Capstone Project 2

Milestone Report 1

Model Report

For

Translations Services Inc (TSI)

Executive Summary

The European Parliament conducts business in many languages due to the huge differences in languages and cultures that make up its members. The transcriptions of its meetings need to be translated into a multitude of languages. One of these is French.

Translations Services Inc (TSI) has been contracted by the European Parliament to translate conference transcriptions from English to French on a continual basis. Due to the overwhelming number of documents to translate, TSI hired me as their Data Scientist to build a Natural Language Processing (NLP) model to automatically translate the documents from English to French.

The data that was used for this project consists of European Parliament provided transcripts of sessions that are organized into two files: europarl-v7.fr-en.en (English sentences from the transcripts) and europarl-v7.fr-en.fr (French sentences from the transcripts). Each line/sentence in the English file corresponds to its translation in the French file. This data set can be found on the LionBridge website: http://www.statmt.org/europarl/

This report will show the steps I took to gather the input data, load it, and manipulate it to prepare it for fitting an NLP model.

1. Environment and Data Set Up

1.1) Environment Set Up

For this project, since I knew I was going to need more processing power than my personal laptop could provide, I signed up for a Google Cloud Computing account. In doing so, you are given a $300.00 credit to use for testing purposes before you will be charged for their services. It is a good amount to start learning about Google Cloud Computing and how to set up a project there.

I took advantage of this opportunity and set up an instance with a decent CPU and added an NVIDIA K80 GPU for more processing power. The instance runs on a Linux DEBIAN Operating System.

I also built a Google Cloud bucket where I could upload my language input files.

1.2) Data Load Set Up

I first had to install the Cloud Storage Client Python Library so that I could import cloud storage and initialize the client. With the client I was able to create a bucket object (Google Cloud object that holds data files) in which I stored the bucket

name for the bucket I created on the Google Cloud Platform and where I uploaded my input files.

With the bucket object, I was able to generate a couple blob objects (Binary Large Objects), one for each input file. From the blob objects, I was able to generate bytes objects and finally string objects, each representing newline separated sentences in each language file.

At this point I was able to take a look at a few English sentences along with their French counterparts to make sure the data had loaded properly. Since this NLP project is about translating from one language to the next, a lot of the NLP techniques such as lemmatization and stemming were not used.

2) Processing the Data

2.1) Subsetting the Data

I initially attempted to use all the records from the English and French Input language files, but I quickly realized that the files were too large for even the Google Cloud Compute instance I spawned to handle. I had to resort to subsetting the files to a fraction of the available records. Each record consists of a sentence. I had to subset my language files to 6000 sentences each.

2.2) Tokenizing

After subsetting the data, I built a tokenize function to tokenize the sentences. The tokenize function takes in one parameter: a list of sentences to tokenize. The function returns the tokenized sentences and the tokenizer that was used.

2.3) Padding

After tokenizing the data, I built a pad function to pad the tokens. The pad function takes in two parameters: a list of sequences (or in this case, tokens) and a length to pad the sequences to. The function returns a padded numpy array of sequences.

2.4) Preprocess Pipeline

After padding the data, I built a preprocess pipeline function that takes in two parameters: a list of English sentences and a list of French sentences. The function tokenizes the English and French lists and then pads them. The function returns a preprocessed English list, a preprocessed French list, an English tokenizer and a French tokenizer.

Summary

To conclude, I used a Google Cloud Platform account as my set up for modeling. I uploaded the two language data files to a Google Bucket on the Platform. I then pulled those files in through my code and loaded the data in its entirety before subsetting it to a more manageable chunk of data for the processing power at my disposition. With the subset of the data, I then tokenized the sentences in each language and padded the tokens. The data is now in a ready state for modeling.