Transformer Neural Networks as a Basis for Language Model GPT-3, and Consequences of Language Models

Nahum Damte
damte005@umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

Generative Pre-Trained Transformer 3 (GPT-3) is a natural language processing model that uses a transformer neural network to produce human-like text. This paper focuses on what exactly language models are, what GPT-3 is, how transformer neural networks works, and goes into a discussion of what it means for a computer to be able to produce human-like text and how to advance while remembering to keep ethics in mind.

Keywords: natural language processing, NLP, transformer neural network, language model, universal adversarial trigger, UAT, ethics

1 Introduction

A Turing test is a test that aims to determine whether or not a computer is capable of thinking in the same capacity that humans do. Alan Turing, in 1950, believed that in 50 years time, computers would be so good at the Turing test that an average person testing a computer would have a 70 percent chance of making the right choice after five minutes of questioning [3]. Although he was wrong, computers have certainly come a long way.

Although not quite able to pass a Turing test, Generative Pre-Trained Transformer 3 (GPT-3) is a very powerful language model capable of producing human-like text. There is an important distinction to be made: while GPT-3 can produce human-like text, this doesn't mean it's capable of thinking in the same way that humans do. The text that it outputs is the result of natural language processing (NLP) - a method that is used to 'teach' computers how to engage with natural languages, or languages spoken between humans.

This paper will discuss further on what GPT-3 is and how it works, what transformer neural networks are and how they further language modeling, and finally, what are some unintended consequences of language modeling and how best to cope with those consequences.

2 Background

To understand how GPT-3 produces human-like text, its necessary first to go over what GPT-3 is along with what are some core concepts behind it.

2.1 What is GPT-3?

GPT-3 is the third generation of generative pre-trained transformers released by OpenAI in June, 2020. GPT-3 is much larger than its predecessors. While the original GPT and GPT-2 were trained on 110 million learning parameters and 1.5 billion learning parameters respectively, GPT-3 was trained on a significantly larger 175 billion learning parameters. [3] For now, think of learning parameters as a way to compare the scale of the language models. They will be discussed in more detail later. "[GPT-3] is trained on Microsoft Azure's AI supercomputer. It is a very expensive training, estimated to have costed 12 million USD." [3] GPT-3 can be used for many different tasks including summarizing, translation, grammar correction, question answering, chat-bots, composing emails, and more. Microsoft aided in GPT-3's training and has exclusive licensing for the model, but people not affiliated with Microsoft have the ability to use the model through OpenAI's API [3].

Using GPT-3 is straightforward. OpenAI's API takes in a query that includes, among other fields, a prompt, or a sequence of words, and tries to offer a completion for the prompt. For example, if you were to give GPT-3 the prompt:

$$\text{solve for x:}$$
$$x + 40000 = 100000$$

it may respond:

$$x = 50000$$

Of course, anyone with some familiarity with basic algebra could look at this prompt and tell you that the solution should be $x = 60000$, so why did the model respond 'incorrectly?' GPT-3 is a language model that tries to generate structurally sound completions to prompts. It's helpful to think of it in this instance like a child that was raised in a room full of mathematicians. The child knows what math is and what it generally looks like, but they don't actually know how to solve problems using any kind of mathematical approach. Interestingly enough, in this case the response given was in the correct order of magnitude based on the prompt. Although the technically correct answer was $x = 60000$, as far as the model is concerned, $x = 50000$ is a sound completion.

It's important to mention that GPT-3 isn't guaranteed to give the same response to any given prompt multiple times. It's possible to tweak its potential responses to be either

```
const OpenAI = require('openai-api');

const openai = new OpenAI('[Insert API key here]');

(async () => {
    const gptResponse = await openai.complete({
        engine: 'davinci',
        prompt: 'tell me how many feet can fit in a shoe?',
        maxTokens: 256, //<- maximum tokens to be returned
        temperature: 0.9, //<- 0-1 how deterministic or creative responses are
        topP: 1,
        presencePenalty: 0,
        frequencyPenalty: 0,
        bestOf: 1,
        n: 1,
        stream: false,
        stop: ['\n', "testing"]
    });

    console.log(gptResponse.data.choices[0].text);
})();
```

Figure 1. GPT-3 OpenAI API Example Query

more objective or more creative, but it isn't possible to guarantee any given response for any prompt. Figure 1 depicts an example of a request including the prompt "**tell me how many feet fit in a shoe**." Sending the exact request to OpenAI's servers multiple times resulted in several responses including:

- *Zeyi asks me. I shake my head. "Two hundred feet," she says. "And can you see the elastic around the soleing?" We all lean in to the shoe; I can see that it's creased. "You can retie that." We all nod in recognition: we've all owned a pair of sneakers with elastic like this. Zeyi points to a piece of paper on the floor. "Is that a shoe pattern?" we ask her. She looks at us and nods. "It's my sketch," she says*
- *Answer: about 12*
- *It depends on the size of the shoe.*

Prompt responses are dependent on other parameters (not referring to the learning parameters mentioned in section 2.1) beyond the prompt itself. Different parameters have varying levels of impact on the response, so I will only go over some more important ones.

First is the 'engine' parameter, which specifies which model in the GPT-3 family to use. Davinci is the most powerful model and the one that people generally refer to when they say GPT-3. Davinci being the most powerful makes it the most expensive and slowest model to query, so other models can sometimes be found to be more helpful. The other models are Curie, Babbage, and Ada, and they each specialize in various natural language processing centered tasks including but not limited to language translation, sentiment classification, and summarization [7].

Another parameter is response length, which is measured in tokens. OpenAI defines tokens as being "common sequences of characters found in text. The models understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens. A helpful rule of thumb is that one token generally corresponds to 4 characters of text for common English text. This translates to roughly ¾ of a word (so 100 tokens = 75 words)." [7]

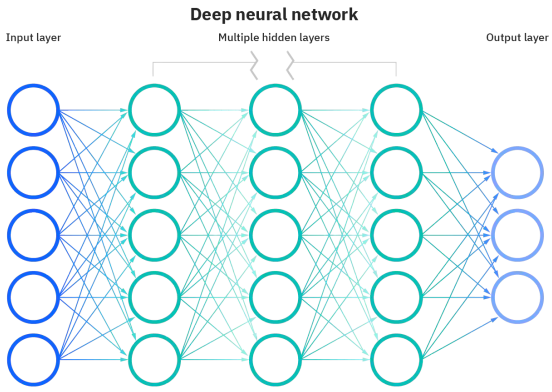

Figure 2. Neural Network Node Structure [5]

The final parameter that will be highlighted is temperature, which measures how objective/interpretive you want the response to be on a scale of 0-1, with 0 being most objective and 1 being most interpretive. One example of a use case for the temperature parameter can be found in implementing a chat bot. If you want the chat bot to respond in a very matter of fact fashion, you would set the temperature lower. If you want the chat bot to emulate some personality, you would set the temperature higher and add into the prompt that you want the responses to be sarcastic, sad, chipper, or any other personality trait(s) [7].

2.2 Natural Language Processing

Natural language processing is an interdisciplinary field combining computer science and linguistics. NLP involves the process of trying to use patterns found in natural languages to make it so that machines are able to engage with those languages. Historically, computer scientists tried to do natural language processing using rules (dictionaries and grammar) that a given natural language generally followed, but now we are able to use machine learning technique to find these patterns for us [2].

2.3 Neural Networks

Neural networks, a type of machine learning model, are series of algorithms that are designed to recognize patterns in data. Through this pattern recognition, they are able to 'learn' how to do various tasks. They are constructed via nodes that form either the input layer, the hidden layer(s), or the output layer of a neural network (see Figure 2). Each node is connected to all the nodes in the next layer and those connections are called edges.

The input layer takes inputs which are generally a vector representation of the data. The learning parameters mentioned in section 2.1 are what go into the model, and they can be described as the input values the neural network tries to learn during training.

The edges apply a weight to the inputs they receive, either from the input layer or a previous layer. Applying a weight to an input is essentially multiplying the value of the weight by the value of the input.

The hidden layers receives inputs which are the sums of the values of all their incoming nodes multiplied by the weights on their incoming edges, and apply an activation function to their inputs which introduces non-linearity to the output of a node. This allows the neural network's decision boundary to be non-linear.

Finally, the output layer returns probability distributions for each potential output, one per node in the output layer.

The basic neural network architecture is good but it lacks the ability to receive and return inputs and outputs sequentially [5].

3 Transformer Neural Network and Model Architecture

Transformer neural networks are currently the best neural network architecture that exist in service to natural language processing. There are two major aspects of transformer neural networks that make them so powerful:

- They are able to receive data that is sequential in nature with no predetermined size
- They are able to process this otherwise sequential data in parallel

See Figure 3 to see the overall transformer neural network architecture. This section will go over the layers in this network's overall architecture, as described in the paper *Attention Is All You Need* [11] along with what attention is and why it's important. This architecture isn't necessarily representative of the implementation of the transformer used in GPT-3, but it can be used to achieve various natural language processing tasks including but not limited to language modeling, question answering, machine translation, and paraphrasing. In the paper, the researchers successfully trained the model in machine translation.

3.1 Input Embedding

The input embedding layer takes tokens in from the input sequence and maps them to an embedding space based on how similar they are to other tokens in that space, similar referring to what context the tokens are found in. Figure 4 depicts three tokens: guitar, violin, and viola, as being close to each other because all of those tokens are generally used in very similar contexts. It's important to mention that while Figure 4 depicts these tokens as existing in a two dimensional space for the sake of the visualization, these tokens actually exist in multi-dimensional space.

The input embedding layer uses pre-trained embeddings. After all, this model isn't usually trained to create word embeddings since that can be done with other, simpler architectures. Additionally, since word embeddings grant meaning

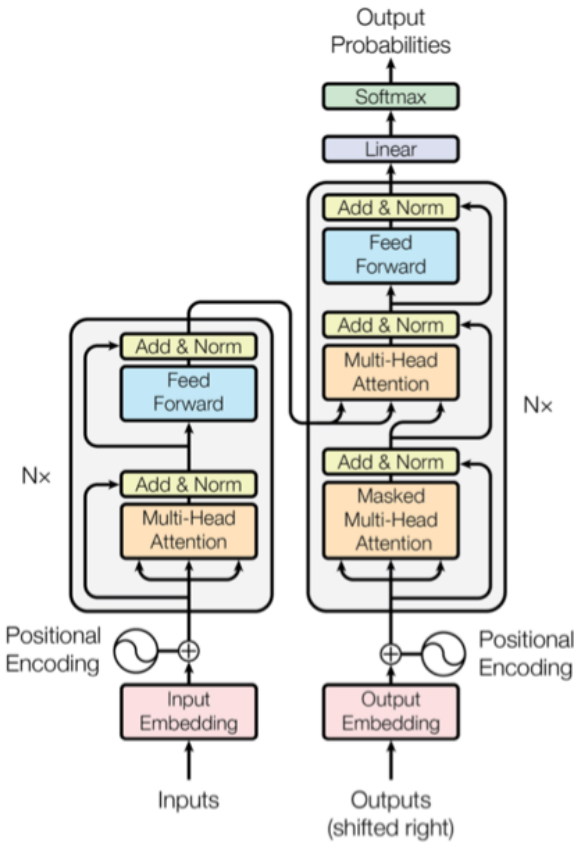

Figure 3. Transformer Neural Network Architecture [11]

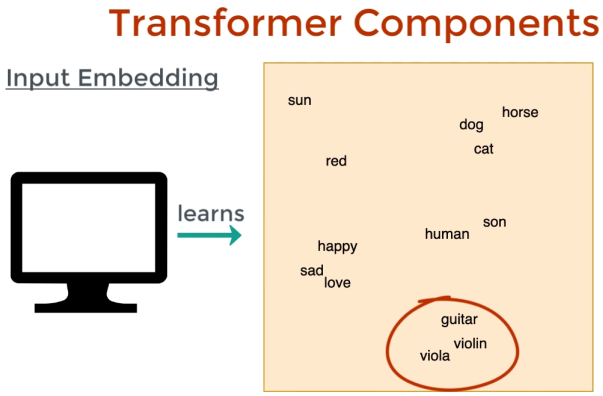

Figure 4. input embedding visualized [1]

to words, it's essential that our model has this meaning to do other NLP tasks.

3.2 Positional Encoding

Since these sequential inputs are being processed in parallel, it's important to have some way to retain positional information. The purpose of the positional encoding layer is

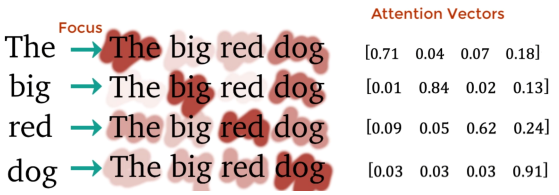

Figure 5. Multi-head attention layer attention vectors [1]

exactly that. The positional encoding layer generates vectors containing positional information for each token with the same dimensions as the previous input embedding layer, so that the two vectors can be summed. The resulting vectors contain both the meaning and the positional information of each token in the sequence.

3.3 Multi-Head Attention

After the input embedding and positional encoding layers come the first multi-head attention layer. Multi-head attention allows the model to learn how to generate different attention vectors that focus on different linguistic phenomenons. By having each attention head focus on different linguistic features, we get better predictions. The model in [11] utilized 8 attention layers.

The multi-head attention layer takes the sequence of input vectors passed in from the positional encoding layer and generates attention vectors based on those inputs. Figure 5 depicts attention vectors for the input sequence 'The big red dog' which is comprised of 4 attention vectors that show the relation between each token in the sequence to each other token in the sequence. For example, the token 'The' is shown to be the most relevant to itself, followed by the tokens 'dog,' 'red,' and 'big.'

To generate attention vectors, you must first take the input sequence vectors and represent them with query, key, and value matrices. These matrices are generated by passing each matrix of tokens through three different linear layers simultaneously, as depicted as the first step in Figure 6. The linear layers are composed of fully connected neurons without the activation function (hence linearity) and these linear layers each have different weights [6]. "The attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key." [11] The compatibility of a query with its corresponding key refers to how 'similar' a key and query is, and can be obtained by calculating the dot product between the elements of the key and query vectors. This implementation uses scaled dot product to make sure the values don't get too large.

The attention function used in this model is called **scaled dot-product attention** (see Figure 7) and can be computed using the following function:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d}}V),$$

where Q, K, and V represent the query, key, and value vectors respectively, and d represents the dimensions of the query, key, and value matrices which directly corresponds to the number of tokens in the input sequence. The T applied to the K in the formula means the key vector is to be transposed, or rotated. If a vector is a matrix with a single column, the transpose of a vector is a matrix with a single row. Finally the softmax function is as follows:

$$softmax(z) = \frac{e^z}{\sum_z e^z},$$

Where z represents each element in the vector and e represents the natural exponent. Its purpose is to take an input and represent it between zero and one, effectively giving you a probability distribution. These outputted attention vectors contain probability distributions seen in figure 5.

The masked multi-head attention sub-layer works similarly to a regular multi-head attention layer, but is made to not attend to positions following the current position. This masking, combined with fact that the output embeddings are offset by one position (the output embedding layer works similarly to the input embedding layer other than offsetting embeddings by one position), ensures that the predictions for position i can depend only on the known outputs at positions less than i [11].

3.4 Feed-Forward

After the first multi-head attention layer comes the first fully connected feed forward layer. The feed forward layer takes the resulting attention vectors from each layer in the multi-head attention layer and applies two linear transformations with the following rectified linear unit (ReLU) activation in between:

$$ReLU(\sum_{i=1}^{n} x_i w_i) = max(0, \sum_{i=1}^{n} x_i w_i)\ [11]$$

3.5 Linear Layer and Softmax Layer

The last layers involved in the transformer are the linear and softmax layer. The linear layer applies a linear transformation to the vector it is passed and the softmax layer takes the output from the linear layer and gives a probability distribution for the correctness of the outputted token-vector. If you were training a natural language model, this probability distribution would measure the model's perceived accuracy of the generated token as a part of its completion [1].

3.6 Overall Model Architecture

This model has an encoder-decoder structure. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations z = $(z_1, ..., z_n)$ that holds all the learned information of that input

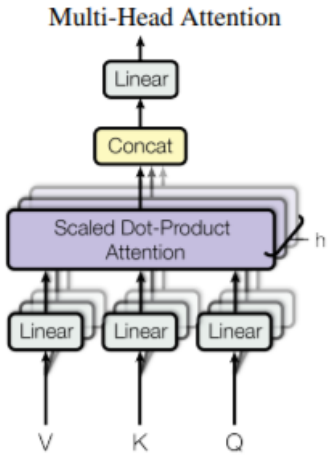

Figure 6. Multi-Head Attention [11]

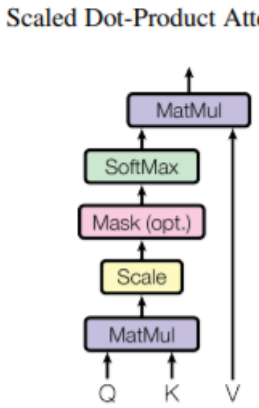

Figure 7. Scaled Dot-Product Attention [11]

[8]. Given z, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model consumes the previously generated symbols as additional input when generating the next. The benefit here is that the model is able to use tokens it has already generated as part of a completion as additional context to aid in the generation of further tokens in its completion sequence. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 3, respectively. [11].

4 Consequences

Though GPT-3 isn't exactly capable of passing a Turing test, it is still ultimately capable of producing high quality human-like text given the proper prompts and parameters. Models like GPT-3 will only get better with time, and that means that with time we will be able to produce more and more

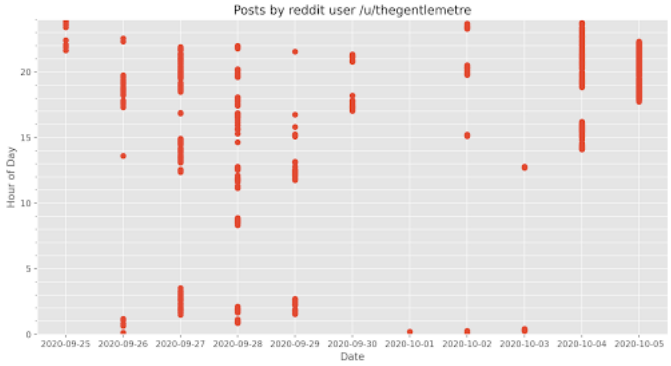

Figure 8. Reddit bot using GPT-3 [9]

high quality and cheap semantic artifacts. Translations, summaries, minutes, comments, web-pages, catalogues, newspaper articles, guides, manuals, forms to fill, reports, recipes, and still other tasks are all within the scope of GPT-3's capabilities - at least as far as drafting up these artifacts goes. It can be said that an integral skill many writers today employ, *cut & paste*, will be eclipsed by the new skill of *prompt & collate* [3].

GPT-3 and other future language processing models open the doors to a lot of opportunities to us. However, they also come with risks. Although OpenAI keeps pretty strict control of their software, OpenAI doesn't have a monopoly on language models. Eventually copycat technologies will emerge, and with them several risks.

4.1 Universal Adversarial Triggers

Universal Adversarial Triggers, a method of adversarially disrupting natural language models, are input-agnostic token sequences that, when prepended to model input, cause a natural language model to exhibit a new, adversarially defined behavior [4]. These UATs are dangerous for a few reason. For one, they pose a security risk to many pre-trained models that rely on similar architectures and data-sets to the GPT family of models. Secondly, bots on social media that are fed UATs can potentially "inflate the presence of fringe ideas online as well as trigger already deployed models that are built upon these pre-trained systems in a fashion reminiscent of the fall of Microsoft's Tay." [4] Tay was a bot released by Microsoft onto twitter back in 2016 that in less than 24 hours was tweeting out Nazi rhetoric among other obscene text and had to be taken down by Microsoft [4].

4.2 War on Bots

Another ramification of machines online producing human like text and interacting with people as if they are people themselves is that it becomes much more difficult for social media companies to distinguish between bots and humans. Right now, a method that many social media companies use to determine whether or not an account belongs to a

machine or a human is to look at its interactions. Bots interact with each other, but humans don't tend to interact with bot content very often. As of right now, bot content is usually relatively unsophisticated, so humans can generally see through them. If a cluster of accounts seem to interact with each other a lot but accounts outside of that cluster tend to avoid them, you can assume that most of the interactions happening in the cluster are fake account bot interactions.

A bot powered by GPT-3 was able to go undetected on Reddit for a little over ten days in 2020. It didn't get caught because anything about its speech came off as off to redditors as suspect, but because someone looked at its post history and realized that it would be impossible for a human being to post at this volume, as can be seen in Figure 8. After some investigation, the people of r/gpt3, a group of GPT-3 enthusiasts on Reddit, realized that the bot was being powered by a website called Philosopher AI that utilizes a GPT-3 model. They were able to come to this conclusion because some of the text that the bot was outputting seemed similar to some of Philosophy AI's text outputs. This particular event was self contained and the owner of the bot didn't seem to have any bad intentions, so nothing bad came out of the whole situation. However, a bad actor could easily use GPT-3 to make it seem like a lot of people were talking about a certain topic, or to spread misinformation [9]. Additionally, GPT-3 powered bots that post in more human fashion would be even harder to detect, as this bot was only caught due to its post history which was highly atypical for humans.

5 Conclusion

Progress is inevitable. While that may be the case, that doesn't mean that we shouldn't bother with trying to come up with counter-measures to counteract bad actors using language models. Whether that be teaching people about Universal Adversarial Triggers (they can actually be used for bot detection), teaching people about general media literacy with some added info on how to suspect you are interacting with a non-human, taking more care to make sure that our uglier biases get trained into these models to a lesser degree, or a mix of some or all those ideas, we ought to not just continue to press forward in our machine learning pursuits with reckless abandon.

Acknowledgments

I would like to thank Dr. Elena Machkasova for advising me through the writing process. This paper would not have been possible without her guidance. I would also like to thank Skatje Myers for her invaluable feedback and OpenAI for giving me access to their servers (currently in private beta) to learn more about GPT-3 and its capabilities.

References

[1] CodeEmporium. 2020. Transformer Neural Networks - EXPLAINED! (Attention is all you need). https://www.youtube.com/watch?v=TQQlZhbC5ps. Accessed: 2021-10-22.

[2] CrashCourse. 2017. Natural Language Processing: Crash Course Computer Science 36. https://www.youtube.com/watch?v=fOvTtapxa9c. Accessed: 2021-10-22.

[3] Luciano Floridi and Massimo Chiriatti. 2020. GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines* 30 (December 2020), 1–14. Issue 4. https://doi.org/10.1007/s11023-020-09548-1

[4] Hunter Scott Heidenreich and Jake Ryland Williams. 2021. The Earth Is Flat and the Sun Is Not a Star: The Susceptibility of GPT-2 to Universal Adversarial Triggers. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society* (Virtual Event, USA) *(AIES '21)*. Association for Computing Machinery, New York, NY, USA, 566–573. https://doi.org/10.1145/3461702.3462578

[5] IBM. 2020. What are Neural Networks? https://www.ibm.com/cloud/learn/neural-networks. Accessed: 2021-10-22.

[6] Hedu Math of Intelligence. 2020. Visual Guide to Transformer Neural Networks - (Episode 2) Multi-Head Self-Attention. https://www.youtube.com/watch?v=mMa2PmYJlCo. Accessed: 2021-10-22.

[7] OpenAI. 2020. Explore the OpenAI API. https://beta.openai.com/overview. Accessed: 2021-10-22.

[8] Michael Phi. 2020. Illustrated Guide to Transformers- Step by Step Explanation. https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html. Accessed: 2021-12-07.

[9] Philip. 2020. GPT-3 Bot Posed as a Human on AskReddit for a Week. https://www.kmeme.com/2020/10/gpt-3-bot-went-undetected-askreddit-for.html. Accessed: 2021-10-22.

[10] Jakob Uszkoreit. 2017. Transformer: A Novel Neural Network Architecture for Language Understanding. https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html. Accessed: 2021-10-22.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.