|  |
| --- |
| KF5012 |
| API DOCUMENTATION |
| Group: Only fools and horses |

|  |
| --- |
| Name: Michael Anderson  ID: W18032133 |

**API DOCUMENTATION**

API Design

The API was created using the flask framework, this framework was chosen because it is lightweight and easier to learn which allowed for faster development, the frameworks built in server and debug mode also helped during development.

The API takes a movie plot sent from a form in the web GIU and uses the trained model to predict the genre of the movie, this prediction is then sent as a response back to the GUI.

The API accepts a POST request using the flask route() decorator to bind a function to the URL. The predict\_genre() function is used to respond to these requests and uses the request.form() function from the flask package to receive the move plot data sent via a POST request from a form in the GUI.

The API imports modelLoader.py which contains a class, inputSan that has functions used to sanitise the received movie plot.

These methods first check if the received movie plot string is empty to prevent the model making a prediction on an empty string. If the movie plot is empty an error message is return to the client notifying them of the error and asking them to enter a valid movie plot, if the movie plot string is not empty then it is sanitised which involves removing contractions between words, removing stop words, remove any none alphabetic characters and making sure all words are in lower case.

After the movie plot is sanitised it is checked again to see if it is empty, which would happen if the movie plot entered only contained stop words or none alphabetic characters, if the movie plot is not empty it will be sent to the model to make a prediction.

This sanitation process is the same process used on the data the model was trained on, sanitising the received movie plot in the same way ensures the model is familiar with the format of the data and can accurately make a prediction on it. This process also ensures that the move plot will be reduced to the words most descriptive of its genre which will also increase the likelihood of the model making an accurate prediction.

The API holds an instance of the trained model that is loaded from file when the API is ran, this instance is held outside of the predict\_genre() function to improve performance by avoiding it being re-loaded every time the function is called.

After sanitisation, the model makes a prediction on the movie plot. The model will return a prediction of the movie’s genre and an accuracy percentage.

The jsonify() function from the flask package is used to create a json object containing the predicted genre and accuracy returned from the model’s prediction. This json object is then sent back to the GUI.

Model Preparation

For the trained model to be used in the API it had to be persisted to file, this was achieved using the joblib module. At first a pickle file was created of the model using the joblib.dump() method but the size of this file was 375mb, so it was necessary to also compress the model by setting the compress argument in the dump method equal to ‘gzip’ this reduced the size of the file to 5.73mb and created a .gz file.

This process was repeated for each version of the model and each of the compressed model files were stored in the ‘Models’ directory to allow the API to access and load them.

A helper class, called fileManager() was created within the modelLoader.py file, this class contains functions to help load and persist objects to file.

The final version of the model was changed, instead of persisting the model itself, a wrapper class was created by Chris that held the instance of the trained model, this class called Model provided functions that gave extra functionality such as making a prediction and calculating the accuracy as a percentage. When the API starts it creates an instance of this Model class and loads the persisted model.

Server Setup

The server currently runs locally using flasks built in server. This is appropriate for this assessment but not for a real distribution. In a real distribution the flask API could be wrapped in a Docker container. This would give the model fault tolerance, meaning if an error occurred with the model a new docker container could be created quickly and easily to take its place. The docker containers could then be deployed using Kubernetes which would give the model the ability the scalability needed to meet a high demand.

User Testing

To test the API several tests were carried. These aimed to test the different aspects of the API.

The API returns an error message if an empty movie plot is sent to it or if a movie plot consisting of stop words is sent to it as these will be removed during sanitation leading to an empty string. Both scenarios were tested. These tests aimed to test the error detection that is built into the API. The stop word test also tests the API’s ability to sanitise movie plots.

The API was also tested on making genuine predictions, so 2 movie plots were selected from IMDB and sent to the API. This tested the API’s ability to receive data, sanitise it and return a response.

The API was also tested using different sizes of movie plots to test its capacity. A large movie plot consisting 300 words was sent to the API as well as 1 word, none stop word movie plot.

Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Test Data | Expected Result | Actual Result | Notes |
| Empty movie plot | “ “ | “ERROR: Empty move plot entered Please enter a  movie plot for prediction” | “ERROR: Empty move plot entered Please enter a movie plot for prediction” | The API returned the correct error message meaning it can correctly handle empty strings |
| Stop word movie plot | “the this and is” | “ERROR: Invalid movie plot entered  Please enter a  movie plot for prediction” | “ERROR: Invalid movie plot entered  Please enter a movie plot for prediction” | The API returned the correct error message meaning it can correctly handle stop words |
| Movie Genre 1 | “The surviving members of the resistance face the First Order once again, and the legendary conflict between the Jedi and the Sith reaches its peak bringing the Skywalker saga to its end” | Acton, Adventure or Fantasy | Adventure, 26.395% | The model correctly predicted the movie meaning the API received the movie plot and correctly returned a response. |
| Movie Genre 2 | “When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, Batman must accept one of the greatest psychological and physical tests of his ability to fight injustice.” | Action or Drama | Action, 53.110% | The model correctly predicted the movie meaning the API correctly received the movie plot, sanitisation worked correctly the response was returned successfully |
| Long Movie Genre | The sentence: “shoot run home explosion scary dark sad love chase police” will be repeated 30 times to create a sentence with 300 words | Any genre | Romance, 23.437% | The model made a prediction for the large movie plot meaning that the API can correctly handle higher capacity movie plots |
| One-word movie genre | “shoot” | western | Western, 35.104% | The model correctly predicted the genre meaning the API works correctly with single word movie plots as long as they are not stop words. |

Testing Discussion

The API was able to correctly process empty strings and strings consisting of stop words, this means the API correctly detected the empty strings and correctly filtered out the stop words.

The API was able to correctly make predictions on the two ordinary movie plots showing that the model correctly receives, sanitises, and returns data.

The API made a prediction on the large movie plot, consisting of 300 words, this shows that the API can handle high capacity genres. This is fine for the scope of this assessment but in a real distribution this could pose a risk as vastly large movie plots could be sent to the API which could overwhelm the server. For a real distribution it is recommended that the API creates an upper limit on the size of a movie plot.