Trellis Data Science Case Study

Document Classification Algorithm

Objective:

Develop a document classification algorithm for a dataset of text files (.txt) labeled under 11 categories: Technology, Sport, Space, Politics, Medical, Historical, Graphics, Food, Entertainment, Business, and Other. The goal is to classify these documents accurately and expose the functionality through an API.

Dataset: https://www.dropbox.com/scl/fo/bsx6t0y86eicr15xm2haa/AJvvER3VtuXJ090Bcvnh1ml?rlkey=mf7s184ymqlw7pdz64n1eymc0&st=z99aunov&dl=0

Requirements:

1. **Model Development:**
   - Choose any appropriate algorithm for document classification.
   - Train the model on the provided dataset and validate its accuracy using appropriate metrics.
2. **API Development:**
   - Develop a RESTful API to expose the document classification model.
   - The API should allow users to submit a text document and receive the predicted category as a response.
   - Ensure the API is built with appropriate error handling and validation mechanisms in place.
   - Python is a must. Choose the framework that you are most comfortable (e.g. Django, Flask, FastAPI).
   - The predictive model should be expose as:

   Endpoint

   `POST /classify_document`

   Request Body

   ```
   {
   "document_text": "string, required"
   }
   ```

   Response

   Detail the possible response statuses and data. Include success and error messages, and describe the response body for each scenario.

   Response Example (You don't need to follow it):

   - **Success Response**
     - **Code:** 200 OK
     - **Content:**

   ```
   {
     "message": "Classification successfully",
     "label": "sport"
   }
   ```

Additional Information

- We understand there are multiple solutions to the problem. The focus is to understand the thought process, don't worry too much on the model performance for example, this isn't being directly evaluated. The process is what matters.
- Make an effort to develop the API in a manner similar to a production environment, considering relevant factors and best practices.
- Consider the computational costs and inference time when scaling up to classify millions of documents.
- The 'other' folder should not be utilized for training the model; it is intended to serve as a test to determine if the model can accurately classify unrecognized documents as 'other'.
- The deliverable will be a **Git repository** containing both the training scripts/notebooks and the API code.