Capstone Project 2
Milestone Report
6/29/2020

"Audio Moderator: A Speech to Translation Pipeline using LSTM and seq2seq"

## Project Recap:

This project aims to translate Chinese audio to English text to assist in content moderation and foreign language audio/video sharing.

## The Problem:

Most platforms that allow users to express themselves and share their own content have safeguards in place to monitor for abusive behaviors. For example, they may use machine learning to detect written hate speech or offensive images. However, there are two issues that make content moderation more difficult for these platforms: non-English language content, as well as audio, video, and steaming content. Under these conditions, the platform often has to rely on a great number of human content moderators. This slows the detection process, and the content then has the potential to harm many more people before it is removed.

This project tackles both the issues of non-English and audio content by building a pipeline to translate foreign language audio input into English text.

## The Client:

This application is relevant to any social media platform with a global audience that allows users to post audio or video content. In particular, it is useful to companies that employ thousands of human content moderators, such as Facebook, YouTube, and Twitter, because it can save the moderators hours of having to sit through a video to determine if the content is allowable. It is also useful for other video-centric platforms with potentials for abuse, such as TikTok and Twitch. And although content moderation is an obvious implementation of a speech to translation application, there is also promise for use outside of the abuse space, such as achieving wider audiences for foreign language films on video streaming services like Netflix by offering automatic subtitles in a target language, or allowing podcasters to publish translated transcripts of their talks.

The Data:

The comes from the Tatoeba Project bilingual sentence pairs, available here. The 22,075 sentence pairs in the Chinese-English set have mostly short, but varied lengths, and no specialist vocabulary. The data comes in a tab-delineated .txt format consisting of three parts: and English sentence, the equivalent sentence written in simplified or traditional Chinese, and an attribution to the human translator on the project website. An example looks like the following:

"Show me an example. 给我个例子。 CC-BY 2.0 (France) Attribution: tatoeba.org #395402 (CK) & #490074 (fucongcong)"

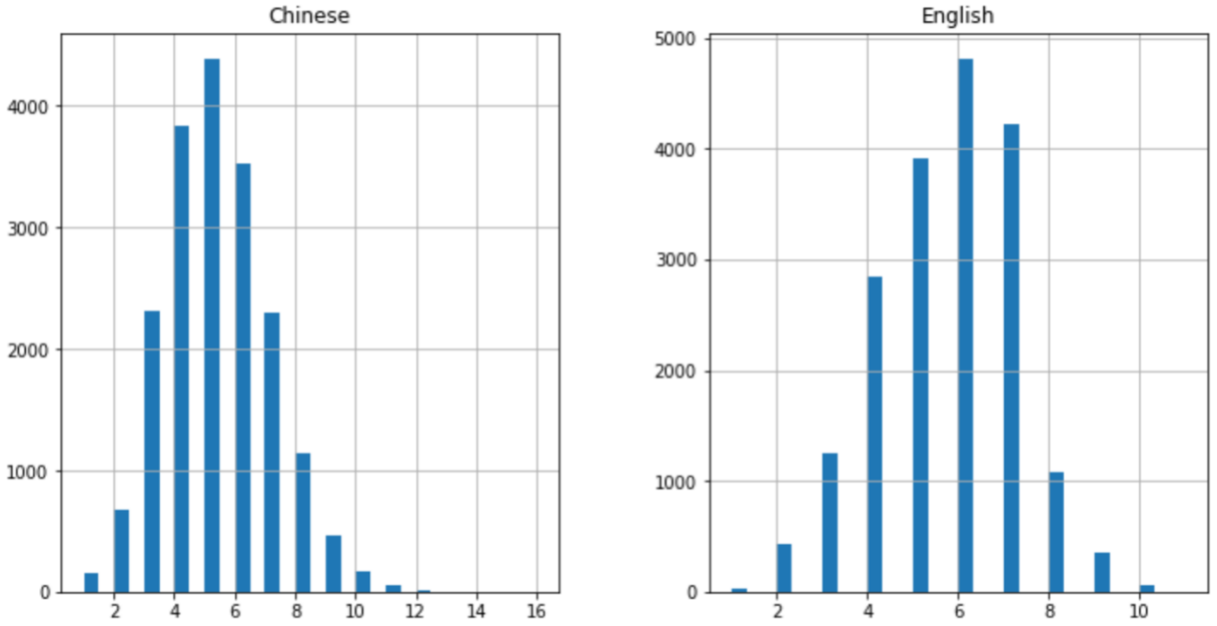
Data Wrangling Steps:

To clean the data, I dealt with each language individually. For the English sentence cleaning, I used string methods to lowercase all words and removed punctuation with regular expressions. For the Chinese, I used a separate library called Jieba to clean the data. Chinese is written without spaces, so each sentence appears as a single string. Jieba cuts Chinese sentences at word boundaries and separates the 1, 2, or 3+ character words with spaces. This enables us to tokenize Chinese as we would for languages written with alphabets. I also cleaned out Chinese punctuation using regular expressions, removed the attribution section, and then tokenized each language using Keras tokenizer. Unlike in other NLP tasks where we might perform lemmatization, stemming, or stop word removal to decrease noise in the data, for machine translation, we will leave these features in. Humans use stop words in speech, so our model needs to learn them too!

Exploration and Visualization:

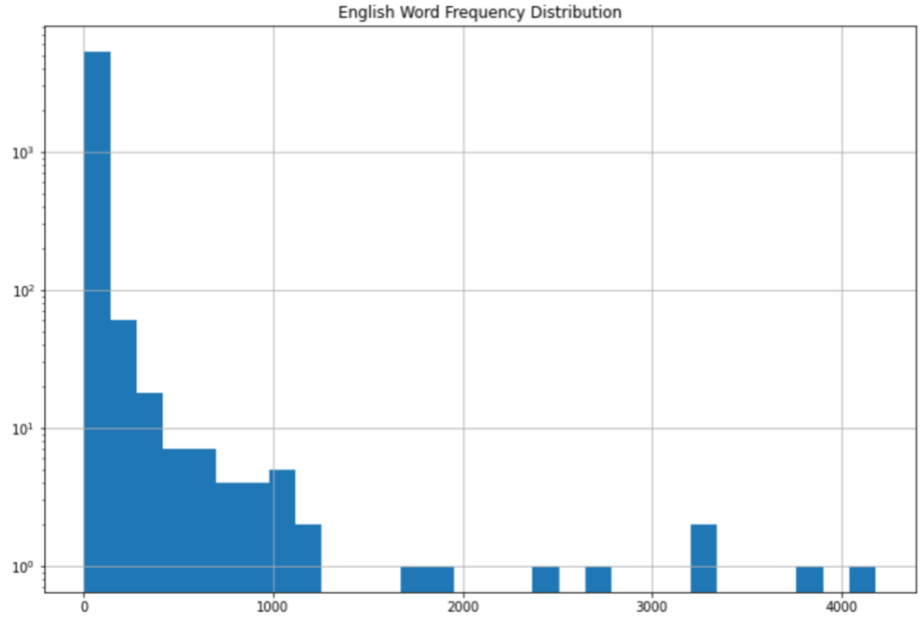
To explore the data, I performed some descriptive statistics on the tokenized data to look into sentence length and word frequency distributions. In the subset of the data I have used so far (19,000 sentences), the Chinese sentences have a vocabulary over double the size of the English vocabulary, 10,369 unique Chinese words as compared to 5,449 unique English words, and they also have a longer maximum sentence length, with 16 words, as compared to 11. On average, Chinese sentences had 5.2 words, and English sentences had 5.6 words.

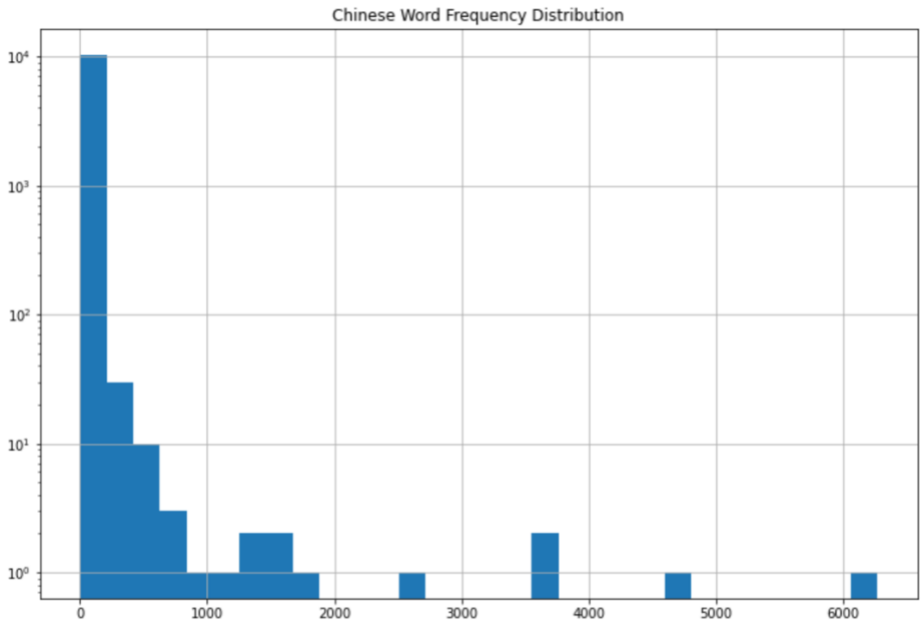
Histogram of Sentence Lengths for Dataset of 19000 Sentences




As for word frequencies, 56% of the unique Chinese vocabulary consisted of words that only occurred once in the data, whereas 40% of the unique English vocabulary was made up of words that only appeared once. For both languages, there were more than 100 words that appeared more than 100 times.

Histograms of Words Frequencies for Chinese and English Words






The data was split into training and testing sets, with a larger proportion of the data than usual assigned to the training, at least for the model building and tuning stages of this project. This was to try to slightly reduce the amount of unseen vocabulary in the testing set. Then the texts were vectorized, and the vectors were padded with zeros on the end to ensure they were all the same length.

Model Selection and Tuning:

On the advice of my mentor, I selected an LSTM encoder-decoder model, and built the model using Keras with TensorFlow backend in Google Colaboratory, in order to take advantage of their free GPU. LSTM was a natural choice because it can remember previous information from a sequence and use it construct the new sequence. The first try produced a translation of English “the” for every Chinese sentence, so it was clear there was a lot of tuning to do.

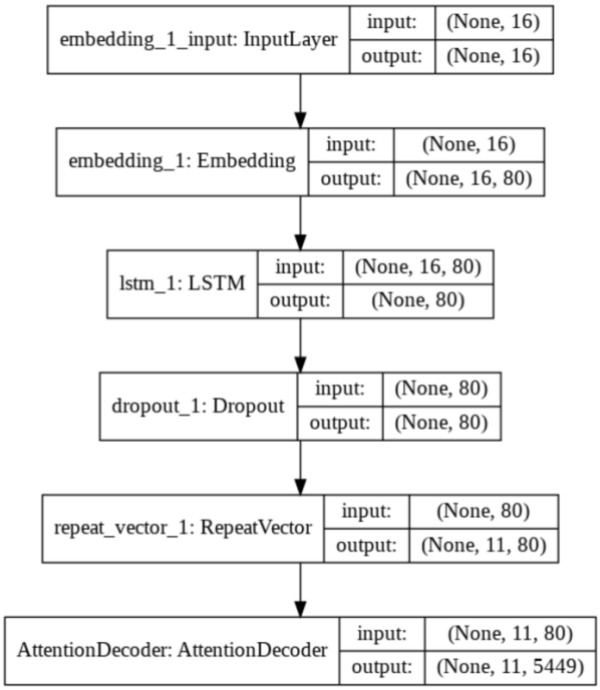
First, I changed the LSTM decoder layer in the model to an Attention decoder. Keras does not currently have a built in Attention layer, so I used a custom one from the Datalogue Keras attention project. Attention is important in Machine translation because it allows the decoder layer to pay further attention to the output sequences of the encoding layer, allowing it to deal with input and output sequences with difference in length, especially long outputs, and pay further attention to the context from the encoding output to produce the decoded sequence. This has the potential to decrease word repetitions.

Next, I also added a dropout layer after the LSTM encoding layer, to prevent overfitting. Dropout allows only a random set of neurons at a specified rate to contribute to the model training in each pass. In other words, it drops a certain percentage of neurons randomly. This contributes to a model better at generalization because it is overall less reliant on neuron weightings that are specific to the training data. Additionally, I also adjusted the number of

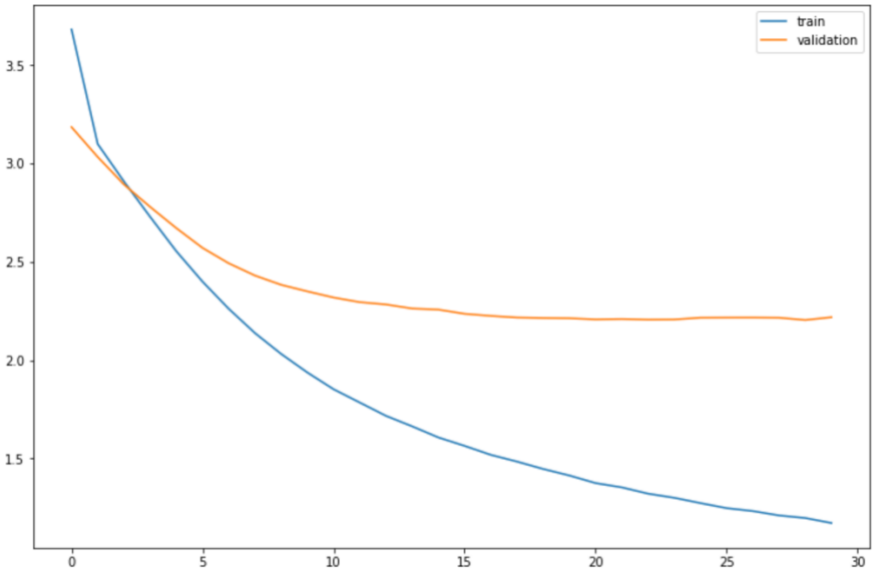
neurons in the layer and adjusted the learning rate in the optimizer function in order to configure how quickly and significantly the model weights were updated.

Although there is still further tuning to be done to this model, the current translations are significantly better than the first round, especially for the training data.

Model Plot of LSTM with Dropout Layer and Attention Decoder




Visualization of Model Training over 30 Epochs



Model Scoring:

Machine Translation quality is often evaluated using BLEU scores, so I have used this method to score my model as well. BLEU scoring follows a string-matching algorithm and compares matching ngrams in the predicted translation with human-translated target sentences. String matching cannot account for strings that are not identical but still semantically related, such as the same verb with different tenses, or synonyms. Additionally, human translators often produce a variety of translations for a single source sentence. Because difference in translations can occur, a BLEU score between 60% to 70% is generally thought to be good. The highest score my model has received so far is 54.3% for unigrams on the training data. With more tuning I hope to improve this.

Some good translations:

```
src=[巴西 的 首都 是 巴西利亚], target=[The capital of Brazil is Brasilia], predicted=[the capital of brazil is brasilia]
src=[他 7 点到 了 火车站], target=[He arrived at the station at seven], predicted=[he arrived at the station at seven]
src=[你 的 父母 有 多 高], target=[How tall are your parents], predicted=[how tall is your parents]
```

Some not-so-good translations:

```
src=[那本书 很 旧], target=[That book is very old], predicted=[that house is small]
src=[我 被 蚊子 叮 了], target=[I was bitten by a mosquito], predicted=[i was drowned annoyed]
src=[汤姆 是 这个 班级 里 唯一 的 男生], target=[Tom is the only guy in this class], predicted=[tom is only guy guy in class class]
```

Identified problems:

The problem of overfitting is a work in progress that will require more research and patience with configuring parameters to optimize the model. However, there are a couple other issues that might also contribute to a less-than-optimal performance: poor segmenting of the Chinese data before tokenization, and the high percentages of words that only occur once in the dataset.

Although most of the Jieba segmenting seemed to be cut on word boundaries, occasionally incorrectly cut segments appeared. For example, the sentence “我不喜歡洋蔥的味道” (“I don’t like the taste of onions”) was split incorrectly at the verb. The character sequence 不喜歡洋蔥 (don’t like onions) should have three segments 不 (don’t) 喜歡 (like) 洋蔥 (onions), corresponding with the negated verb and its object, but instead it was segmented into two parts, 不喜 and 歡洋蔥, splitting the verb. This happened both with simplified and traditional scripts. After simplifying all characters, this particular error did not occur, but there was still some mis-segmentation, such as 队会赢 (“team will win”) as a single segment, when it should probably be three individual ones: 队 (team) 会 (will) 赢 (win). Mis-segmenting can lead to treating certain parts of the sentence as new, unseen strings, which can produce errors in translation. For example, the “onions” in this sentence were often translated incorrectly. To solve this problem, I can experiment with different Chinese NLP libraries to perform segmentation, or I can segment by individual characters for tokenization. This splits words that are more than one character long, but maintains the correct order, and more importantly does not produce segments that are mistakes.

The other main issue is the high frequency of words that only occur once in the dataset. The words cause problems, especially when they appear in the testing data. Having not seen them before, the model will not be able to make a prediction, and instead will omit words or default to more frequent words, leading to sentences like “let's take a the on the sky” for 讓我們在公園裡散步吧 (Let's take a walk in the park). I think splitting characters individually may also help with this issue, as will converting all traditional characters to simplified (or vice versa). These methods will reduce vocabulary overall, since there will be less character combinations from n-grams, and no repeated vocabulary differing by script.

Remaining Steps:

In addition to testing different preprocessing on the Chinese data, I still need to add the speech-to-text step. And finally, I need to clean up the mark up in the notebook and rearrange some of the code cells to make everything a bit neater.