Neural Machine Translation

Supervised By:
Periklis A. Papakonstantinou
Co-Supervised By:
Nathaniel Hobbs
Submitted By:
Deepti Khatri

Contents

ABSTRACT

PROBLEM

Deployment of the model on a website, where user input text in source language and gets the output translated in the target language.

METHODS USED

We have used python to train our neural machine translation model, therefore using the same for developing the website. Flask is a python based microframework used to develop small scale website.

HTML and CSS is used for front end development.

Once the flask app is ready Heroku is used for its deployment. Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud. In this project we deploy using Heroku-git.

RESULTS OBTAINED

A website where user can get translation for Hindi – English and French – English pair. The output is 3 different translation from different models. Different model mean different parameter used while training the model – different learning rate, different embedding layes.

LANGUAGES USED

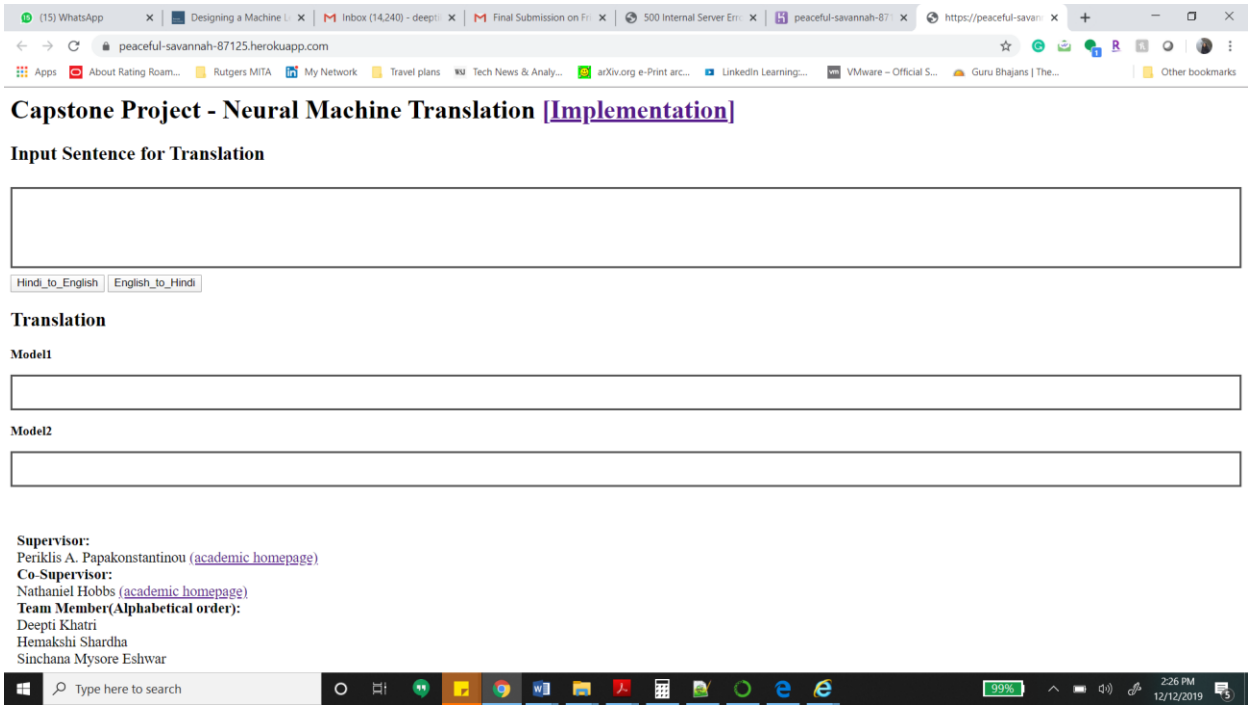
Python, Flask, HTML, CSS, Heroku and Heroku-git.

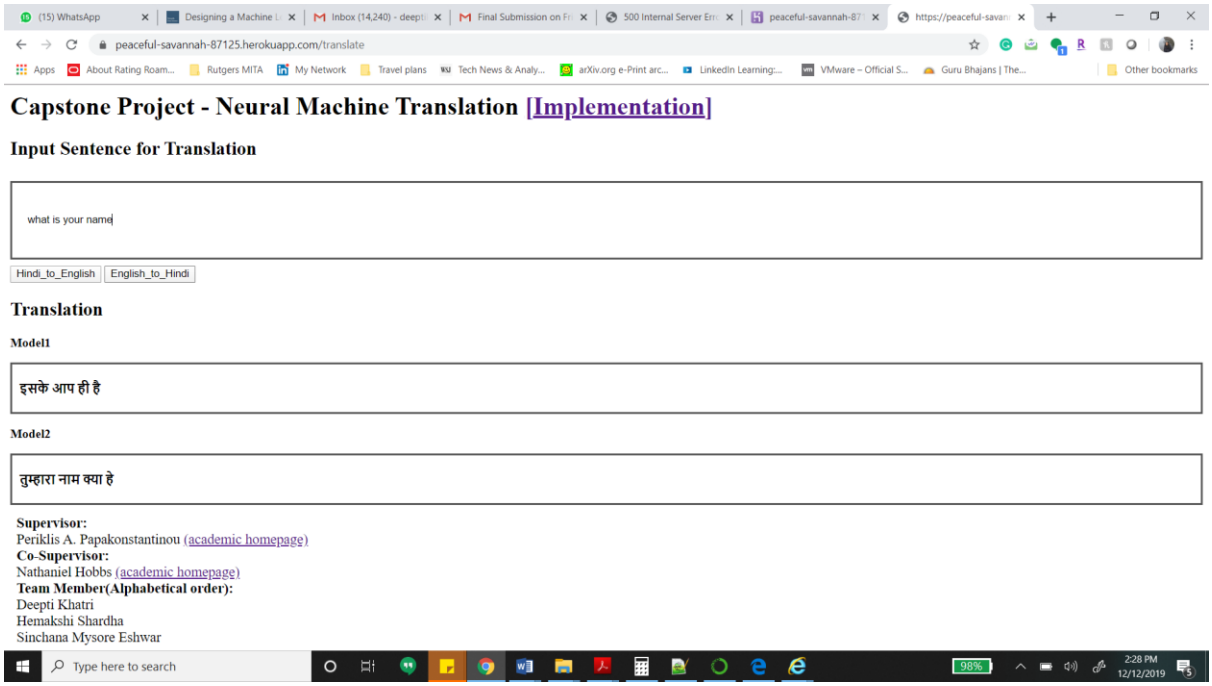
RESULTS AND DISCUSSION

RESULTS

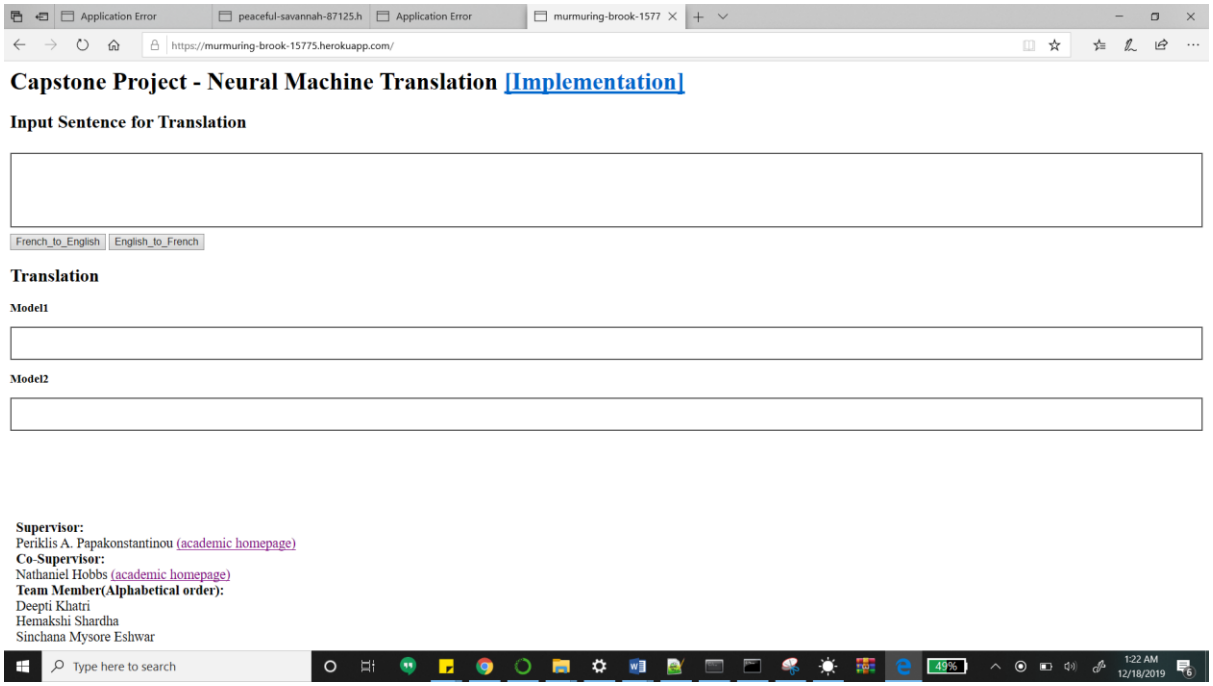
Following are the snapshots of the website Front End: -

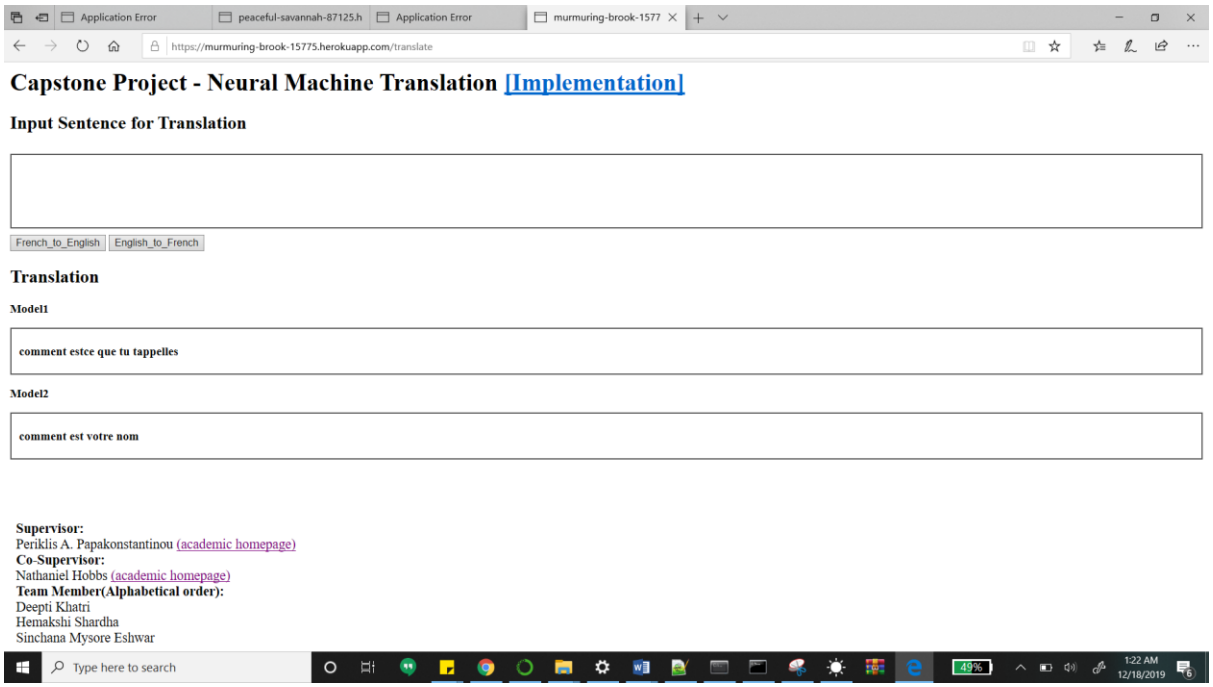
https://peaceful-savannah-87125.herokuapp.com





https://murmuring-brook-15775.herokuapp.com/





Following are the steps to make Flask app: -

HTML source code: -

```
<html>
<head>
<style>
input[type=text] {
  width: 100%;
  padding: 40px 20px;
  margin: 8px 0;
  box-sizing: border-box;
  border: 2px solid #555;
  outline: none;
}
.bottomleft {
  position: absolute;
  bottom: 8px;
  left: 16px;
  font-size: 18px;
}
</style>
</head>
<body>
  <h1> Capstone Project - Neural Machine Translation
  <a href="https://github.com/khatrideepti/Neural-Machine-Translation---LSTM">[Implementation]</a></h1>
  <div>
  <form action="/translate" method="POST">
    <h2>Input Sentence for Translation</h2>
    <input type="text" id="french" name="french">
                <input type="submit" name="bsubmit" id="bsubmit" value="Hindi_to_English">
                <input type="submit" name="bsubmit" id="bsubmit" value="English_to_Hindi">
  </form>
                <h2> Translation </h2>
                <p><b>Model1</b></p>
                <h4>{{ prediction1 }}</h4>
                <p><b>Model2</b></p>
                <h4>{{ prediction2 }}</h4>
                <br>
<div class="bottomleft"><b>Supervisor:</b>
                                        <br>Periklis A. Papakonstantinou
                                        <a href="http://papakonstantinou.org/periklis/">(academic homepage)</a></h1>
                                        <br>
                                        <b>Co-Supervisor:</b>
                                        <br>Nathaniel Hobbs
                                        <a href="http://www.nathanielhobbs.com/">(academic homepage)</a></h1>
                                        <br>
                                        <b>Team Member(Alphabetical order):</b>
                                        <br> Deepti Khatri
                                        <br>Hemakshi Shardha
                                        <br>Sinchana Mysore Eshwar
</div>
</div>
</body>
</html>
```

Flask Script

Before starting with the coding part, we need to download flask and some other libraries.

```
pip install flask
```

Create folder **templates**. In application, templates is used to render HTML which will display in the user's browser. This folder contains html form file *index.html.*

```
mkdir templates
```

Create script.py which has following code.

```
#importing libraries
import flask
from flask import Flask, request, render_template
import pickle
from keras.models import load_model
import keras.backend as K
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
import os, sys
import string
import pandas as pd
import re
import numpy as np
import tensorflow as tf

#creating instance of the class
app = Flask(__name__)
#app.debug = True
#app.config['SECRET_KEY'] = 'letswrapup'
#toolbar = DebugToolbarExtension(app)

def softmax_over_time(x):
  assert(K.ndim(x) > 2)
  e = K.exp(x - K.max(x, axis=1, keepdims=True))
  s = K.sum(e, axis=1, keepdims=True)
  return e / s


LATENT_DIM_DECODER = 256
t=0
max_len_target=18
```

```
class NMT:
    def __init__(self, encoder, decoder, source_file, target_file, input_tokenizer  ):

        self.hi_encoder=load_model(encoder,custom_objects = {"softmax_over_time": softmax_over_time, "t":t})
        self.hi_decoder=load_model(decoder,custom_objects = {"softmax_over_time": softmax_over_time, "t":t})
        with open(source_file, 'rb') as handle:
            self.idx2word_eng = pickle.load(handle)
        with open(target_file, 'rb') as handle:
            self.idx2word_trans = pickle.load(handle)
        with open(input_tokenizer, 'rb') as handle:
            self.tokenizer_inputs = pickle.load(handle)
        self.word2idx_inputs = {k:v for v, k in self.idx2word_eng.items()}
        self.word2idx_outputs = {k:v for v, k in self.idx2word_trans.items()}

    def decode_sequence(self,input_seq,maxlen):
        #print("Sentence:",input_seq)
        input_seq=input_seq.lower()
        input_seq=input_seq.split(' ')
        #print("input_seq:",input_seq)
        token_input_seq=[]
        for word in input_seq:
            idx = self.tokenizer_inputs[word]
         #   print(idx)
            token_input_seq.append(idx)
        #print("Tokenized sentence:",token_input_seq)


        input_seq_final= pad_sequences([token_input_seq], maxlen=maxlen)
        #print("paddded sentence:",input_seq_final)

        enc_out = self.hi_encoder.predict(input_seq_final)
        # Generate empty target sequence of length 1.
        target_seq = np.zeros((1, 1))

        # Populate the first character of target sequence with the start character.
        # NOTE: tokenizer lower-cases all words
        target_seq[0, 0] = self.word2idx_outputs['<sos>']

        # if we get this we break
        eos = self.word2idx_outputs['<eos>']


        # [s, c] will be updated in each loop iteration
        s = np.zeros((1, LATENT_DIM_DECODER))
        c = np.zeros((1, LATENT_DIM_DECODER))


        # Create the translation
        output_sentence = []
        for _ in range(max_len_target):
            o, s, c = self.hi_decoder.predict([target_seq, enc_out, s, c])

            # Get next word
            idx = np.argmax(o.flatten())
         #   print(idx)
            # End sentence of EOS
            if eos == idx:
                break

            word = ''

            if idx > 0:
                word = self.idx2word_trans[idx]
          #      print(word)
                output_sentence.append(word)

            # Update the decoder input
            # which is just the word just generated
            target_seq[0, 0] = idx

        return ' '.join(output_sentence)
```

```
fe1=NMT('f_e_encoder_model.h5','f_e_decoder_model.h5','f_e_idx2word_eng.pickle','f_e_idx2word_trans.pickle','f_e_tokenizer.pkl')
fe2=NMT('f_e_encoder_model2.h5','f_e_decoder_model2.h5','f_e_idx2word_eng.pickle','f_e_idx2word_trans.pickle','f_e_tokenizer.pkl')
ef1=NMT('e_f_encoder.h5','e_f_decoder.h5','e_f_idx2word_eng_1.pickle','e_f_idx2word_trans_1.pickle','e_f_tokenizer.pkl')
ef2=NMT('e_f_encoder.h5','e_f_decoder.h5','e_f_idx2word_eng_1.pickle','e_f_idx2word_trans_1.pickle','e_f_tokenizer.pkl')



graph = tf.get_default_graph()
@app.route('/')
def index():
        return flask.render_template('index.html')

@app.route('/translate',methods =['GET', 'POST'])
def translate():
        try:
        global graph
                with graph.as_default():
                        if request.form.get('bsubmit')=='French_to_English':
                                input_seq = request.form['french']
                                result1 = fe1.decode_sequence(input_seq,9)
                                result2 = fe2.decode_sequence(input_seq,9)
                                return render_template('index.html',prediction1=result1, prediction2=result2)
                        elif request.form.get('bsubmit')=='English_to_French':
                                input_seq = request.form['french']
                                result1 = ef1.decode_sequence(input_seq,8)
                                result2 = ef2.decode_sequence(input_seq,8)
                                return render_template('index.html',prediction1=result1, prediction2=result2)
        except:
                result="Our model encountered some unknown words in the given sentence. We are still trying to improve it."
                return render_template('index.html',prediction=result)
```

Here we import the libraries, then using app=Flask(__name__) we create an instance of flask. @app.route('/') is used to tell flask what url should trigger the function index() and in the function index we use render_template('index.html') to display the script *index.html* in the browser.

Let's run the application

```
python script.py
```

This should run the application and launch a simple server. Open http://127.0.0.1:5000/ to see the html form.

Index.html is displayed and when user enters a input sequence and click on French_to_English/English_to_French button, the model French_to_English/English_to_French object fe1 and fe2/ef1 and ef2 as mentioned in the script.py file is called and translation is generated and returned under the Translation heading within respective text boxes.

DEPLOYING FLASK APP USING HEROKU

Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud. In this project we deploy using Heroku git.

We need to install git as well as Heroku CLI.

Following the steps to deploy flask app and make a website: -

Step1

Install **gunicorn** in the python environment**.**

```
pip install gunicorn
```

Step2

In our local machine, we have installed a lot of libraries and other important files like flask, gunicorn, sklearn, tensorflow, keras etc. We need to tell heroku that our project requires all these libraries to successfully run the application. This is done by creating a **requirements.txt** file.

```
pip freeze > requirement.txt
```

Step3

Procfile is a text file in the root directory of your application, to explicitly declare what command should be executed to start your app. This is an essential requirement for heroku**.**

```
web: gunicorn script:app
```

This file tells heroku we want to use the web process with the command gunicorn and the app name.

Step4

In our project folder we have a lot of hidden or unnecessary files which we do not want to deploy to heroku. For example venv, instance folders or .cache files. In order to not include them we create a .gitignore file.

```
*.pyc
__pycache__/

instance/

.pytest_cache/
.coverage
htmlcov/

dist/
build/
*.egg-info/

.DS_Store
```

Now the project folder has all the necessary files and model weights, which are ready to be pushed to Heroku.

Step5

Open terminal and login to Heroku

```
heroku login
```

This will open a Heroku website in web browser and needs login details.

Step6

Now create Heroku app using below cmd

```
heroku create
```

App with a random name gets created

Step7

Using below commands to initialize git repository, add all the codes and commit changes.

```
git init
git add .
git commit -m “Initial commit”
```

Step8

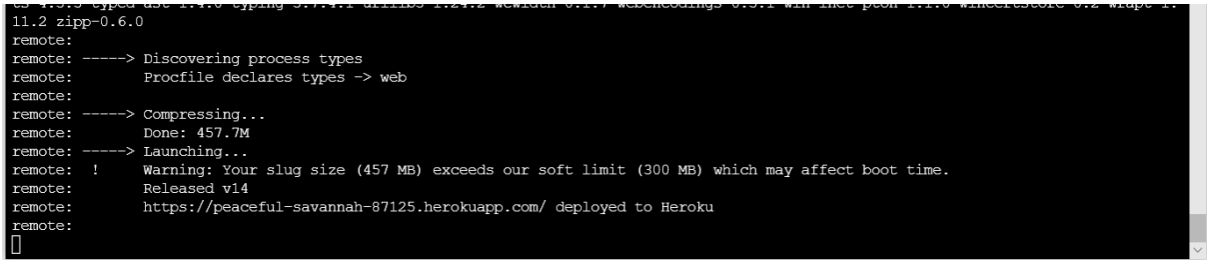
Using below command push the entire app on Heroku and open the url in the brower

```
git push heroku master
heroku open
```

Website opens and you can make neural machine translations.

LIMITATIONS

- We cannot put more than 500MB of data on a website.



- Sometimes needs to clear cache using below mentioned commands

```
heroku plugins:install heroku-repo
heroku repo:purge_cache -a appname
git commit --allow-empty -m "Purge cache"
git push heroku master
```

FUTURE SCOPE

- We can make add more models to a webpage to see random predictions of all the models

BUSINESS OBJECTIVE ACCOMPLISHED

Neural machine translation system can be used by anyone to translate Hindi to English, English to Hindi, French to English and English to French over the internet.

REFERENCES

- ***https://devcenter.heroku.com/articles/getting-started-with-python?singlepage=true***
- ***https://towardsdatascience.com/designing-a-machine-learning-model-and-deploying-it-using-flask-on-heroku-9558ce6bde7b***