest and have not been published in any publication before. The data in the tables for analysis, comments and evaluation are collected by the author himself from different sources, clearly stated in the reference section.

In addition, the project also uses a number of comments, assessments as well as data of other authors, other agencies and organizations, with citations and source annotations.

If I find any fraud, I will take full responsibility for the content of my project. Ton Duc Thang University is not related to copyright and copyright violations caused by me (if any).

*Ho Chi Minh, date*

*Author*

*(Sign)*

*Nguyen Trung Nam*

# LECTURER’S ASSESSMENT

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

*Ho Chi Minh, date*

*(sign)*

# SUMMARY

Briefly present the research problem, approaches, problem solving and some achieved results, basic findings within 1-2 pages

# CONTENT

[**THANK YOU**](#_heading=h.9h6ostyvhkil) **1**

[**LECTURER’S ASSESSMENT**](#_heading=h.sofx3py7fedr) **3**

[**SUMMARY**](#_heading=h.8wdisu9m3w61) **4**

[CONTENT](#_heading=h.1fob9te) **1**

[**BUILDING VIETNAMESE QUESTION-ANSWERING SYSTEM WITH DEEP LEARNING METHODS**](#_heading=h.y2gm1ma6l3ct) **3**

[1.1 Introduction to the problem](#_heading=h.rqz0k6s9y3am) **3**

[**1.2 Analyzing**](#_heading=h.u6ye015t427y) **3**

[1.2.1 Requirements](#_heading=h.q9xdvirfg9zn) 3

[1.2.2 Problem solving methods](#_heading=h.snpqji3td6zd) 13

[1.3 Model](#_heading=h.5ryqarwgmsw5) **13**

[1.3.1 Transformer model](#_heading=h.7g9uc5nzdybx) 14

[1.4 Experiments](#_heading=h.kw1l0sp64swv) **23**

[1.4.1 Data](#_heading=h.hw78dd39n7hk) 23

[1.4.2 Data processing](#_heading=h.bmmkh488q1su) 24

[1.4.3 Technology](#_heading=h.2wncano23ygt) 24

[1.4.4 Evaluation](#_heading=h.y6md6mfjtpz) 25

[1.5 Result](#_heading=h.nja2635aneih) 25

[1.6 Conclusion](#_heading=h.5w6y85o85a5o) 25

[**REFERENCE**](#_heading=h.y7b3zk5y0a3) **26**

6

**LIST OF FIGURE**

Figure 2.1: FTP Architecture [1](#_heading=h.44sinio)

Figure 2.2: TP Architecture 1

# BUILDING VIETNAMESE QUESTION-ANSWERING SYSTEM WITH DEEP LEARNING METHODS

# 1.1 Introduction to the problem

The research objective of the thesis is to apply learned knowledge to build an automatic reply system, using Deep Neural Networks deep learning network, based on sequence-to-sequence framework and attention mechanism to generate outputs the auto 2 answer from a corresponding input string. The model is trained end-to-end GNMT (Google's Neural Machine Translation) on an available open domain dataset. From there, build, install and test an automatic question-answering model through the use of neural networks to train a Vietnamese data warehouse based on a conversation dataset from Vietnamese NLP.

# 1.2 Analyzing

### 1.2.1 Requirements

What are the requirements for this problem, please specify those requirements?

1. **Natural language processing**

Natural language processing (NLP for short) are techniques and methods to support computer systems to understand, process, and recognize natural languages such as Vietnamese and English. Machine translation, information extraction and information retrieval are among the studies of natural language processing

Automatic machine translation is one type of research that has been developed for many years and has achieved good results in recent times thanks to the application of intelligent deep learning methods. There are many approaches to solve this problem such as: rule-based machine translation, statistical machine translation and example-based machine translation, …

* The rule-based machine translation method is built on the syntactic and semantic law system and must have a fairly complete dictionary of information for items such as semantics, pragmatics,....
* Statistical Machine Translation (SMT) is built based on statistical results from a bilingual corpus. The intermediate results of this machine translation method are statistical tables of words, phrases and transformation rules without the need for linguistic knowledge. With these 5 methods, the larger and better the corpus is, the more efficient the translation system will be. The current statistical translation method is improving translation quality by training models based not only on single words but also on phrases.
* The traditional example-based machine translation method uses sample sentences, also known as example sentences. These sentences are stored on a database with full information such as legends, links between the components of two sentences in two languages. This method also needs a set of syntactic rules of the source language sentences to build a database for example sentences. The word difference will be determined through the classification dictionary, the input sentence will be analyzed by syntactic rule set and determine the syntactic pair of source and target sentences. Another approach to the example-based machine translation method is to build a bank of example sentences. The source sentence only needs to match each part with the example sentence by suitable algorithms (using synonyms in the classification dictionary).
* corpus-based machine translation is now also being applied to many automatic translation systems, obtaining the correct source and target mapping pairs automatically is an essential requirement for corpus-based translation methods.

Information extraction (IE) is another branch of research that specializes in automatically extracting structured semantic information from unstructured or semi-structured data sources. ) such as text documents or web pages.

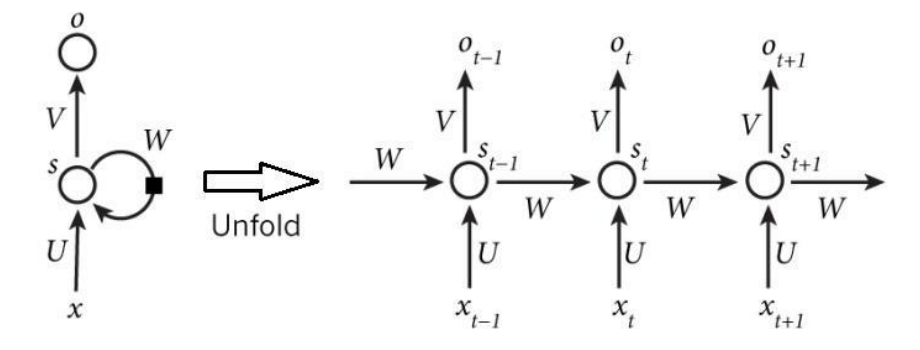
1. **Deep Learning**

Deep Learning is inspired by biological neural networks and includes many layers in an artificial neural network made up of hardware and GPUs. Deep Learning uses a cascade of layers of nonlinear processing units to extract or transform features (or representations) of data. The output of one layer serves as the input of the next layer. Deep learning focuses on solving problems related to artificial neural networks to upgrade technologies such as speech recognition, machine translation, natural language processing, etc.

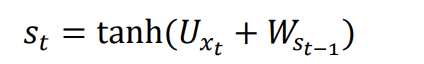
1. **Recurrent Neural Network(RNN)**

RNN is a memory model, capable of remembering previously calculated information. Unlike previous traditional Neural Network models, the input information is completely independent of the output information.

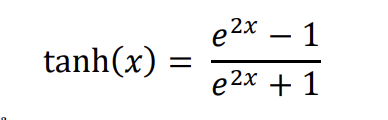
Most RNNs are designed as a series of iterative modules, which often have a simple structure with only one mesh layer. RNN is similar to traditional ANN training. The value at each output depends not only on the calculation result of the current step, but also on the calculation result of the previous step.



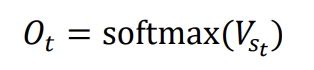
RNN has the ability to represent the dependency relationship between the components in the sequence (if the input string has 6 words, the RNN will spread out into 6 layers, each layer corresponds to each word, the index of each word is numbered from 0 to 1. 5. In Figure 2.8 above, xt is the input at time t, st is the hidden state at time t, calculated based on the previous hidden states combined with the input of the current time. with the formula:



The tanh function is a non-linear function of the form of a hyperbolic tangent defined by the formula:

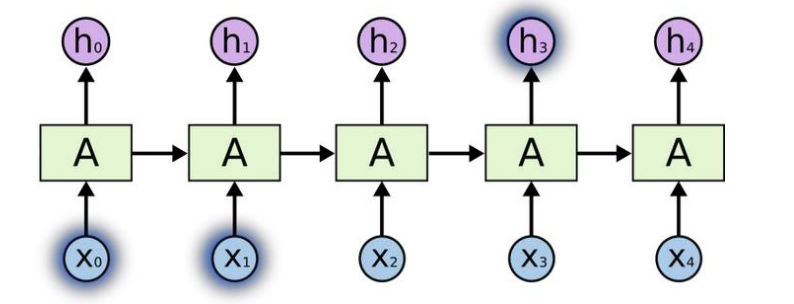


Ot is the output at time t, which is a vector containing the probabilities of all words in the dictionary. softmax is called the exponential mean function to generalize the logical function transform a vector K-dimensional space with real values to any K-dimensional space vector has a value in the range(0,1].



Unlike traditional ANNs, where a different parameter must be used at each layer, RNNs use only one set of parameters (U,V,W) for all steps.

The original idea of RNN was to connect previous information to support current processing. But sometimes, just rely on some recent information to perform the current task. For example, if we predict the last word in the sentence "flying dragonfly will rain", then we don't need to search too many words before that 12, we can immediately guess the next word will be "rain". In this case, the distance to the relevant information is shortened, and the RNN can learn and use the past information.



Where there is more information in a sentence, i.e. depends on the context. For example, when predicting the last word in the text “Tôi sinh ra và lớn lên ở Việt\_Nam … Tôi có\_thể nói thuần\_thục Tiếng\_Việt.” The closest word of information shows that the next word is the name of a language, but when we want to know specifically which language, we need to go back further, to find the Vietnamese\_Vietnamese context. And so, the RNN may have to find relevant information and the number of points becomes very large.

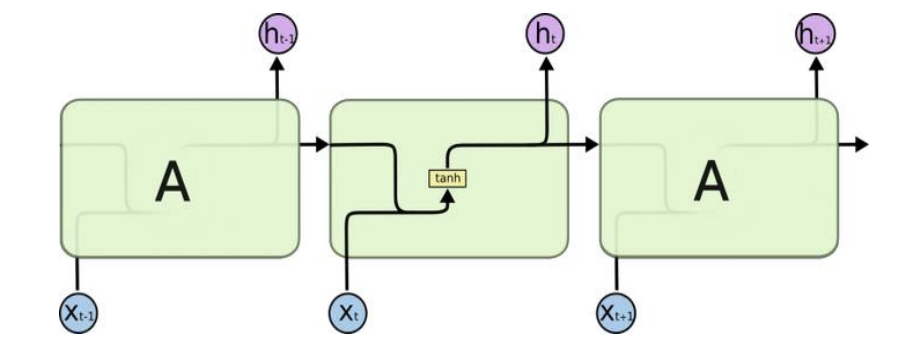
1. **Long-short term memory**

It is a special type of recurrent neural network, a technique based on gradients. During operation it allows to cut off the redundant gradients. During the LSTM learning process, it is possible to narrow the redundant delay time of the execution steps through the set of error constants

LSTM has the following basic components:

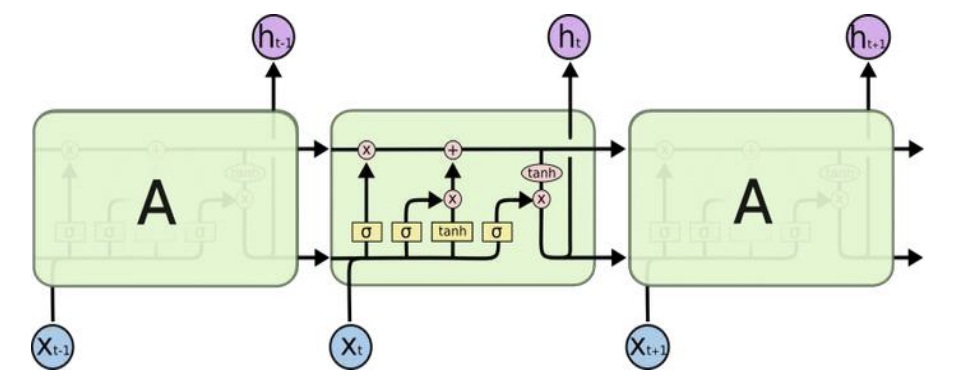
* Cell state
* Gate
* Sigmoid
* Tanh

LSTM is designed to eliminate the problem of too long dependencies. We observe the RNN model again below, the layers are all connected to each other into neural network modules. In standard RNN, this repeating module has a very simple structure consisting of only one simple layer.



Modul 1 layer

About LSTM network architecture: like RNN, it is a series of repeating modules. However, in terms of structure, LSTM has 4 layers of neural networks interacting with each other, called hidden layers. Some variations of LSTM are implemented based on changing the connection location between floors and ports.



Modul 4 layer

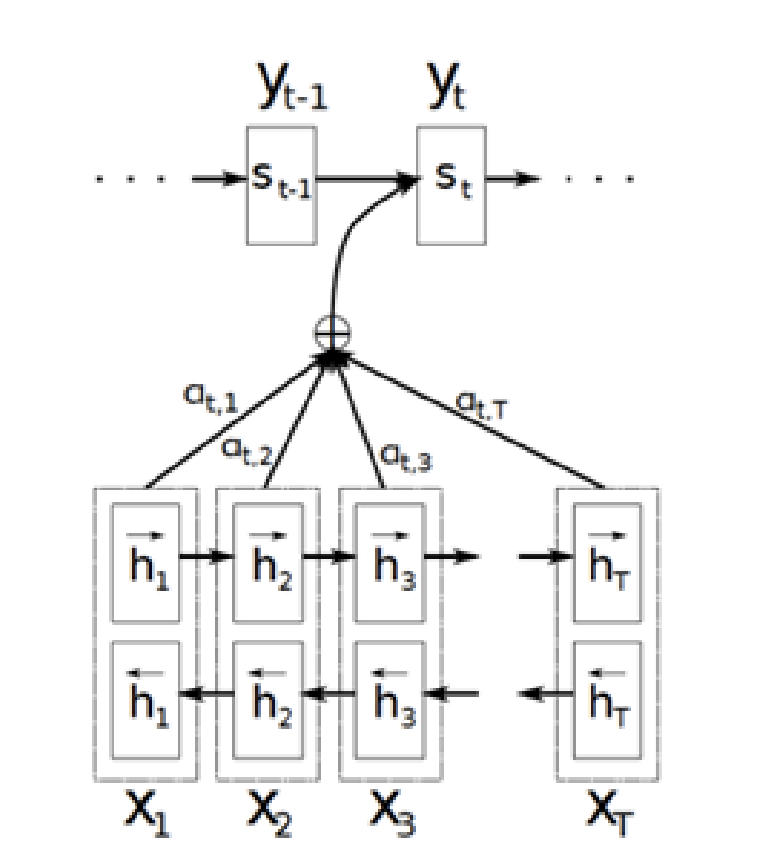
**Analysis of LSTM model:**

* LSTM is designed as a digital circuit board, including logic circuits and logic operations. Information moving in the circuit will be stored and propagated in the way that we design the circuit.
* The key to the LSTM is the cell state, which is the vertical line that runs across the top diagram. It is like a conveyor belt, running straight through the entire chain, only a few small linear interactions are performed, making the information in the process of propagation less changed.
* In LSTM, the gate structure is capable of adding or removing information to the cell state. These gates are created by the sigmoid function and a pointwise multiplication operator.
* The Sigmoid activation function describes the amount of information that is allowed to pass at each network layer, with a value from 0 to 1. Obtaining a value of 0 means "don't let anything through", otherwise if it is obtained. a value of 1 means "let everything through". An LSTM has three such ports to protect and control cell state.

1. **Attention technique**

The basic sequence-to-sequence model has two disadvantages: it requires the RNN decoder to use all the encoding information from the input string whether it's long or short, and the RNN encoder needs to encode the input string into a single vector, of fixed length. However, word generation at a time in the output sequence is more dependent on certain elements in the input sequence (such as the context surrounding the current word compared to other words in the sentence). The attention technique is introduced to solve this problem.

Instead of decoding the input sequence into a fixed context vector, with this attention technique, the words in the input sequence will be encoded by the RNN encoder into a sequence of vectors. Next, the RNN decoder applies soft attention by taking the sum of the weights of the sequence of encoding vectors (the weights are calculated by a neural network).

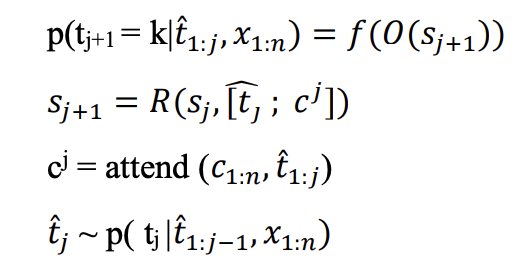


The sequence-to-sequence model uses soft attention in machine translation

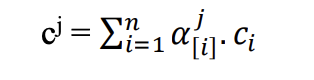
RNN encoder, RNN decoder and attention mechanism are trained simultaneously from the data. Attention uses GRU units instead of LSTM memory cells. In this case, a bidirectional input is used where the input sequences are fed both forward and backward, and then concatenated before being passed into the decoder.

The attention technique is performed from receiving the input string encoded with biRNN (bidirectional RNN) to generate n vector of the following form:

At each step j, the decoder will choose which parts of c1:n to generate the context vector c j through the attention technique to predict the jth step (c j = attend (𝑐1:𝑛,𝑡̂ 1:𝑗 ))



At each execution step, the context vector cj generated through the attention technique is the sum of the weights of the c1:n vectors with the following form:



The values of 𝛼[𝑖] 𝑗 calculated through the neural network MLP (multi layer perceptron) will be normalized by the softmax function.

### 1.2.2 Problem solving methods

There are two main approaches in the problem of building chatbots: Rule-based approach and Data-based approach.

* With the Rule-based approach, there are studies such as:
* Pattern-action rules (Eliza)
* A mental model (Parry)
* The Corpus-based approach includes studies such as:
* Information Retrieval
* Deep neural nets

Of these two approaches, the data-based approach has been studied and deployed a lot in recent years and has become the main research direction. In this thesis, I follow the second approach based on data, using LSTM deep learning network, applying sequence-to-sequence learning method and attention mechanism to generate automatic answers. verb from a corresponding input string. The model is trained end-to-end in the direction of GNMT (Google's Neural Machine Translation) and in the direction of question classification, based on the available data set CORNELL-MOVIE-DIALOGS and the dataset of questions collected.

# 1.3 Model

General training data model

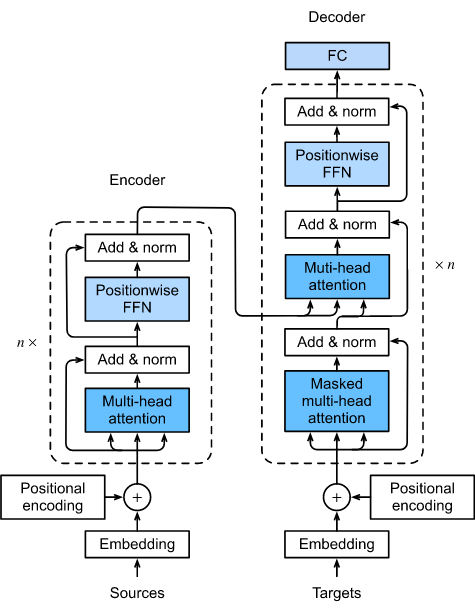
* Step 1: Collect data

We will use the conversations in movies and TV shows provided by Cornell Movie-Dialogs Corpus, which contains more than 220 thousand conversational exchanges between more than 10k pairs of movie characters, as our dataset.

* Step 2: Data normalization After data collection is completed, data normalization will be performed to remove inappropriate values, such as stop words, symbols. punctuation,….; recover error words….
* Step 3: Select training algorithm Select one of Machine Learning's algorithms to train the training dataset.
* Step 4: Divide the training and prediction dataset Divide the collected data set normalized by the ratio n: m, n parts are used for training, m parts are used for prediction
* Step 6: Storing the predictive model After the data training is completed, proceed to store the training model for future prediction.

### 1.3.1 Transformer model

Transformer is a deep learning model designed to serve to solve many problems in speech and language processing, such as automatic translation, language generation, classification, entity recognition, etc. speech recognition, text-to-speech. However, unlike RNNs, Transformers do not process elements in a sequence sequentially. If the input is a natural language sentence, the Transformer does not need to process the beginning of the sentence first and then the end of the sentence. Due to this feature, Transformer can take advantage of GPU parallel computing and reduce processing time significantly.



Do not use Recurrent architecture (regression) like RNNs, but Transformer uses self-attention. In its architecture, Transformer contains 6 encoders and 6 decoders. Each encoder contains two layers: Self-attention and feedforward network (FNN).

Self-Attention is a mechanism that helps encoders look at other words while encoding a particular word, so Transformers can understand the relationship between words in a sentence, even when they are far apart. The decoders also have the same architecture but between them there is an attention layer so that it can focus on the relevant parts of the input.

Self-Attention has 4 steps:

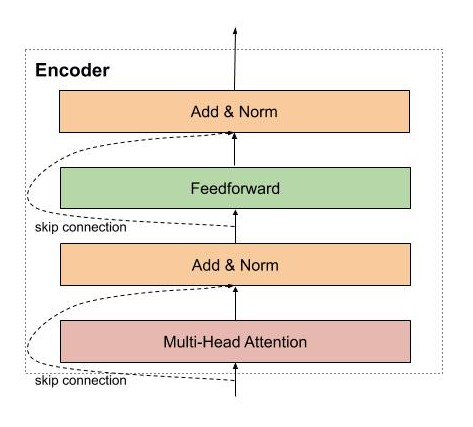
1. Generate a set of three vectors from the encoder's input vectors. At the first encoder, the input vector is the word embedding of the word. So for each word, we will have 3 vectors Query, Key and Value. These vectors are created by matrix multiplication between the input vector and the three weight matrices corresponding to the query, key, and value that we use during training. These three vectors play different roles and are all important for attention.
2. Calculate points. For each word, we need to calculate the score of other words in the sentence for this word. This value helps to decide which words need attention and how much attention when encoding a word. The score is calculated as the dot product between the Query vector of the word in question and the Key vectors of the words in the sentence in turn. 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, and so on.
3. Score normalization. In the original paper, the score is divided by 8 (square root of 64 – the number of dimensions of the Key vector). This makes the slope more stable. Next, this value is passed through the softmax function to ensure that the score values are all positive and that the sum does not exceed 1.
4. Multiply the Value vector by each of the above calculated point values and add them together. The intention of this is to preserve the vector value of the words that need attention and to remove the vector of unrelated words (by multiplying it by a very small number, for example 0.001).

**1.3.2 Features of the proposed model**

Encoder

The transformer model's encoder can consist of many similar encoder layers. Each transformer's encoder layer includes two main components: multi head attention and feedforward network, in addition to skip connection and normalization layer.

Of these two main components, you will be more interested in multi-head attention because it is a new layer introduced in this article, and it is it that makes the difference between the LSTM model and the Transformer model. we are finding out.



The first encoder receives the representation matrix of the words plus positional information via positional encoding. Then this matrix will be processed by Multi Head Attention. Multi Head Attention is actually self-attention, but in order for the model to be able to pay attention to many different patterns, the author simply uses multiple self-attention.

**Self Attention Layer**

You can imagine a self attention mechanism like a search engine. Given a given word, this mechanism will allow the model to search among the other words, which are "like" so that the information will then be encoded based on all of the above words.

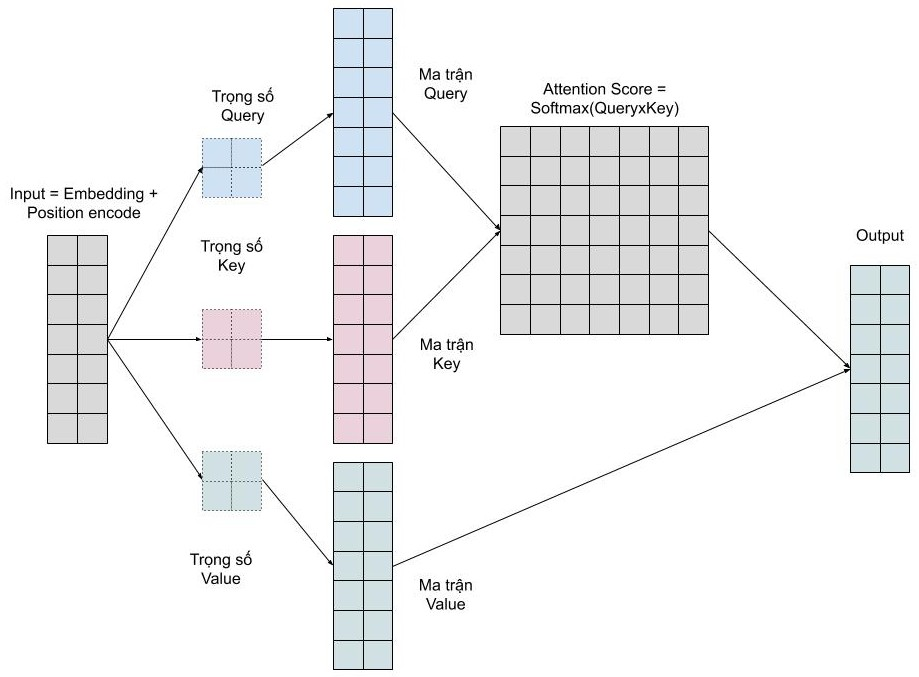
First, for each word we need to create 3 vectors: query, key, value vector by multiplying the matrix representing the input words by the corresponding learning matrix.

* query vector: vector used to contain information of the searched and compared word. Like a google search query.
* key vector: vector used to represent information about the words compared with the word to be searched above. For example, like the web pages that google will compare with the keywords you search for.
* value vector: vector represents the content and meaning of words. You can imagine, it's like website content that is displayed to users after searching.

To calculate the correlation, we simply need to compute the dot product based on the query and key vectors. Then use the softmax function to normalize the correlation index in the range 0-1, and finally, calculate the weighted average between the vector values using the newly calculated correlation index.

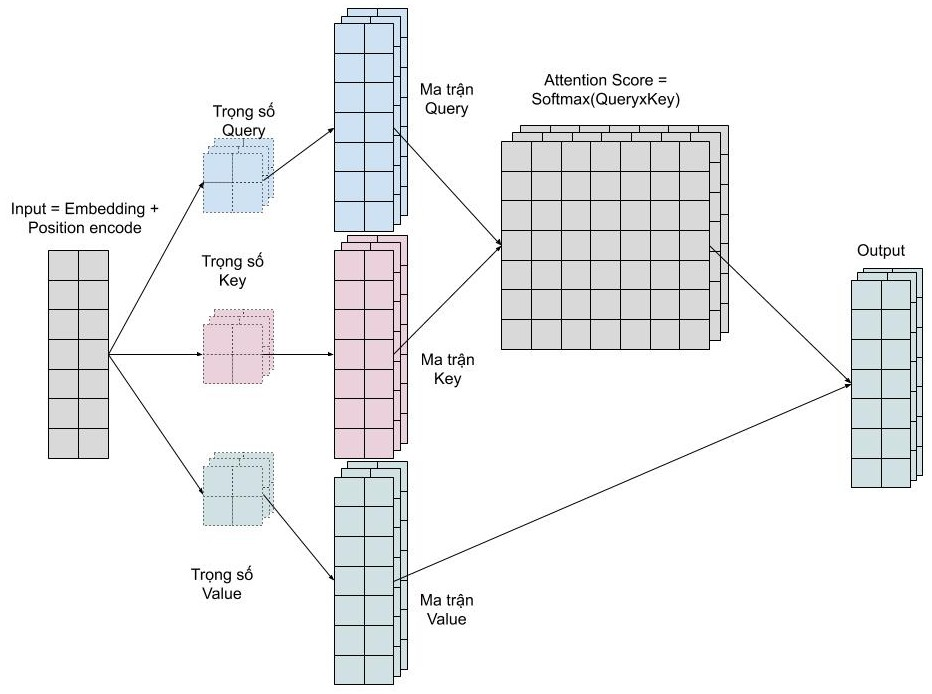
The attention vector computation process can be summarized into three steps as follows:

* Step 1: Calculate query, key, value matrix by initializing 3 weight matrix query, key, vector. Then multiply the input by these weight matrices to form 3 corresponding matrices.
* Step 2: Calculate attention weights. Multiply the two key and query matrices calculated above together to mean the comparison between the query and the key to learn the correlation. Then normalize to [0-1] using softmax function. 1 means the query is the same as the key, 0 means it is not.
* Step 3: Calculate the output. Multiply attention weights by the value matrix. This means that we represent a word as the weighted average of the value matrix.



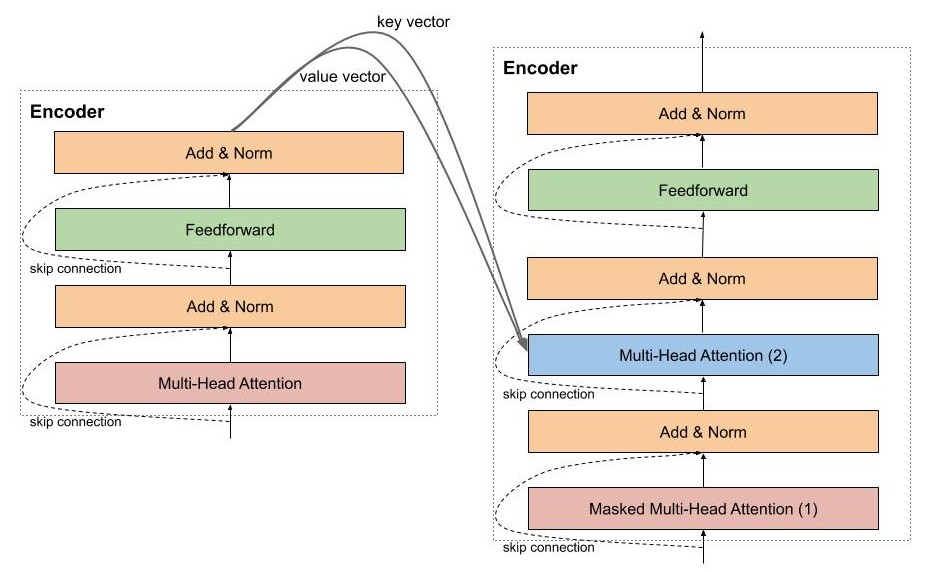
Multi Head Attention

We want the model to be able to learn many types of relationships between words. For each self-attention we learn a pattern, so to be able to extend this possibility we simply add more self-attention. That is, we need many query, key, value matrices only. Now the key, query, value weight matrix will have one more depth dimension.



**Decoder**

Decoder performs the function of decoding the source sentence vector into the target sentence, so the decoder will receive information from the encoder as 2 vector key and value. The decoder's architecture is very similar to that of the encoder, except that there is an extra multi head attention in the middle that is used to learn the relationship between the word being translated and the words in the source sentence.



**Masked Multi Head Attention**

Masked Multi Head Attention is of course the multi head attention that we talked about above, which has the function to encode target sentence words during translation, however, when installing, we need to note that we have to hide it. future words are not yet translated by the model, to do this we simply multiply by a vector containing the values 0.1.

In the decoder there is also another multi head attention function that notices the words in the encoder model, this layer receives the key and value vectors from the encoder model, and outputs from the layer below. Simply because we want to compare the correlation between the word being translated and the source words.

**Final Fully Connected Layer, Softmax and Loss function**

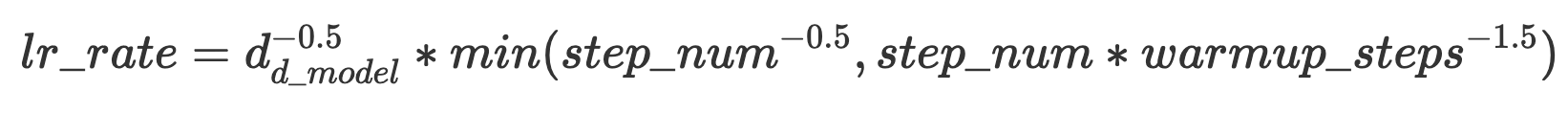
Like many other models, we need to add a fully connected layer to convert the output from the previous layer into a matrix with the word size that you need to predict. Then comes softmax so you can calculate the probability of the next word appearing.

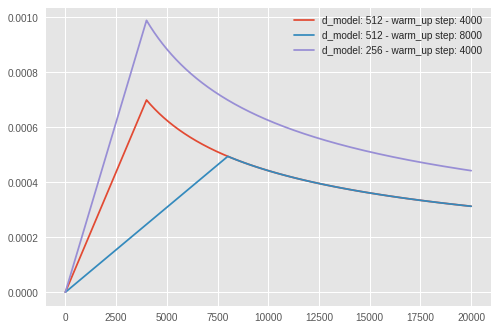
Loss function is of course only cross-entropy, like in other classification models that you are familiar with.

**Special Techniques for Transformer Training**

Optimizer

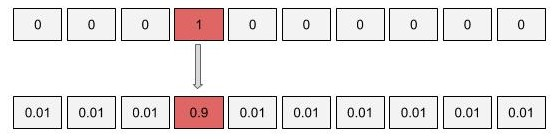
To train the transformer model, you still use Adam, however, the learning rate needs to be adjusted during the learning process according to the following formula





Label Smoothing

With the transformer's multi-million-parameter model, overfit is easy to happen. To limit the overfit phenomenon, you can use the label smoothing technique. Basically, the idea of this technique is quite simple, we will penalize the model when it is too confident in its prediction. Instead of coding the label as a one-hot vector, you'll change the label a bit by distributing a little probability over the remaining cases.



# 1.4 Experiments

### 1.4.1 Data

We will use the conversations in movies and TV shows provided by Cornell Movie-Dialogs Corpus, which contains more than 220 thousand conversational exchanges between more than 10k pairs of movie characters, as our dataset.

Brief description: This corpus contains a metadata-rich collection of fictional conversations extracted from raw movie scripts:

* 220,579 conversational exchanges between 10,292 pairs of movie characters - involves 9,035 characters from 617 movies
* in total 304,713 utterances -
* movie metadata included:
  + genres
  + release year
  + IMDB rating
  + number of IMDB votes
  + IMDB rating
  + character metadata included:
    - gender (for 3,774 characters)
    - position on movie credits (3,321 characters)

### 1.4.2 Data processing

To keep this example simple and fast, we are limiting the maximum number of training samples toMAX\_SAMPLES=25000 and the maximum length of the sentence to be MAX\_LENGTH=40.

We preprocess our dataset in the following order:

* Extract MAX\_SAMPLES conversation pairs into list of questions and `answers.
* Preprocess each sentence by removing special characters in each sentence.
* Build tokenizer (map text to ID and ID to text) using TensorFlow Datasets SubwordTextEncoder.
* Tokenize each sentence and add START\_TOKEN and END\_TOKEN to indicate the start and end of each sentence.
* Filter out sentence that has more than MAX\_LENGTH tokens.
* Pad tokenized sentences to MAX\_LENGTH

### 1.4.3 Technology

We use Python Language and mostly we use libraries of tensorflow.

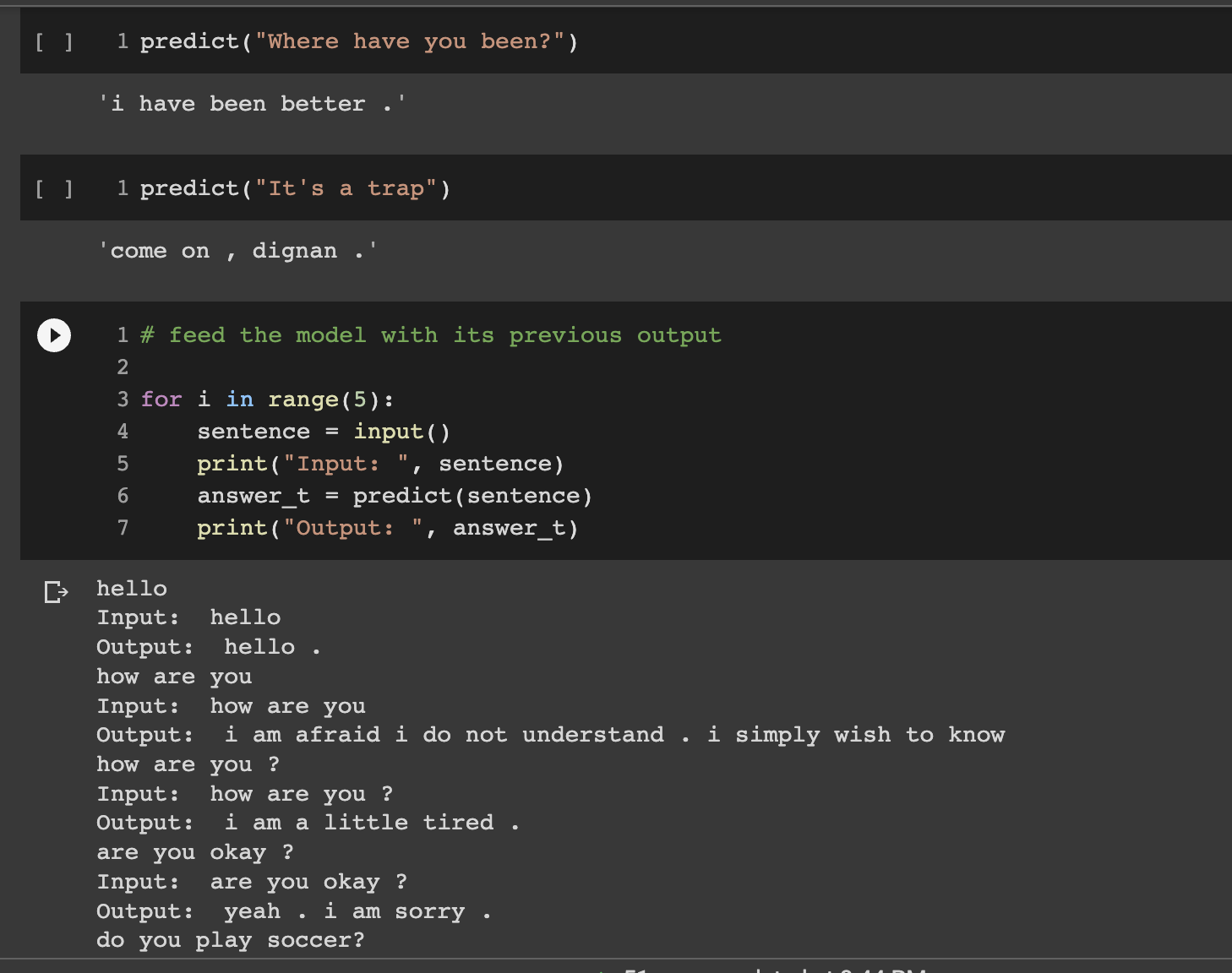
### 1.4.4 Evaluation

How to evaluate by any measure

How to calculate these measures?

# 1.5 Result

Here are some feedbacks obtained after training on the full dataset with a capacity of 34 MB (225000 + 225000) with parameter NUM\_LAYERS = 2 D\_MODEL = 256 NUM\_HEADS = 8 UNITS = 512 DROPOUT = 0.1 EPOCHS = 40 ​​. The initial test results generate the appropriate responses.



# 1.6 Conclusion

# REFERENCE

1. https://medium.com/mlearning-ai/nlp-day-22-how-to-create-a-chatbot-with-transformers-fbb194608217
2. https://blog.vinbigdata.org/transformer-neural-network-mo-hinh-hoc-may-bien-doi-the-gioi-nlp/
3. https://pbcquoc.github.io/transformer/
4. https://github.com/bryanlimy/tf2-transformer-chatbot
5. https://en.wikipedia.org/wiki/Transformer\_(machine\_learning\_model)
6. <https://machinelearningmastery.com/the-transformer-model/>
7. https://github.com/undertheseanlp/NLP-Vietnamese-progress

**SELF ASSESSMENT**

(For groups with 1 member)

| **No** | **Content** | **Benchmark** | **Evaluate** | **Note** |
| --- | --- | --- | --- | --- |
| 1  (9) | **1.1 Introduction** | 0.5 | 0.25 |  |
| **1.2 Analysis** | 1.0 | 0.5 |  |
| **1.3 Design** | 1.5 | 0.5 |  |
| **1.4 Experiment** | 4.5 | 1.5 |  |
| **1.5 Result** | 1 | 0.25 |  |
| **1.6 Conclusion** | 0.5 | 0 |  |
| 2  (1) | **Report (**Pay attention to the notes 2,3,4,6 on the previous page, if wrong, heavy points will be deducted**)** | 1đ | 0.5 |  |
| **Score** | | | 4 |  |