

Project Report: Question Detection Model

Overview

This report outlines the development and deployment of a machine learning model designed to identify whether sentences from video transcripts are questions. The project includes data preprocessing, model training, API development, and dockerization for deployment.

Data Preprocessing

Data preprocessing involved cleaning text data, encoding categorical labels, and preparing the data for model training.

Steps:

1. **Text Cleaning:** Lowercased all text, removed punctuation, and stripped whitespaces.
2. **Label Encoding:** Converted text labels into integers using `LabelEncoder`.

```
from sklearn.preprocessing import LabelEncoder

data.drop(columns=data.columns[0], inplace=True)
data['sentence'] = data['sentence'].str.lower().str.replace(r'[^w\s]', '', regex=True).str.strip()

label_encoder = LabelEncoder()
data['label'] = label_encoder.fit_transform(data['label'])
```

Model Training

We utilized sentence embeddings and trained multiple models, selecting Logistic Regression based on its performance.

Vectorization

- Sentence Embeddings: Transformed sentences using the SentenceTransformer model.

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('all-MiniLM-L6-v2')
sentence_embeddings = model.encode(data['sentence'].tolist())
```

Model Selection

Logistic Regression was chosen due to its high performance on our dataset, evaluated using the AUC metric.

```

from sklearn.linear_model import Logistic Regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score

```

```

X_train, X_test, y_train, y_test = train_test_split(sentence_embeddings, data['label'], test_size=0.2)
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train, y_train)

```

API Development

Developed a Flask API to serve predictions from the trained model. The API preprocesses input sentences, generates embeddings, and returns predictions.

```

from flask import Flask, request, jsonify
import joblib

app = Flask(__name__)
lr_model = joblib.load('logistic_regression_model.pkl')
sentence_model = SentenceTransformer('all-MiniLM-L6-v2')
label_encoder = joblib.load('label_encoder.pkl')

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    cleaned_sentence = data['sentence'].lower().replace(r'~\w\s', ' ').strip()
    sentence_embedding = sentence_model.encode([cleaned_sentence])
    prediction = lr_model.predict(sentence_embedding)
    decoded_prediction = label_encoder.inverse_transform([int(prediction[0])])[0]
    return jsonify(prediction=decoded_prediction)

if __name__ == '__main__':
    app.run(debug=True)

```

Dockerization

Containerized the Flask application using Docker to ensure consistent environments across different deployments.

Dockerfile

```

FROM python:3.8-slim
WORKDIR /app
COPY . /app
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 5000
CMD ["python", "app.py"]

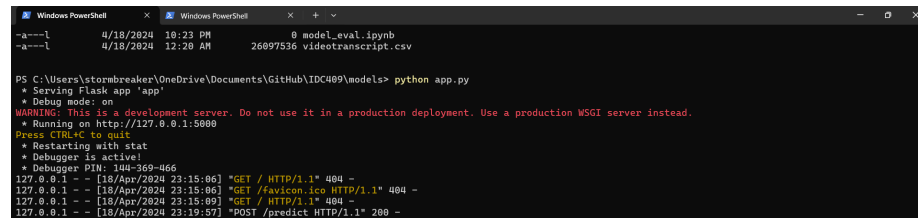
```

Build and Run

```
docker build -t my-flask-app .  
docker run -p 5000:5000 my-flask-app
```

Terminal Images

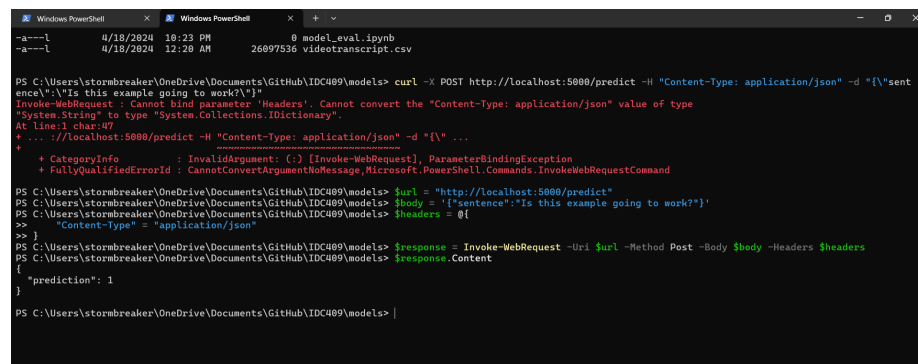
Flask running



```
PS C:\Users\stormbreaker\OneDrive\Documents\GitHub\IDC409\models> python app.py  
* Serving Flask app "app"  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 144-369-466  
127.0.0.1 - - [18/Apr/2024 23:15:06] "GET / HTTP/1.1" 404 -  
127.0.0.1 - - [18/Apr/2024 23:15:06] "GET /favicon.ico HTTP/1.1" 404 -  
127.0.0.1 - - [18/Apr/2024 23:15:09] "GET / HTTP/1.1" 404 -  
127.0.0.1 - - [18/Apr/2024 23:15:57] "POST /predict HTTP/1.1" 200 -
```

Figure 1: Running Flask

Model doing prediction



```
PS C:\Users\stormbreaker\OneDrive\Documents\GitHub\IDC409\models> curl -X POST http://localhost:5000/predict -H "Content-Type: application/json" -d '{"sentence": "Is this example going to work?"}'  
Invoke-WebRequest : Cannot bind parameter 'Headers'. Cannot convert the "Content-Type: application/json" value of type  
"System.String" to type "System.Collections.IDictionary".  
At line:1 char:47  
+ ... ://localhost:5000/predict -H "Content-Type: application/json" -d '{" ...  
+ ~~~~~  
+ CategoryInfo          : InvalidArgument: (:) [Invoke-WebRequest], ParameterBindingException  
+ FullyQualifiedErrorId : CannotConvertArgumentNoMessage,Microsoft.PowerShell.Commands.InvokeWebRequestCommand  
  
PS C:\Users\stormbreaker\OneDrive\Documents\GitHub\IDC409\models> $url = "http://localhost:5000/predict"  
PS C:\Users\stormbreaker\OneDrive\Documents\GitHub\IDC409\models> $body = '{"sentence": "Is this example going to work?"}'  
PS C:\Users\stormbreaker\OneDrive\Documents\GitHub\IDC409\models> $headers = @{}  
>> "Content-Type" = "application/json"  
>>  
PS C:\Users\stormbreaker\OneDrive\Documents\GitHub\IDC409\models> $response = Invoke-WebRequest -Uri $url -Method Post -Body $body -Headers $headers  
PS C:\Users\stormbreaker\OneDrive\Documents\GitHub\IDC409\models> $response.Content  
{  
  "prediction": 1  
}  
  
PS C:\Users\stormbreaker\OneDrive\Documents\GitHub\IDC409\models>
```

Figure 2: Prediction