EE4990 Independent Study, Fall 2019 – Final Report

Student: Garrett Luskey

Advisor: Hynek Boril

Summary: This project was to use a machine learning model called a Transformer to generate Stack Overflow answers. Stack Overflow is a question forum for programmers. The Transformer model is used by Google to translate languages. What makes this model work so well is the use of self-attention with masked output to input. Self-attention is a machine learning technique introduced in 2017 used to associate context with sequenced data, also self-attention is becoming the new standard for creating machine learning models for sequential data.

To achieve this python was used with TensorFlow to create a Transformer model. Keras was used along with TensorFlow to create some of the Transformer layers. To get a training dataset Google Big Query was used for getting Stack Overflow questions and answers. TensorFlow Datasets was used to tokenize the data and creating datasets for training. All source code can be found at https://github.com/uwp-mlc/stack-overflow-answer-generator.

Study Objectives and Outcomes

My main focus was on predicted answer language and context relation to the given question using a Transformer model. The first goal for this project was to create the Transformer architecture (Figure 1) in TensorFlow. Google had a tutorial which was followed closely throughout development.

The largest issue that was run into was the converting the query result from the Stack Overflow database into a dataset provided by TensorFlow Datasets. It was discovered that the data needed to be tokenized. Tokenizers were created from the data and the data was processed using these generated tokenizers. The tokenized data was stored into a local Mongo database. However, the query of the NoSQL Mongo database was slow and using the Pickle module provided by python allowed us to store and load the data lists quickly. The tokenized question and answers were stored in separate lists associated by index. These lists could then be converted into TensorFlow datasets. These datasets where zipped together to return the input and expected value in one iteration. These datasets were padded for both question and answer. Now that a dataset was acceptable by the model, training could begin.

Running out of memory for CPU and GPU tasks was another issue that happened. This happened because the size of the data was not limited so Transformer instances were being generated with sizes of tokenizer vocabulary size times length of questions. Question length can reach sizes of 5000 tokens which would generate a Transformer instance approximately 50GB in size. These networks needed to be trained in memory so 50GB was unreasonable. To combat this the datasets where filtered given a max question and answer token size of 200. This allowed the training of 24 Transformer instances at once on a GeForce 1080ti (batch size of 24).

Training this network went smoothly after these issues were fixed. The Stack Overflow dataset contained the HTML questions and answers resulting in the network trained on questions and answers in HTML. During training the first notable language structure that was developed was the opening and closing HTML tags. The next noticeable improvement was correctly predicting the English sentence structure. These innovations happened in under two hours of training. The next improvement noticed was the contextual relation between the programming language used in the question and the programming language in the answer.

As of this point the most trained model with around 30 hours of training can identify the contextual issue in the question. In the example 1 the question is asking how to get every odd element in a list, the network gives a recommendation on what to use with “td”. The network also goes on to how to access elements in the list which relates back to what the question is asking. Note that the answer is not correct and there are some syntax errors in the provided code. However, the syntax is correct or very close to correct as example 2 shows. The only issue with syntax is a missing closing parenthesis and double quote on the final Uri.parse statement. The network struggles more with long questions as training only prepared the network for a max of 200 tokens. In example 3, the predicted answer shows repeating output which happens most of the time for long questions and rarely for shorter questions.

Notable achievements for this network are the ability to generate links as in example 4. Another is the contextual relation with the question. The network “answers” the question using contextual clues and may recommend some tool or technique that is has seen in the past like in example 5 with “dplyr”. The network recommended a JavaScript package called dplyr which is a grammar used for data manipulation. The question is asking how to use a specific function to manipulate a dataset, the network related the word “dataset” to “dplyr”. The network used the “knowledge” from training to suggest a tool that relates to datasets.

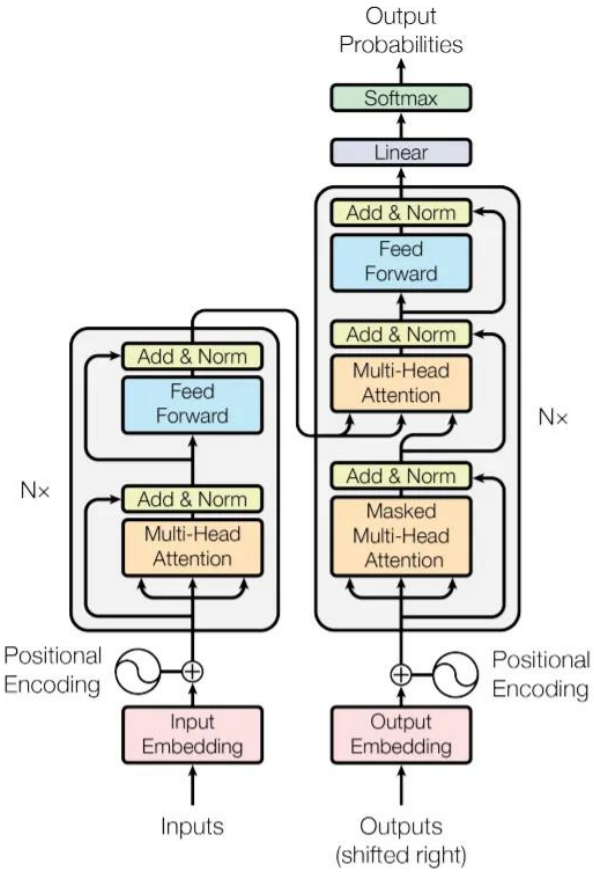

Figure 1: Transformer model architecture

Example 1: Question

https://stackoverflow.com/questions/16835825/nth-child2n-of-attribute-value

Example 1: Predicted answer

You can use <code>td</code> to get the element with the element you want to be able to get the element with the class.

<pre><code>td[class="row-row">> td[class="row-row"]').text();

</code></pre>

You can also use <code>td[class="row-row"]</code> to get the element with the element you want.

Example 2: Question

https://stackoverflow.com/questions/16600156/fragment-in-viewpager-not-restored-after-popbackstack

Example 2: Answer

You can use the following code to get the activity from the activity:

<pre><code>Intent intent = new Intent(this, R.id.activity);

intent.putExtra("android:id="@+id/activity");

startActivity(intent);

</code></pre>

This will return the activity:

<pre><code>Intent intent = new Intent(android.content.Intent.ACTION\&undscVIEW,
Uri.parse("android.content.html"), Uri.parse("android.content.Intent.ACTION\&undscVIEW);

startActivity(intent);

</code></pre>

Example 3: Question

https://stackoverflow.com/questions/57618005/print-scrollable-panel-which-more-than-two-or-more-pages-containing-pictureboxes

Example 3: Answer

<p>You can use
<code>System.Windows.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.For
ms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Form
s.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.
Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.F
orms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.For
ms.Forms.Forms.Forms.Forms.Forms.Forms.Forms.forms.forms.forms.forms.forms.forms.forms.forms.f
orms.forms.forms.forms.forms.forms

Example 4: Question

https://stackoverflow.com/questions/42984621/api-working-fine-in-postman-but-not-in-angular2

Example 4: Answer

<p>I solved this by adding a new link to the Google Chrome SDK and adding the following link to the Google Chrome SDK and it worked.</p>

<p>https://developers.google.com/chrome/extensions/chrome-sdk</p>

Example 5: Question

https://stackoverflow.com/questions/42894970/how-do-i-change-na-to-0-with-lapply

Example 5: Answer

<p>You can use <code>dplyr</code> to get the index of the element in the list:</p>

<pre><code>df$list <- data.frame(x=1:10)

df$list <- data.frame(x=1:10)

df$list <- data.frame(x=1:10)

df$list <- data.frame(x=1:10)

df$list <- data.frame(x=x,y=10)

df$list <- data.frame(x=x,y=x=y,y=y=y)

df$list[x=x:10]

</code></pre>

Hands-On Experience

A bullet list of techniques and algorithms you worked on/with, as well as software tools you used; put each technique, algorithm or software tool as a separate bullet.

- Python
- MongoDB
- TensorFlow
- TensorFlow Datasets
- Keras
- GPU-Computing
- Git

Conclusion and Future Work

This project has taught me a lot. First on how the Transformer network functions on a black box level. Also, I have learned some concepts of self-attention like attention heads and a general idea on how self-attention works. One takeaway from this project would be the natural language processing techniques used, such as, generating tokenizers from a corpus and using tokenizers to create model acceptable data. The largest takeaway is the TensorFlow Datasets tool and how easy and powerful it can be. In the future I see this project being expanded using higher end equipment and a larger input for the network. In conclusion, the results shown by the network were impressive and I wonder if with enough training and size, this network could generate helpful answers.