Data Preprocessing
First, let's import the necessary libraries and load the dataset.

python
Edit
Full Screen
Copy code
```
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

nltk.download('stopwords')
nltk.download('wordnet')

data =
pd.read_csv('https://gist.githubusercontent.com/farhaan-settyl/ecf9c1e7ab7374f18e4400b7a3d2a161/raw/8e7a8a3a0a6a8a3a0a6a8a3a0a6a8a3a0a6a8a3a/sample_data.csv')
```
Next, we'll preprocess the external status descriptions by removing special characters, converting to lowercase, removing stopwords, and lemmatizing the words.

python
Edit
Full Screen
Copy code
```
def preprocess_text(text):
    text = re.sub(r'\W+', ' ', text)
    text = text.lower()
    words = text.split()
    stop_words = set(stopwords.words('english'))
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
    return ' '.join(words)

data['external_status_description'] = data['external_status_description'].apply(preprocess_text)
```
Model Development
We'll use a simple LSTM model for this task.

python
Edit
Full Screen
Copy code
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_features = 10000
max_len = 200

tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(data['external_status_description'])
X = tokenizer.texts_to_sequences(data['external_status_description'])
X = pad_sequences(X, maxlen=max_len)
Y = pd.get_dummies(data['internal_status']).values

model = Sequential()
model.add(Embedding(max_features, 128, input_length=max_len))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(Y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Model Training and Evaluation

python

Edit

Full Screen

Copy code

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
model.fit(X_train, Y_train, batch_size=32, epochs=10, validation_data=(X_test, Y_test))
```

API Development

We'll use FastAPI to create a simple API.

python

Edit

Full Screen

Copy code

```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class StatusDescription(BaseModel):
    description: str

@app.post("/predict/")
async def predict(status: StatusDescription):
    description = status.description
    description = preprocess_text(description)
    description = tokenizer.texts_to_sequences([description])
```

```
    description = pad_sequences(description, maxlen=max_len)
    prediction = model.predict(description)
    predicted_status = np.argmax(prediction)
    return {"internal_status": data['internal_status'].iloc[predicted_status]}
```

Testing and Validation

We can test the A