**PR 1: Purchases.txt Dataset**

**Write a Pig Latin scripts for finding TF-IDF value for book dataset (A corpus of eBooks available at: Project Gutenberg)**

To calculate TF-IDF values for a corpus of eBooks using Python, you can use libraries such as nltk, pandas, and scikit-learn. Here's a step-by-step guide to achieve this:

### Step 1: Prepare the Environment

Ensure you have the required libraries installed. You can install them using pip:

pip install nltk pandas scikit-learn.

### Step 2: Load and Preprocess the Data

Assume your eBooks are stored in a directory where each file represents a separate document.

**import os**

**import pandas as pd**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**# Path to the directory containing eBooks**

**books\_dir = '/path/to/gutenberg\_books'**

**# Load the dataset**

**def load\_books(directory):**

**documents = []**

**filenames = []**

**for filename in os.listdir(directory):**

**with open(os.path.join(directory, filename), 'r', encoding='utf-8') as file:**

**documents.append(file.read())**

**filenames.append(filename)**

**return documents, filenames**

**documents, filenames = load\_books(books\_dir)**

### Step 3: Tokenization and TF-IDF Calculation

Use TfidfVectorizer from scikit-learn to compute the TF-IDF values.

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**# Initialize the TF-IDF Vectorizer**

**vectorizer = TfidfVectorizer(stop\_words='english', max\_df=0.95, min\_df=2)**

**# Fit and transform the documents**

**tfidf\_matrix = vectorizer.fit\_transform(documents)**

**# Get feature names (terms)**

**terms = vectorizer.get\_feature\_names\_out()**

**# Convert the TF-IDF matrix to a pandas DataFrame**

**tfidf\_df = pd.DataFrame(tfidf\_matrix.toarray(), index=filenames, columns=terms)**

### Step 4: Output the TF-IDF Values

Now, you can output the TF-IDF values to a CSV file or perform further analysis.

**# Save the TF-IDF values to a CSV file**

**tfidf\_df.to\_csv('tfidf\_values.csv')**

**# Display the TF-IDF DataFrame**

**print(tfidf\_df)**

### Complete Script

Here is the complete script:

**import os**

**import pandas as pd**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**# Load the dataset**

**def load\_books(directory):**

**documents = []**

**filenames = []**

**for filename in os.listdir(directory):**

**with open(os.path.join(directory, filename), 'r', encoding='utf-8') as file:**

**documents.append(file.read())**

**filenames.append(filename)**

**return documents, filenames**

**# Path to the directory containing eBooks**

**books\_dir = '/path/to/gutenberg\_books'**

**documents, filenames = load\_books(books\_dir)**

**# Initialize the TF-IDF Vectorizer**

**vectorizer = TfidfVectorizer(stop\_words='english', max\_df=0.95, min\_df=2)**

**# Fit and transform the documents**

**tfidf\_matrix = vectorizer.fit\_transform(documents)**

**# Get feature names (terms)**

**terms = vectorizer.get\_feature\_names\_out()**

**# Convert the TF-IDF matrix to a pandas DataFrame**

**tfidf\_df = pd.DataFrame(tfidf\_matrix.toarray(), index=filenames, columns=terms)**

**# Save the TF-IDF values to a CSV file**

**tfidf\_df.to\_csv('tfidf\_values.csv')**

**# Display the TF-IDF DataFrame**

**print(tfidf\_df)**

**Explanation**

1. **Load the Data**: The load\_books function reads each document from the specified directory.
2. **TF-IDF Vectorization**: TfidfVectorizer from scikit-learn computes the TF-IDF values for the terms in the documents.
3. **Output**: The TF-IDF values are stored in a pandas DataFrame and saved to a CSV file.

This script calculates the TF-IDF values for a corpus of eBooks and outputs them to a CSV file for further analysis. Adjust the books\_dir variable to point to your dataset directory.