Question Answering System using BERT Language Model

Marija Chaushevska
Institution
Jožef Stefan International Postgraduate School
Jamova 39, 1000 Ljubljana, Slovenia
e-mail: Marija.Chaushevska@ijs.si

ABSTRACT

In recent years, Question Answering systems have become more popular and widely used by users. Despite the increasing popularity of these systems, their performance is not even sufficient for textual data and requires further research. These systems consist of several parts that one of them is the Answer Selection component. This component detects the most relevant answer from a given paragraph that contains the answer of the question.
Hence, in this paper, within Text Mining course, I am building and analyzing Question Answering System using the SQuAD dataset, that is fine-tuned on pre-trained BERT (Bidirectional Encoder Representations from Transformers) Language Model. The task for the seminar assignment is for a given question and reference text, which contains the answer, the model is required to mark the correct answer. The model performs very well on unseen paragraphs and given questions related to the paragraphs.

I. INTRODUCTION

Humans have always sought to find answers to their questions. Based on the type of questions they encounter, they are looking for answers. In the past, the questioner found many answers within the books. This method had two significant problems. First, all books were not readily available, and second, it took a long time to read the book and find the answer. With the advent of the Internet, resource inaccessibility problem was primarily resolved, but the second problem still remained. To overcome this problem, information retrieval systems and search engines have been developed. These systems receive a query from the user and return the documents containing the answer. The user could find the answer by going through these documents. The emergence of search engines was not a precise solution to the second problem because these systems returned the documents and the questioner needed to go through each of the documents in order to find the answer. To overcome the second problem, question answering systems were developed. These systems, instead of the whole document, return a word, phrase, or sentence as an exact answer.
Question answering is an important NLP task and longstanding milestone for artificial intelligence systems. Question Anwering systems allow a user to ask a question in natural language, and receive the answer to their question quickly and succinctly. Machine question answering remains one of the most important problems in artificial intelligence and natural language processing for both its rich potential in numerous applications requiring information synthesis and knowledge retrieval as well as its inherent difficulty, testing the boundaries of computers to process human language.
Some examples of natural language document collections used for question answering systems include:

- A local collection of reference texts
- Internal organization documents and web pages
- Compiled newswire reports
- A set of Wikipedia pages
- A subset of World Wide Web pages

The remainder of the paper is organized as follows:
In Section II, I will define the task (problem) and importance of Question Anwering Systems. Section III presents the dataset that is used for training and evaluation of the model. Section IV describes BERT (Bidirectional Encoder Representations from Transformers) language model and key ideas behind the model, as well as the reason why I have decided to use BERT for the Question Answering task. Section V shows how to fine-tune BERT pre-trained model on your own data and how it performs on the test data, when we pass paragraph and a list of questions. I conclude the paper in Section VI.

II. Problem definition – Question Answering task

Question answering (QA) is a well-researched problem [6] in NLP. In spite of being one of the oldest research areas, QA has application in a wide variety of tasks, such as information retrieval and entity extraction. Recently, QA has also been used to develop dialog systems and chatbots [7] designed to simulate human conversation. Traditionally, most of the research in this domain used a pipeline of conventional linguistically-based NLP techniques, such as parsing, part-of-speech tagging and coreference resolution.
Giving out a human-language question and giving a proper answer for it. Sometimes a specific question is asked and also sometime a open ended question can also be asked. Recent research works has given even more difficult

questions. Question answering application is the computer science field in which collective to natural language processing and information retrieval. Question Answering goes with building defined systems in which it answers the question that is posted by the humans in a natural language format. In implementation process, a computer program usually generates its own answers by asking or querying a structure database of information or knowledge which is usually termed as knowledge base. Sometimes even it can take out some answers from some unstructured dataset collection of information or knowledge. In Question Answering tasks, the model receives a question regarding text content and is required to mark the beginning and end of the answer in the text.

Figure1 shows an example of Question Answering system, where for each observation in the training set, we have a context, question, and text. The goal is to find the text (answer) for any new question and context provided. The sentence (answer) from the paragraph must semantically matches the question and extract the common semantic or syntactic factor from the context. This is a closed dataset meaning that the answer to a question is always a part of the context and also a continuous span of context. The highlighted yellow sentence on Figure1 is the sentence that has the right answer of the question, and from that sentence we have to get the correct answer (highlighted green).

As we can notice from the Figure1, no matter how easy or difficult the question may be, the answer to that question always lies within the paragraph itself.

From the wide range of Deep Learning approaches [5] and algorithms that are used to analyze and solve Question Answering problem, for this seminar work, I have decided to use BERT Language Model.

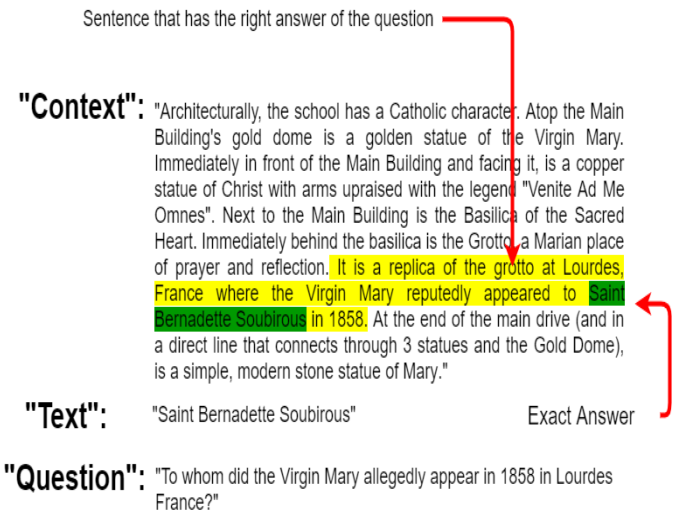

Figure1. Example of Question Answering System

III. SQuAD dataset

For training and evaluating Question Answering model I am using Stanford Question Answering Dataset (SQuAD 1.1) that I have downloaded from the [2] website and it contains two structured json files: train-v1.1.json (train data) and dev-v1.1.json (validation data or development set). Development set (dev-v1.1.json) is 4.7 MB, whereas train data (train-v1.1.json) is 29.6 MB. Samples in this dataset include (title, question, answer, context paragraph) tuples. The contexts and questions are just strings. The answers are dicts containing the subsequence of the passage with the correct answer as well as an integer indicating the character at which the answer begins. The train dataset consists of 442 different titles and 18,895 paragraphs, which are appropriately distributed in each title (each title consist different number of paragraphs). On the other hand, validation data consists of 48 different titles and 2067 paragraphs. Each paragraph in train and validation (development) data consisits of list of questions and answers related to the corresponding paragraph. SQuAD is a prime example of large-scale labeled datasets for reading comprehension. It contains over 100,000 question-answer pairs on 500+ articles. The QA goal is to find, for each question, a span of text in a paragraph that answers that question. Model performance is measured as the percentage of predictions that closely match any of the ground-truth answers.

The context paragraphs in SQuAD are from high-quality wikipedia articles, and cover a diverse range of topics across a variety of domains, from music celebrities to abstract concepts. A passage is a paragraph from an article, and is variable in length. Each passage in SQuAD has accompanying reading comprehension questions. These questions are based on the content of the passage and can be answered by reading through the passage.

One defining characteristic of SQuAD is that the answers to all of the questions are segments of text, or spans, in the passage (Figure2). These can be single or multiple words, and are not limited to entities – any span is fair game.

In addition, the span-based QA setting is quite natural. For many user questions into search engines, open-domain QA systems are often able to find the right documents that contain the answer. The challenge is the last step of “answer extraction”, which is to find the shortest segment of text in the passage or document that answers the question.

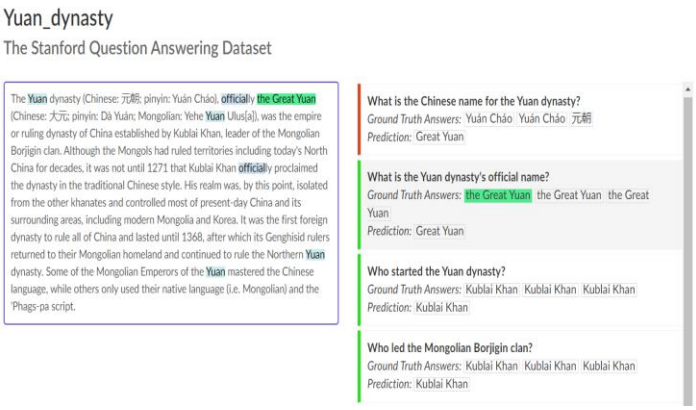

Figure2. Example of SQuAD 1.1 data

IV. BERT (Bidirectional Encoder Representation from Transformers)

Google BERT (Bidirectional Encoder Representations from Transformers) [1] and other transformer-based models further improved the state of the art on eleven natural language processing tasks under broad categories of single text classification (e.g., sentiment analysis), text pair classification (e.g., natural language inference), question answering (like SQuAD 1.1) and text tagging (e.g., named entity recognition). It is a pre-trained deep language representation model. It has pushed many NLP benchmarks to new state-of-arts. BERT was trained on Wikipedia and Book Corpus, a dataset containing +10,000 books of different genres. BERT exhibited unprecedented performance for modelling language-based tasks. BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial taskspecific architecture modifications. BERT is conceptually simple and empirically powerful. As a contextual model, it captures relationships in a bidirectional way.

Key ideas that BERT model is based on:

- Attention model - without RNNs (LSTM/GRU) is computationally more attractive and even has better performance (ability remember information beyond just about 100+ words) than RNNs.
- Representing words as subwords or ngrams - on average a vocab of 8k to 30k ngrams can represent any word in a large corpus.
- Encodes context bidirectionally
- Transfer Learning
- Word2Vec and GloVe word embeddings are context independent - these models output just one vector (embedding) for each word, combining all the different senses of the word into one vector.

BERT makes use of Transformer [10], an attention mechanism that learns contextual relations between words (or sub-words) in a text. Deep Transformer layers, enabling it to capture dependencies across long sequences. Transformer includes two separate mechanisms (Figure3) — an encoder that reads the text input and a decoder that produces a prediction for the task. BERT architecture does not contain LSTM networks, because they are slow to train ,words are passed sequentially and are generated sequentially, not truly bidirectional, but Transformer architecture address all of this concerns. They are faster and words can be processed simultaneously and the context of words is better learned, as they can learn context from both directions simultaneously. The main idea of Transformer was to combine the advantages from both CNNs and RNNs in a novel architecture using the attention mechanism. Transformer architecture achieves parallelization by capturing recurrence sequence with attention and at the same time encodes each item's position in the sequence. As a result, it leads to a compatible model with significantly shorter training time.

The input is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index. On the Figure3 is shown the Transformer architecture, which consists of two key components: encoder and decoder. The encoder takes the words simultaneously and generates embeddings for every word simultaneously. This embeddings are vectors that encapsulate the meaning of the word. The decoder takes the embeddings from the encoder.

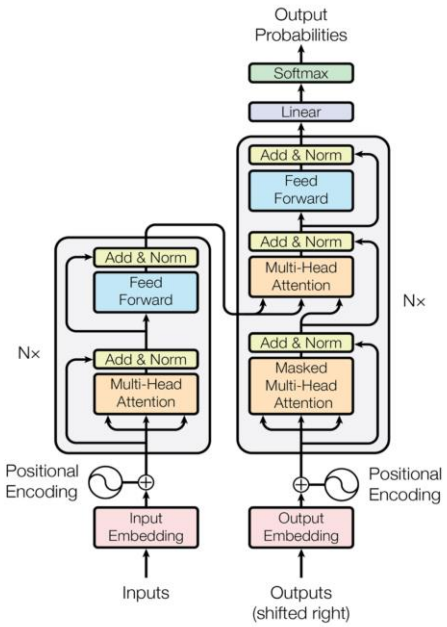

Figure3. The Transformer – model architecture

Training of BERT is done in two phases:

- Pre-training [1] (shown on Figure4 on the left) – where the model understands what is language and context. Pre-training of BERT is performed in an unsupervised manner on fairly large unlabeled text corpora (such as Wikipedia articles). During the pretraining phase, BERT performs two particular tasks: Mask Language Modeling (MLM) and Next Sentence Prediction (NSP).
  The MLM model randomly masks some of the tokens ,with a [MASK] token, from the input, and the objective is to predict the masked word based on its surroundings (left and right of the word). The idea is to learn the connection between smaller text components such as words and tokens.
  NSP task is performed in order to learn connections between sentences.
- Fine tuning (shown on Figure4 on the right) – where the model learns how to solve the problem. Fine-tuning of BERT is always associated with a particular practical task such as for example Question Answering. The pretraining version of BERT (that is, the weights obtained from the Masked Language Modeling and Next Sentence Prediction training routines outlined above) are used as starting weights for a supervised learning phase. Depending on the specific task, various components of BERTs input can be used. I will explain in detail the fine-tuned BERT model for Question Answering task on SQuAD model in the next Section V.

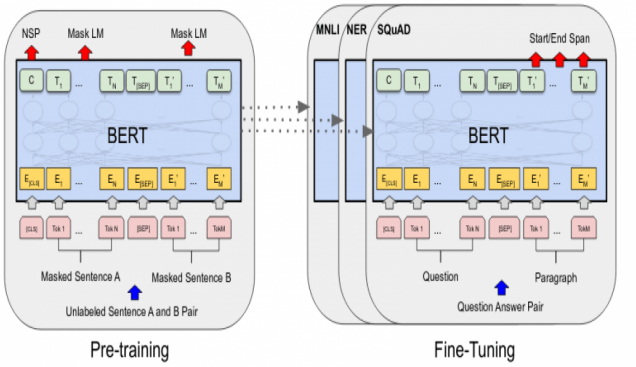

Figure4. Pre-training and fine-tuning phase

V. Fine-tuning BERT on my own (SQuAD) dataset

As I mentioned in the previous section, one of the key ideas that BERT Language Model is based on is Transfer Learning. The idea behind Transfer Learning is to take a model that was trained on a very large dataset and then fine-tune that model on your own dataset.

In the fine-tuning phase, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models and in the fine-tuning training, most hyper-parameters stay the same as in BERT training. For the Question Answering task, BERT takes the input question and the paragraph that contains the answer, and passage as a single packed sequence. In the latter, the BERT input sequence for Question Answering task, shown on Figure5, is the concatenation of the special classification token [CLS], tokens of a text sequence (question), special separation token [SEP] and tokens of the second text sequence (paragraph), and [SEP]. In order to do that, model needs a character position at which the answer ends in the passage (we are given the starting position).

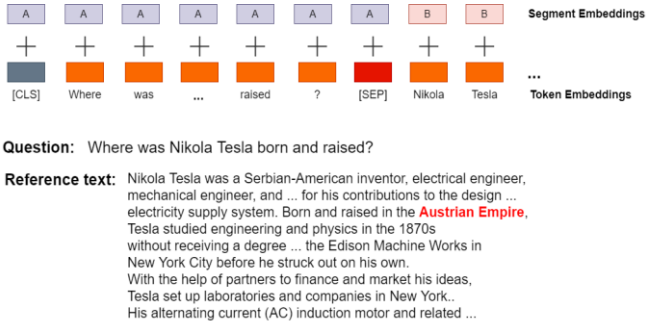

Figure5. Input in Question Answering System

The input embeddings are the sum of the token embeddings and the segment embeddings. BERT can take as input either one or two sentences, but for Question Answering task it takes two sentences. The input is processed in three vectors, three embedding layers (Figure6), in the following way:

- **Token Embeddings:** BERT passes each input token (the words in the input text) through a Token Embedding layer so that each token is transformed into a vector representation. It uses WordPiece embeddings with a 30,000 token vocabulary. The first token of every sequence is always a special classification token ([CLS]), as well as the token added at the end of the tokenized sentence ([SEP]).
- **Segment Embeddings:** A marker indicating Sentence A or Sentence B is added to each token. This allows the model to distinguish between question and paragraph sent to the model. In the below example, all tokens marked as A belong to the question, and those marked as B belong to the paragraph.
- **Position Embeddings:** This embeddings are indicating the position of each word in the sentence.

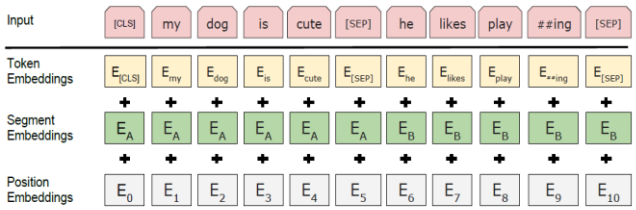

Figure6. BERT input embeddings

As I mentioned in the Section III, SQuAD 1.1 data files are large, as well as the fact that I am using Google Colab for running the model, I had to cut off the dataset. From the train dataset (train-v1.1.json) I have choosen 20 titles for training data, out of 442, and 12 paragraphs per title. From the validation dataset (dev-v1.1.json) I choose 5 titles, out of 48, and 5 paragraphs per titile. So in total, my train data has 240 paragraphs to be trained, and my validation data has 25 paragraphs to validate. Each paragraph in both datasets has a list of questions and answers.

Firstly, to train a model on the data you have, you need the tokenized context/question pairs, and integers indicating at which token positions the answer begins and ends. As I mentioned before, the role of the token embeddings layer is to transform words into vector representations of fixed dimension. In the case of BERT, each word is represented as a 768-dimensional vector. The tokenizer that I am using to tokenize context/question pairs is the BertWordPieceTokenizer.

I have completed Question Ansering task with fine-tuning BERT-base model, because the BERT-Large model requires significantly more memory than the BERT-Base. BERT-Base model has: 12 Transformer blocks (layers), 768-hidden vectors, 12- self-attention heads and 110M parameters in total. To fine-tune BERT for a Question-Answering system, it introduces a start vector and an end vector. BERT model is fine-tuned, so that the context (paragraph) and the question are preprocessed and passed as inputs. In the fine-tuned phrase BERT model expects three inputs: input_word_ids, input_masks_ids and input_types_ids. Also in the fine-tuning phase, it is necessary to modify the input and output layers. I pass the question followed by a passage (context) that contains the answer as a input and the output layer has two outputs: the first for predicting the probability that the current subtoken is the start of the answer and the second output for the end position of the answer. The probability of each word being the start-word is calculated by taking a dot product between the final embedding of the word and the start vector, followed by a softmax activation function over all the words. The word with the highest probability value is considered. A similar process is followed to find the end-word. Training the model is relatively straightforward. The bottom layers have already great English word representation, and I only really need to train the top layer, with a bit of tweaking going on in the lower levels to accommodate our task for question answering.

I fit the model, with 4 epochs and batch size 8. As an optimizer, I am using Adam optimizer from Keras and SparseCategoricalCrossentropy as a loss function. The model is trained on train dataset and it is validate on validation (development) data. A validation data is used to tune the hyperparameters (i.e. the architecture) of a classifier. I have achieved 57.30% accuracy on the training data, but let's not forget that the data on which that I was training the model is much smaller compared to the original SQuAD 1.1 dataset. I am quite sure that if I was using the whole SQuAD 1.1 dataset I would achive higher accuracy. The model was running approximately 12 hours on Google Colab.

I test the model on many examples, so that I pass paragraphs, that model has not seen in the train dataset, and a list of questions for each paragraph with a given title of the paragraph. Then I passed them in the model, and the model predicts the start and end token from the paragraph for each passed question, and outputs the answer. On the figures below (Figure7 – Figure 12) are shown results on some of the train data examples. As we can notice, if the model cannot find the answer of some question, then it leaves the space blank. For the test example shown on Figure7 (title: Supervised Learning), the model was not able to find the answer of the first question and left it empty. For examples shown on Figure8 (title: Oxygen) and Figure9 (title: Nikola Tesla), the model correctly answer all of the questions that I passed. For the test example shown on Figure10 (title: Maria Sklodowska Curie), the model did not answer on the second question. On the test example for Jozef Stefan Institute the model did not answer on the fifth question, as well as the answer on the fourth question should be only "404", but it answered "962 employees, 404". The last test example, shown in this paper, is a paragraph for myself (Figure12) where the model could not answer on only one question, but answered on all other questions correctly.

```
Q: What kind of task is supervised learning?
A:
Q: What will the optimal scenario allow for the algorithm?
A: correctly determine the class labels
Q: What does it infere?
A: labeled training data
Q: Whad does supervised learning algorithm analyze and produce?
A: an inferred function
```

Figure7. Test example – Supervised Learning

```
Q: The atomic number of the periodic table for oxygen?
A: 8
Q: How many atoms combine to form dioxygen?
A: two atoms of the element bind to formdioxygen
Q: Which gas makes up 20.8% of the Earth's atmosphere?
A: Diatomic oxygen gas
Q: What is the atomic number for oxygen?
A: 8
Q: Of what group in the periodic table is oxygen a member?
A: chalcogen
```

Figure8. Test example - Oxygen

```
Q: Where was Nikola Tesla born and raised?
A: Austrian Empire
Q: Who was Nikola Tesla?
A: Serbian-American inventor, electrical engineer, mechanical engineer,and futurist
Q: When was he studied engineering and physics?
A: 1870s
Q: What was he studied?
A: physics
Q: Where was Nikola Tesla worked?
A: the Edison Machine Works in New York City
```

Figure9. Test examplpe - Nikola Tesla

```
Q: Who is Marie Sklodowska Curie?
A: Polish and naturalized-French physicistand chemist
Q: In how many scientific fields she won the Nobel Price?
A:
Q: When she become a professor?
A: 1906
Q: Where she was professor in 1906 ?
A: Paris
Q: Where was she born?
A: Warsaw
Q: Where was she studied?
A: Warsaw's clandestine Flying University and began her practical scientifictraining in Warsaw.
```

Figure10. Test example - Marie Sklodowska Curie

```
Q: What is Jozef Stefan?
A: IJS) (Slovene: Institut Jožef Stefan
Q: What are the main research areas?
A: physics, chemistry, molecular biology,biotechnology, information technologies, reactor physics, energy and environment
Q: How many employees the institute had in 2013?
A: 962
Q: How many Ph.D scientists it had in 2013?
A: 962 employees, 404
Q: What is the mission of Jozef Stefan?
A:
Q: In which country is Jozef Stefan Insitute?
A: Slovenia
```

Figure11. Test example - Jozef Stefan Institute

```
Q: When was Marija Chaushevska born?
A:
Q: Where was Marija born?
A: Skopje, North Macedonia
Q: Who is her best friend?
A: Miljana
Q: Where does Marija study?
A: Ljubljana, Slovenia
Q: When was Miljana born?
A: 1998
Q: What is Marija's mother?
A: dentist
```

Figure12. Test example for myself

VI. CONCLUSION

In this paper, within Text Mining seminar, I showed how pre-trained BERT model works for Question Answering task and how to fine-tune BERT model on own dataset.
I could definitely conclude that BERT is undoubtedly a breakthrough in the use of Machine Learning for Natural Language Processing. The fact that it's approachable and allows fast fine-tuning will likely allow a wide range of practical applications in the future. We can use BERT to extract high-quality language features from the SQuAD text just by adding a single linear layer on top. From the test examples, shown above in Section V, I could definitely say that BERT model fine-tuned on small dataset works very well, on unseen examples, for Question Answering problem. In the future, I am very eager to try to build sequence-to-sequence attention Question Answering model using the newest version of SQuAD dataset [3] (version 2.0), which is much more bigger than the version 1.1, which I am using for this seminar work.

References

[1] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
[2]https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/
[3] Yuanjun Li, Yuzhu Zhang - Question Answering on SQuAD 2.0 Dataset
[4] Eylon Stroh, Priyank Mathur - Question Answering Using Deep Learning
[5] Yashvardhan Sharma, Sahil Gupta - Deep Learning Approaches for Question Answering System
[6] Do-Hyoung Park, Vihan Lakshman - Question Answering on the SQuAD Dataset
[7] Silvia Quarteroni. 2007 - A Chatbot-based Interactive Question Answering System. 11th Workshop on the Semantics and Pragmatics of Dialogue: 8390.
[8] Jamshid Mozafari, Afsaneh Fatemi, Mohammad Ali Nematbakhsh - BAS: An Answer Selection Method Using BERT Language Model

[9] Zhangning Hu - Question Answering on SQuAD with BERT
[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser - Attention Is All You Need