# CSDS493 Research Project Report
# Deep learning for Auto Comment Generator

Mingjian Lu mxl1171
Duo Xu dxx128
Jun Zhao jxz1395
Linjiang Ke lxk334
Ying Lu yxl3073

May 2, 2022

**Abstract**

This article mainly describes the process of our research project, from literature research to data collection to model selection and comparison, and finally draws conclusions. After literature research, our topic of choice is Review SynthesisForCodeCommits. It is designed to automatically generate text descriptions Source code natural language based on source code analysis, which can then reflect The design goals of the source code, the logic of the program, the role of the program, and The meaning of the relevant parameters. The general process can be described as follows, data collection. The dataset used to build the annotation generation system is used to train the model, validate and test, extract the code and corresponding annotations or extract the specific rules required by the annotation generation system. data can Obtain or download from open source communities or websites in the form of crawlers, and generate annotations based on an algorithm. This step can be divided into two subtasks: source code representation and text generation, quality assessment of the generated annotations. If the quality is not satisfactory, go back to the first step to prepare more and more suitable data, adjust the source code model, generate and evaluate the annotations again, and the process does not terminate until the generated code annotations meet the requirements. This article will summarize the process of our research project this semester and explain the above in detail. We choose to experiment and improve on the method of Xing Hu et al.'s article by reading the paper and propose our own topic Deep learning for Auto Comment Generator, use deep learning to train the model and compare it with common scoring methods in natural language processing. .

# Contents

# 1 Introduction

In the daily development and production process of developers, the comments on the code are often missing, expired or mismatched, which will greatly reduce the efficiency of our developers in the development process. At the same time, according to the need of collaborative work and improving code readability, developers often need to fill in the annotations, so a lot of time is wasted. And we can find that software maintenance usually occupies a lot of time in the software time cycle and more than 90% of the software engineering resources, and a large part of this is caused by the lack of code comments, whether intentional or unintentional.[SPL03] A good code comment generation tool can effectively improve the speed at which programmers understand the code because our code is generated according to the context and related comments, which means that it combines a lot of context information.

So in this paper, we implement the function of code comment based on java. Java is still the most popular programming language, and java code is widely used and has a relatively long history, the level of users is uneven, and there are many non-standard code projects, so we have adopted a new technology by using python, which can combine the advantages of traditional seq2seq-based model and the transformer-based model, and the pros and cons of the model are adjusted by adjusting the weight situation, so as to complete the comments of the code. In order to maximize the code comments accuracy, and at the same time, we call the generated comments Interpretable Comments because the comments generation process simulates the training process of the human brain language system.

As for how to define whether the completed code is efficient? We believe that considering whether a code is efficient for programmers, it should include three aspects: 1) what the methods do internally, 2) why these methods existed, and 3) how to use the methods for better improvement.[MM14] The annotations under this evaluation standard could better meet the needs of developers. Therefore, in the later stage, when we analyzed the efficiency of generated comments, we integrated several different methods, which were further improved based on ensuring that the original code did not affect our understanding. The completion of annotations not only requires our software to have a good generation function but also because annotations form part of the project, which requires them to have no major semantic errors and better readability. So in the progress, there have been high requirements for internal natural language processing algorithms and tuning methods.

# 2 Data

## 2.1 Data Introduction

We obtained the dataset from the project of Xing Hu, the author of Ref. and partially adapted the initial dataset provided by him and divided it into three files to form our dataset, which includes a training set, a validation set and a test set.

For the source file, we separated it into three files called camel_code, source_code and comments:

- camel_code is the code input for subsequent network training;
- source_code is processed to get AST, and then get sbt sequence;
- comments is the real value of the target.

## 2.2 Data Processing

### 2.2.1 Processing of Abstract Syntax Trees

To obtain the ast sequence of the code, we need to select the traversal method to traverse the abstract syntax tree of the code. There are four important methods of tree traversal, prior-order traversal, middle-order traversal, post-order traversal, and hierarchical traversal, and the difference between these four access methods is the different order of accessing each node of the tree. According to the characteristics of the abstract syntax tree, we prefer that the children of the same parent node can be clustered together and not separated, and keep the sequential order; at the same time, the parent

node follows the child node, so that it can reflect the structural information of the child node. Based on this idea, we propose the traversal method shown in Figure 1 for traversing the information of the abstract syntax tree and being able to retain its structural information.

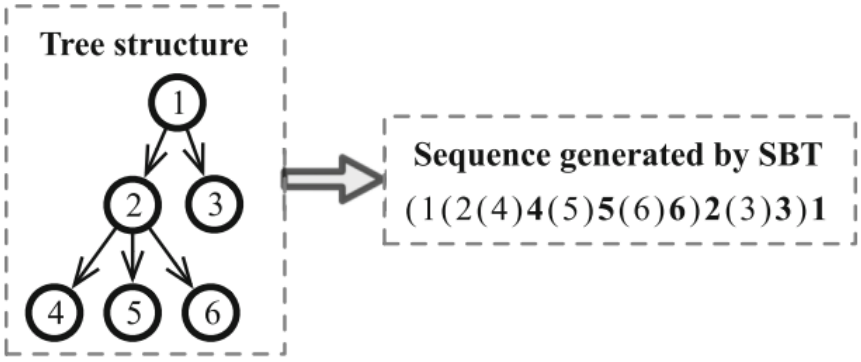


Figure 1: An AST transfers to a sequence by SBT.

#### 2.2.2 Steps of Data Processing

After defining the dataset path, we processed the data in the following steps:

1) Add start token and end token to the code and comments;
2) Construting three dictionaries:
   - The first dictionary counts the frequency (number of occurrences) of each different word, using a dictionary record that sorts the words in this dictionary by the number of occurrences, the higher the number of occurrences, the higher the sorting;
   - The second and third return word->id and id->word dictionaries, which start traversing each word according to frequency, from highest to lowest, and construct a unique id for this word, and in the id->word dictionary, if the frequency of occurrence of the word does not exceeds frequnecy, then the id will point to the unk token.
3) Vectorization, converting the code input into vector input and unifying the length of the input;
4) Divide the dataset according to the following ratio test:val:train=20000:20000:445812;
5) Wrapping the dataset.

# 3 Approach

## 3.1 Evaluation approach

In our project, we used the BLEU technique to evaluate our three models. BLEU[PRWZ02] is a method for automatic evaluation of machine translation proposed by Papineni et al. Its full name is bilingual evaluation understudy. BLEU score has been a widely used accuracy measure for NMT. It calculates the similarity between the generated sequence and the reference sequence, and is used, for example, to assess the correspondence between machine translation and professional human translation. In terms of assessing the reliability of the results, the results of BLEU are highly compatible with the manual evaluation, and thus have been widely used in the evaluation of machine translation and the evaluation studies of automatic commenting systems. the key evaluation component of BLEU is n-gram accuracy, i.e., the proportion of n-grams to be matched out of the total number of n-grams in the translation under evaluation, and n can be 1, 2, 3, or 4. respectively, for each n-gram precision is calculated for each n-gram, and the final precision is calculated as a weighted geometric mean. For

candidate texts, the more similar they are to the reference text, the higher the score.

However, the matched sentences may have a high n-gram match due to their short length, so it may happen that the translation system only translates a part of the sentence in the translation task, but the translation is very accurate, then it will still have a high match. To avoid this phenomenon, in the final scoring result, the authors introduced a length penalty factor with the following evaluation formula:

$$BLEU = BP * e^{(\sum_{n=1}^{N} W_n log P_n))}$$

where $W_n$=$\frac{1}{N}$. The upper limit of N is 4, and in our project, we set $N$=4. $P_n$ is the ratio of length n subsequences in the candidate sequence and also in the reference sequence. $BP$ is the brevity penalty.

$$BP = \begin{cases} 1 & \text{if} \quad c > r \\ e^{(\frac{1-r}{c})} & \text{if} \quad c \leq r \end{cases}$$

where $c$ represents the length of candidate sequence and $r$ represents the length of reference sequence.

## 3.2 Train approach

### 3.2.1 Seq2seq Model

The first model we used to train our data sets is seq2seq model[SVL14]. This model is widely used in machine translation. For example, Google Translate has adopted the seq2seq model since 2016. Seq2seq model is consisted of 2 RNN networks. As you can see in the Figure 2, on the left side is the first RNN network, it is called Encoder, and on the right side is the other RNN network it is called Decoder. Our seq2seq model introduces an attention mechanism[LPM15], so that its prediction accuracy will be higher. The processes of seq2seq model with attention mechanism are as follows:

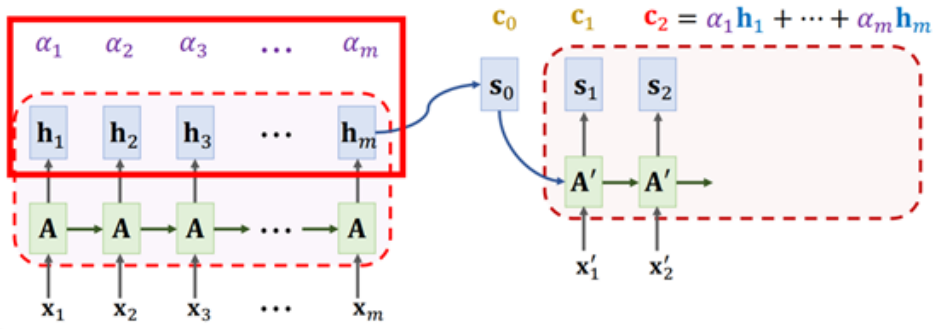


Figure 2: Structure of Seq2seq Model.

1) The Encoder computes the Hidden State that is generated for each time step. e.g. the input is $x = [x_1, x_2, ..., x_m]$ and the output corresponding to each time step is the vector $h = [h_1, h_2, ..., h_m]$. In the original Seq2Seq model, only the last state $h_m$ is preserved, but in Seq2Seq with attention model, each intermediate state $h_i$ is preserved.

2) At this point, the first input state of Decoder is $s_0$, and the Alignment Score of $s_0$ with each $[h_1, h_2, ..., h_m]$ is calculated (it can be interpreted as the correlation score of $s_0$ with each $h_i$, and there are various ways to calculate this Alignment Score: the vector $a = [a_1, a_2, ..., a_m]$.

3) Then the Alignment Score is softmaxed so that all elements of the vector $a$ are compressed to the range [0,1] and the sum of all elements is 1 (that is, each element represents 1 scale value).

4) The Context Vector $c_0 = a_1 * h_1 + a_2 * h_2 +, ..., + a_m * h_m$ is obtained by making an inner product of the vector $a$ and $h$.

5) The Context Vector $c_0$ is concatenated with the input state vector $s_0$ of Decoder and the input $x'0$ of Decoder's current time step $t_0$ to get $[c_0, s_0, x'_0]$, which is input to Decoder to get Decoder's state $s_1$ for the next time step $t_1$. If this is a machine translation task, $s_1$ (e.g., input a fully connected layer) is further used to predict the output $x'_1$ of this time step $t_0$ and used as input for the next time step $t_1$.

6) Iterate through steps 2-5 above until the Decoder reaches the specified length or outputs the specified stop signal (e.g., output [END])

### 3.2.2 Transformer Model

Another model we use in our project is Transformer model[HLX+18].Transformer is a model that utilizes attention[VSP+17] mechanism to increase the speed of model training. For the attention mechanism, please refer to this article. trasnformer can be said to be a deep learning model based entirely on the self-attention mechanism, because it is suitable for parallel computing, and the complexity of its own model leads to its accuracy and performance. Higher than the previously popular RNN recurrent neural network. Transformer model contains an Encoder and a Decoder. Both Encoder and Decoder contain 6 blocks.The Figure 3 show the structure of transformer model.

1) Model obtains the representation vector X of each word of the input sentence.(X contain Embedding of the word and the Embedding of the word position)

2) Model pass the obtained word representation vector matrix into the Encoder, and after 6 Encoder blocks, the encoding information matrix C of all words in the sentence can be obtained.

3) Model pass the encoding information matrix C output by the Encoder to the Decoder, and the Decoder will in turn translate the next word i+1 according to the currently translated word from 1 to i.

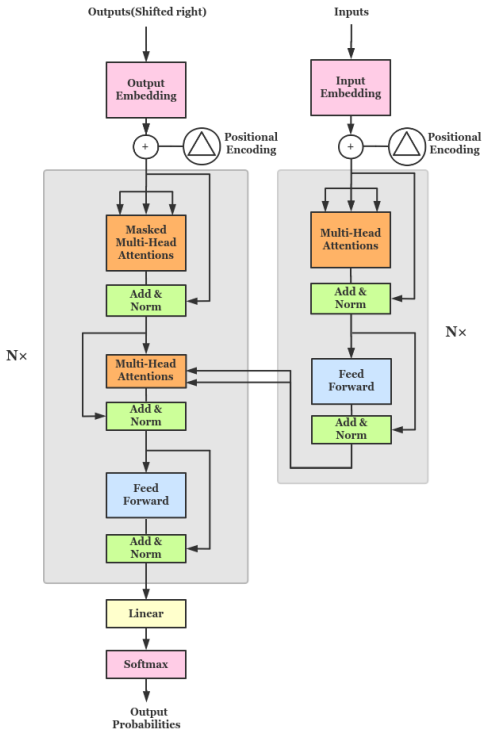


Figure 3: Structure of Transformer Model.

#### 3.2.3 Combined Model

After testing the transformer and seq2seq models, we think and build a model that mixes the two, which we call the combined model. The working principle of the model is shown in the following formula. We first obtain the seq2seq model $Output_{seq2seq}$ and the transformer model $Output_{transformer}$for the same code , and multiply the two results by the corresponding weights $W_{transformer}$ and $W_{seq2seq}$to obtain the final result.

$$Output_{combined} = max\left\{W_{transformer} \cdot Output_{transformer}, W_{seq2seq} \cdot Output_{seq2seq}\right\}$$

The weights for transformer and seq2seq model calculate by the following format using average output result for transformer $Average_{transformer}$and seq2seq$Average_{seq2seq}$ divide the sum of these two average scores output so that we can get a average weight to show how these two performance on the entire dataset.

$$W_{seq2seq} = \frac{Average_{seq2seq}}{Average_{seq2seq} + Average_{transformer}}$$

$$W_{transformer} = \frac{Average_{transformer}}{Average_{seq2seq} + Average_{transformer}}$$

In our dataset we take $W_{seq2seq}$ equal to 0.5920943071978862 and $W_{transformer}$ equal to 0.40790569280211364.

# 4 Experiment Setup

## 4.1 Research Environment

### 4.1.1 Python

This project is based on python 3.6, with Pycharm community edition for the compiler. We first utilized Java for the project, but the python language can integrate some existing libraries in a more convenient way, so we decide to use python.

### 4.1.2 Tensorflow

The models we trained like Seq2seq model and transformer model are based on a kind of deep learning framework named Tensorflow[ten] with the version of 2.0.0. Based on this python library, we can use some exiting methods to construct neural network, which make our codes of training section more concisely.

### 4.1.3 Tqdm

The training process of some deep learning models are not able to see unless we utilize some visualization tools like Tqdm[TQD]. Tqdm module can help users view the progress of our training by adding a for loop before the training process. And the version of this module is 4.64.0.

### 4.1.4 Numpy

Numpy, a very common python module, is used in the field of matrix or array calculation. The version of numpy in our project is 1.19.2[Num].

### 4.1.5 NLTK

The auto comment generation problem, which can be regarded as a translation problem between source code and natural language, needs to use some natural language processing libraries like NLTK [NLT](with the version of 3.6.7). By interating this module, we can use some existing methods to do word segment or part-of-speech tagging.

### 4.2 Training Process

After data collection and data preprocessing we start our training using tensorflow for seq2seq and transformer model.The training on training set and validation set loss and on 15 epochs for seq2seq is shown as Figure 4.And the training loss for training set and validation set on 15 epochs for transformer is shown as Figure 5

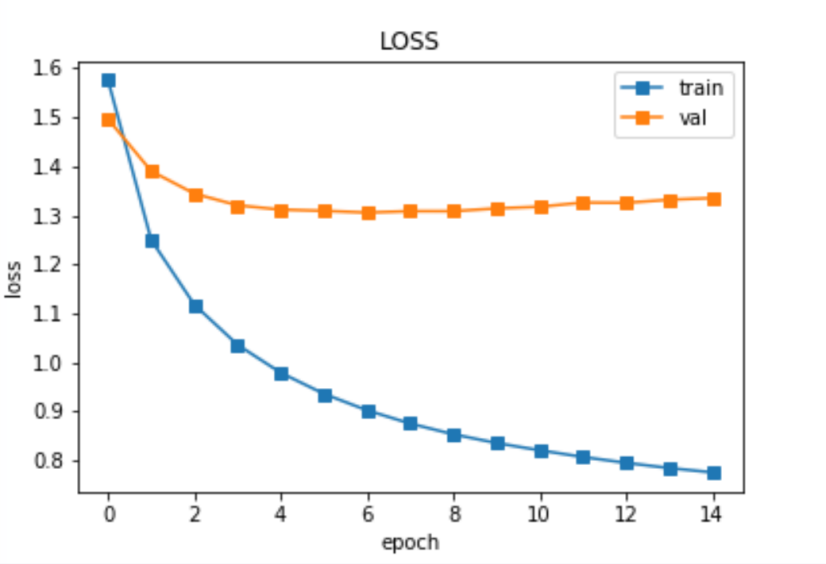


Figure 4: Training process for seq2seq on 15 epochs

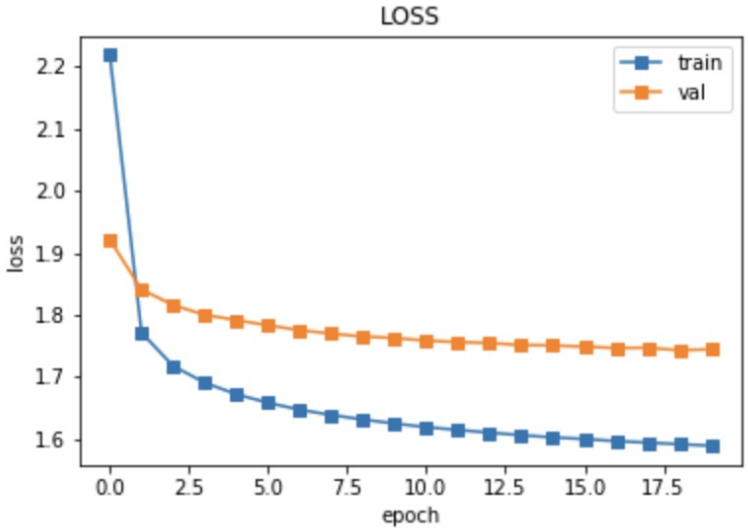


Figure 5: Training process for transformer on 15 epochs

## 5 Result

### 5.1 BLEU score result

We apply our three approach to our entire dataset and calculate the average BLEU score for 20000 comments.The following Table 1 show our result for the dataset.

| Java Code | Comments | Seq2seq | Transformer | Combined |
|---|---|---|---|---|
| public synchronized void info ( string msg )<br>{ log record record = new log record ( level . info , msg ) ;<br>log ( record ) ;<br>} | logs a info message | log the log in<br>milliseconds . | log a info<br>message | log a info<br>message . |
| public void handle gateway receiver create ( gateway receiver recv )<br>throws management exception<br>{ if ( ! is service initialised ( str_ ) ) { return ; }<br>if ( ! recv . is manual start ( ) ) { return ; }<br>create gateway receiver m bean ( recv ) ; } | handles gateway<br>receiver creation | creates a gateway<br>for a gateway . | handle a new<br>signal receiver . | handle a new<br>heartbeatlistener<br>for gps |
| public void data changed ( i data provider data provider ) ; | this method will be<br>notified by data provider<br>whenever the data changed<br>in data provider | called when a hotspot<br>exits a map from<br>a hotspot in a map as<br>a <unk>as a <unk>. | invoked when<br>the data of the<br>data<br>provider has<br>changed . | called when the<br>data of the data<br>from the storage<br>provider has changed . |
| public void range ( i hypercube space , i visit kd node visitor )<br>{ if ( root == null )<br>{ return ; }<br>root . range ( space , visitor ) ; } | locate all points within the<br>twodtree that fall within the<br>given ihypercube and visit<br>those nodes via the given<br>visitor . | make a closure for the<br>given value is a copy<br>of the first | visit the vertices<br>in the range of<br>vertices in the<br>given range . | make a closure for the<br>given value<br>is a copy of the first |
| public void add mail status ( e type , exception exception ) {<br>items . add ( new send mail status item <e >( type , exception ) ) ; } | add a new mail status to the<br>mail status list . | add a <unk>with<br>the specified action . | add a new email<br>to the group . | add a new email to<br>the group . |
| public int read raw little endian 3 2 ( ) throws io exception {<br>return buf . order ( byte order . littl e_ endian ) . read int ( ) ; } | read a bit little endian integer<br>from the stream . | reads a bit from the<br>next bits from a single<br>byte from the stream<br>as a little endian integer | read a little endian<br>integer from the<br>stream . | read a little endian<br>integer from the<br>stream . |
| public static <t extends parse object >void unpin all in background (<br>list <t >objects , delete callback callback ) {<br>parse task utils . callback on main thread async (<br>unpin all in background ( defaul t_ pin , objects ) , callback ) ; } | removes the objects and every<br>object they point to in the local<br>datastore recursively . | unpins a set of objects<br>from a set of objects that<br>are not a result . | removes the objects<br>from the local<br>storage system . | removes the objects<br>from the local<br>storage system . |

Table 1: Source code and comparison between original comment and seq2seq,tranformer and combined

### 5.2 Average score result

Through our testing on 20000 sentences we compared these three method performance and calculate the average BLEU score for these methods as shown in Table 2.

| Method | Sentences | Score |
|---|---|---|
| Seq2seq | 20000 | 17.666432 % |
| Transformer | 20000 | 17.176969% |
| Combined | 20000 | 17.670123% |

Table 2: Average BLEU score result for seq2seq,transformer and combined model .

## 6 Discussion

In this project we are confronted with many problems. To be more specific, when we aim to reimplement the paper and plug their application into the IntelliJ IDEA, the deployment of server-side and client-side is complicated. Besides, the quality of the source code has great effect on the performance of our approach, but the data of source code are complex and difficult to select. Therefore, the data is directly collected from a related paper and converted into a more specific formatted sequences.

However, we achieved some breakthroughs based on the initial paper. What we improved is that we try to use transformer and a revised model to improve the effectiveness of our approach. Besides, we use the Structure-based Traversal (SBT) method to represent our input in a regulated way.

Nevertheless, there are some limitations existing in our project. We only apply the dataset provided by the paper. Besides, the computing resources of our project are tight, which makes the training time on our own PC too long. In other words, when we want to utilize some deep learning methods, our own PCs are not available to hold that GPU computing. Therefore, we try to move the training process on HPC to implement more deep learning models.

## 7 Conclusion

This paper focus on the auto comment generation problem by regarding it as translation problem source code and natural language. The source code which works as input can be converted to ASTs sequences and then traversed to a kind of structured format by Structure-based Traversal (SBT) method. For

the model we used, we utilize Seq2seq and transformer model. However, the performance of these two model are not well so we try to integrate them. And the ensemble outperform than other models we used. For future work, we aim to integrate our program into IntelliJ IDEA as a plugin so that it could be used more conveniently. Besides, we try to compare more deep learning methods to improve the effectiveness of our models in the domain of natural language processing.

# References


[HLX+18] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. Deep code comment generation. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pages 200–20010. IEEE, 2018.

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[MM14] Paul W McBurney and Collin McMillan. Automatic documentation generation via source code summarization of method context. In *Proceedings of the 22nd International Conference on Program Comprehension*, pages 279–290, 2014.

[NLT] Nltk documentation. https://www.nltk.org/install.html.

[Num] Numpy documentation. https://numpy.org/install/.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[SPL03] Robert C Seacord, Daniel Plakosh, and Grace A Lewis. *Modernizing legacy systems: software technologies, engineering processes, and business practices*. Addison-Wesley Professional, 2003.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[ten] Tensorflow 2.0.0 installation. https://www.tensorflow.org/install/pip.

[TQD] Tqdm documentation. https://tqdm.github.io/.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.