REAL-TIME TAG RECOMMENDATION SYSTEM FOR STACK OVERFLOW

About:

Our project focuses on crafting a tag recommendation system for Stack Overflow questions, leveraging the Stack Overflow Kaggle dataset. Using Spark and the Spark-NLP library, we've developed a robust machine learning model. This model predicts tags for questions, aiding users in discovering relevant content.

Dataset:

In order to train our model we will be using the Stack overflow kaggle dataset to evaluate the implementation. For the final presentation, we will be using real-time test data from Stack overflow API.
Specifications of training dataset: Count: 30M+ records Size: 3.5GB

Prerequisites:

Installations:

This is an example of how to list things you need to use the software and how to install them in mac.

Install Homebrew (if not installed):

Open Terminal and run the following command

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Install Python:

Once Homebrew is installed, use the following command in Terminal to install Python

```
brew install python
```

Install Java:

Use the following command in Terminal to install AdoptOpenJDK

```
brew install --cask adoptopenjdk
```

Install Scala:

Use the following command in Terminal to install Scala

```
brew install scala
```

Apache Spark:
Follow the below documentation to install apache spark
https://sparkbyexamples.com/spark/install-apache-spark-on-mac/

PySpark:
Use the following command in Terminal to install Pyspark
`pip3 install pyspark`

Spark NLP:
Use the following command in Terminal to install spark-nlp
`pip install spark-nlp`

NLTK:
Use the following command in Terminal to install NLTK
`pip install nltk`

Data Preprocessing:

The data preprocessing phase begins by consolidating questions and their corresponding tags into a single DataFrame, filtering out questions with scores below 5 for relevance. All tags are converted to lowercase, and the most common 50 tags are identified. Filtering tags based on this commonality, the dataset is refined, retaining only the relevant tags. Additionally, text data undergoes extensive cleaning steps, including the removal of HTML, stopwords, special characters, and stemming. The 'Title' and 'Body' columns are individually preprocessed to optimize text data for further analysis. Finally, the processed data is exported to a CSV file for subsequent analysis and model development.

```
Python
python data_preprocessing_step_1.py
```

Note: Make sure that the following files are present in the current directory - "*questions.csv*", "*tags.csv*"

Running the code with Python:

The provided Python script orchestrates tag prediction for Stack Overflow questions using a Random Forest classifier. It begins by prepping the data, encoding tags, and transforming text into TF-IDF vectors. The script then trains the model, evaluates its accuracy on a test set, and records the overall runtime. Executing this script manages

the entire workflow, from data preparation to model assessment, facilitating efficient tag prediction for Stack Overflow queries.

```
Python
python model_training_python_step_2.py
```

Note: Make sure that the following file which is generated from the data preprocessing step to be present in the current directory - "*data.csv*"

Running the code with Pyspark:

The included PySpark script demonstrates tag prediction for Stack Overflow questions. It initiates a Spark session, preprocesses the data by tokenizing and converting text to TF-IDF vectors, then builds and evaluates a Random Forest classifier. Executing this script via PySpark manages the end-to-end process of data preparation, model creation, and evaluation for accurate tag prediction in Stack Overflow queries.

```
Python
python model_training_pyspark_step_3.py
```

Note: Make sure that the following file which is generated from the data preprocessing step to be present in the current directory - "*data.csv*"

The above two methods have been implemented for comparing the runtime of the model training using python and pyspark.

Model Testing:

The repository contains a Flask API that harmonizes PySpark and Spark NLP for data preprocessing and classification tasks. Utilizing a RandomForestClassifier, it processes CSV data, performs text preprocessing, computes TF-IDF, and offers endpoints (/api/<type>) to access accuracy metrics or filtered data based on specified types ('train' or 'test'). Ensure Python, Apache Spark, and Spark NLP are installed, configure settings accordingly, and run the Flask app to leverage this API for flexible data analysis and classification needs.

```Python
python model_testing_pyspark_step_4.py
```

Note: Make sure that the following file which is generated from the data preprocessing step to be present in the current directory - “*data.csv*”

Upon running the provided command, a local server link will be generated. Accessing this link along with the endpoint path "/api/test" will yield the desired result.

API: http://127.0.0.1:5000/api/test
Http method: GET