Testing Model on PC and Deploying Using Flask:

1. Create virtual environment and activate it

It's better to create a virtual environment and install the required libraries in it so that the new libraries don't conflict with the existing ones. Create your project's folder, open cmd and type the following commands for environment creation and activation

```
C:\Users\User\Desktop\New folder (2)>python -m venv C:\Users\User\Desktop\New folder (2)

C:\Users\User\Desktop\New folder (2)>folder\Scripts\activate.bat
```

Figure 1: Environment creation and activation

2. Set python packages

Once the environment is activated, we have to install the required libraries as a list of python packages and their versions in a separate file called requirements.txt for example.

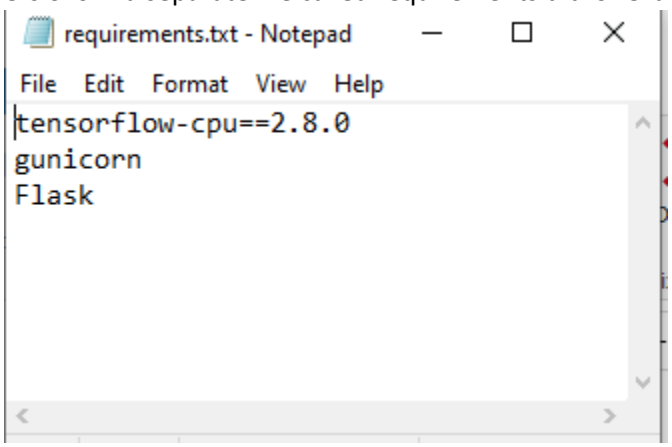

Figure 2: Prerequisite libraries list

Tensorflow library contains many other libraries like numpy, scipy etc.. so they will get installed along the way.

Then we can use the below command to install them.

```
pip install -r requirements.txt
```

3. Project Structure

Name	Date modified	Type	Size
__pycache__	4/30/2022 3:52 PM	File folder	
folder	4/29/2022 8:07 PM	File folder	
model	4/30/2022 5:28 PM	File folder	
templates	4/30/2022 4:46 PM	File folder	
app.py	5/1/2022 1:19 AM	PY File	2 KB
command.md	5/1/2022 5:53 PM	MD File	1 KB
en_word_index.txt	4/29/2022 2:41 PM	Text Document	3 KB
fr_word_index.txt	4/29/2022 2:43 PM	Text Document	6 KB
requirements.txt	5/3/2022 11:46 AM	Text Document	1 KB

- “folder” folder is my virtual environment
- “model” folder contains “En_Fr_Translation.h5” and “model.py” . h5 file is our model trained, and “model.py” is used to import model and execute the prediction.
- “templates” folder contains an “index.html” file used for flask.
- “app.py” file is the web application using flask
- “en_word_index.txt” and “fr_word_index.txt” are our English and French vocabularies used in the “model.py” script
- “Command.md” includes the commands needed

model.py 3

C: > Users > User > Desktop > New Folder (2) > model > model.py > main

```python
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
import re
import numpy as np
import pickle
import logging

class En_Fre_Translation:

    def __init__(self, model_path):
        logging.info("En_Fre_Translation class initialized")
        self.model = load_model(model_path)
        logging.info("Model is loaded!")

    def load_text(self, path):
        a_file = open(path, "rb")
        output = pickle.load(a_file)
        return output

    def translate(self,text,max_seq_length):
        text=re.sub(r'[^\w\s]', '', text)
        en_word_index=self.load_text('en_word_index.txt')
        fr_word_index=self.load_text('fr_word_index.txt')
        sentence = [en_word_index[word] for word in text.split()]
        sentence = pad_sequences([sentence], maxlen=max_seq_length, padding='post')
        result=self.model.predict(sentence[:1])[0]
        index_to_words = {id: word for word, id in fr_word_index.items()}
        index_to_words[0] = '<PAD>'
        result=' '.join([index_to_words[prediction] for prediction in np.argmax(result, 1)])
        result=result.split()
        output=[]
        for i in result:
            if i !='<PAD>':
                output.append(i)
        return (' '.join(output))

def main():
    model = En_Fre_Translation('.\model\En_Fr_Translation.h5')
    predicted_text = model.translate("she is driving the truck",21)
    logging.info( f"This is the translation: \n {predicted_text}")

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)
    main()
```

Figure 3: screenshot of model.py script used to load model and give a prediction for a specified example

To test our saved model, we created a model.py script that contains all needed statements to make a prediction for a specified example “she is driving the truck” (Figure 3).

Run the model.py using the below command (Figure 4)

```
C:\Users\User\Desktop\New Folder (2)>folder\Scripts\activate.bat

(folder) C:\Users\User\Desktop\New Folder (2)>python model\model.py
INFO:root:En_Fre_Translation class initialized
INFO:root:Model is loaded!
INFO:root:This is the translation:
 elle conduit le nouveau camion noir
```

Figure 4: screenshot of activating environment and running model.py

4. Creating Web application using flask

Here we create the routes for the endpoints of our application. We used html files to create an input form so the user can write English sentence and press the submit button to translate it to French using route ("\"). The sentence is translated using translate function that imports the model.py file to make the prediction.

```
app.py
C: > Users > User > Desktop > New Folder (2) > app.py > ...
from flask import Flask, request, render_template
import logging
import os

from model.model import En_Fre_Translation

# define model path
model_path ='./model/En_Fr_Translation.h5'
#model_path ='model\En_Fr_Translation.h5'

app = Flask(__name__)

# create instance
model = En_Fre_Translation(model_path)
logging.basicConfig(level=logging.INFO)

def translate(sentence):
    logging.info("translation request received!")
    prediction = model.translate(sentence,21)
    logging.info("translation from model",prediction)

    return prediction

@app.route("/")
def my_form():
    return render_template('index.html')

@app.route("/", methods=['GET','POST'])
def input_text():
    if request.method == 'POST':
        text=request.form['en_sent']
        logging.info("input sentence is =",  text)
        translated = translate(text)
        logging.info("Send translation request!")

    return translate(text)

def main():
    """Run the Flask app."""
    app.run(host="0.0.0.0", port=8000, debug=True)


if __name__ == "__main__":
    main()

```

Figure 5: "app.py" script to create flask application

The flask app runs on the host at port 8000. To test the app, go to cmd and write:

```
(folder) C:\Users\User\Desktop\New Folder (2)>python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
INFO:werkzeug: * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:8000
 * Running on http://10.0.0.13:8000 (Press CTRL+C to quit)
INFO:werkzeug: * Restarting with stat
```

Figure 6: Run application on local host

We used this url "http://10.0.0.13:8000" to run application on the browser. Once the page is opened, you can type your sentence, press submit button to view the translation.

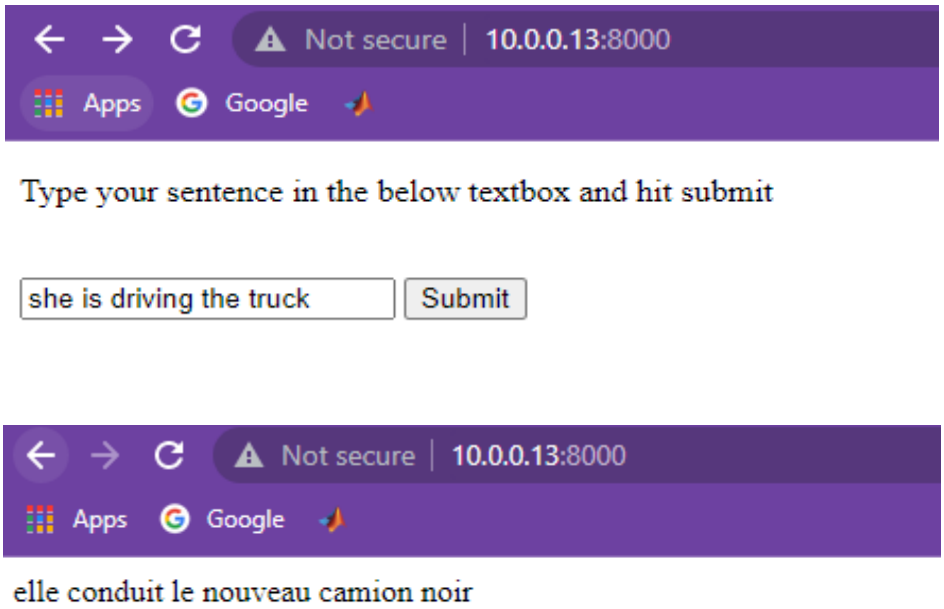

Figure 7: Sentence translation

Deploy Using Docker

Docker solves this "but it works on my machine" problem by also packaging the OS and the system libraries into a Docker container — a self-contained environment that works anywhere where Docker is installed.

1. Create Docker file

This image contains description of our service that includes all the settings and dependencies. Docker uses the image to create a container. To do it, we need a Dockerfile which is a file with instructions on how the image should be created.

Brief Explanation about the instructions:

Typically, the base image already contains the OS and the system libraries like Python itself, and we only need to install the dependencies of our project. In our case, we use python 3.9. Then, we create the working directory and name folder "En_Fr_Translation". Then, we are copying and moving the requirements.txt file and installing all the libraries it contains. Then, we copy and move the model folder. After that, we specify which port our application uses, In our case, it's 8000. Then from our new directory we are running the flask application (same app as above).

Dockerfile ×

C: > Users > User > Desktop > Project > Dockerfile

```
FROM  python:3.9

# Create the working directory
RUN set -ex && mkdir /En_Fr_Translation
WORKDIR /En_Fr_Translation

# Install Python dependencies
COPY requirements.txt ./requirements.txt
RUN pip install -r requirements.txt

# Copy the relevant directories
COPY model/ ./model
COPY . ./

# Run the web server
EXPOSE 8000
ENV PYTHONPATH /En_Fr_Translation
CMD python3 /En_Fr_Translation/app.py
```

Figure 8: Dockerfile

Note that the "Dockerfile" has no extension, and the requirements file contains just 2 libraries, see Figure 9.

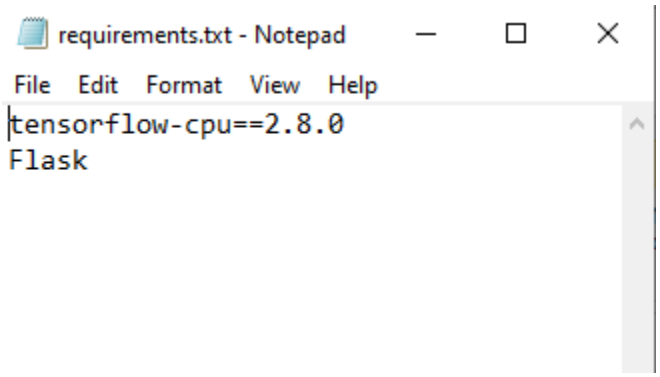

Figure 9: Prerequisite libraries list

2. Create Docker Image

Once dockerfile is created, now we can build the image and test it following the statements in the command.md file. Then to build the image we need to type the commands in the Figure 10, here our image name is translation.

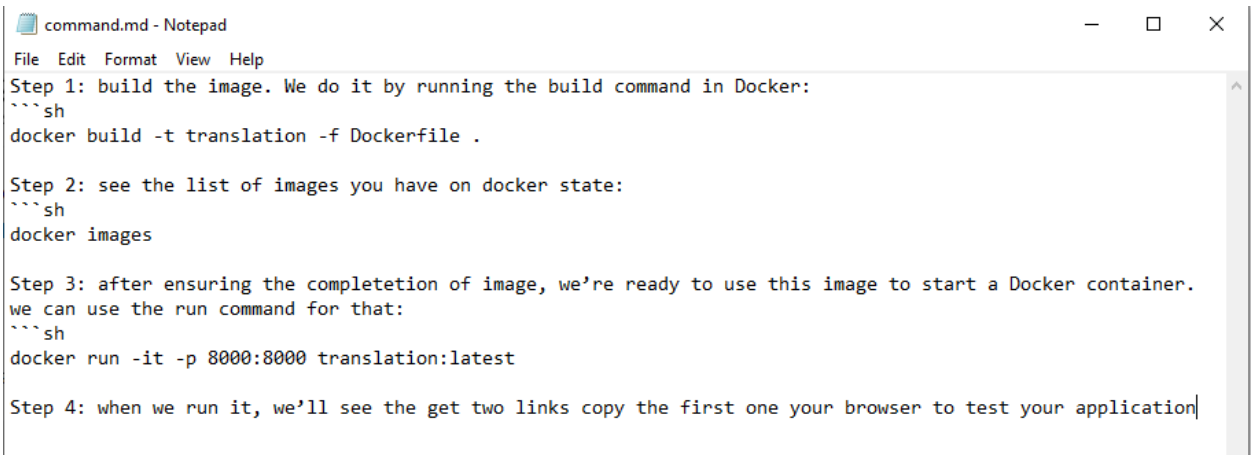

Figure 10:Building image using Docker

```
C:\Users\User\Desktop\Project>docker build -t translation -f Dockerfile .
[+] Building 3133.6s (13/13) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.1s
 => => transferring dockerfile: 448B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.1s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.9                                          21.9s
 => [auth] library/python:pull token for registry-1.docker.io                                           0.0s
 => [internal] load build context                                                                       4.7s
 => => transferring context: 15.13MB                                                                    3.8s
 => [1/7] FROM docker.io/library/python:3.9@sha256:a83c0aa6471527636d7331c30704d0f88e0ab3331bbc460d4ae2e53bbae64d  0.0s
 => CACHED [2/7] RUN set -ex && mkdir /En_Fr_Translation                                                0.0s
 => CACHED [3/7] WORKDIR /En_Fr_Translation                                                             0.0s
 => CACHED [4/7] COPY requirements.txt ./requirements.txt                                               0.0s
 => [5/7] RUN pip install -r requirements.txt                                                        3093.9s
 => [6/7] COPY model/ ./model                                                                           0.4s
 => [7/7] COPY . ./                                                                                     0.5s
 => exporting to image                                                                                 12.0s
 => => exporting layers                                                                                11.9s
 => => writing image sha256:0d6d7b356e5d8a57a8047050c89cc4bbb7538b33e084c0d24c540212ee1a283a             0.0s
 => => naming to docker.io/library/translation                                                          0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\User\Desktop\Project>docker images
REPOSITORY              TAG       IMAGE ID       CREATED         SIZE
translation             latest    0d6d7b356e5d   6 minutes ago   2.14GB
en_fr_trans             latest    8d951158316b   2 days ago      5.29GB
docker/getting-started  latest    cb90f98fd791   3 weeks ago     28.8MB
```

```
C:\Users\User\Desktop\Project>docker run -it -p 8000:8000 translation:latest
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
INFO:werkzeug: * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:8000
 * Running on http://172.17.0.2:8000 (Press CTRL+C to quit)
INFO:werkzeug: * Restarting with stat
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 103-720-635
INFO:werkzeug:172.17.0.1 - - [03/May/2022 11:26:10] "GET / HTTP/1.1" 200 -
INFO:werkzeug:172.17.0.1 - - [03/May/2022 11:26:11] "GET /favicon.ico HTTP/1.1" 404 -
INFO:werkzeug:172.17.0.1 - - [03/May/2022 11:27:33] "POST / HTTP/1.1" 200 -
```

Figure 11: Creating and Deploying model using Docker

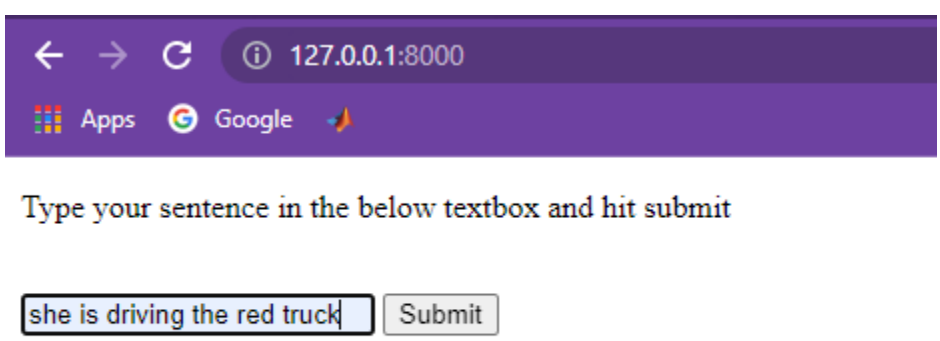

Figure 12: Testing Model using Docker