CCSS Text Similarity Recommendation System
- Andria Grace

Introduction
The objective of this project is to develop a recommendation system for Common Core State Standards (CCSS) using text similarity techniques. The system utilizes Natural Language Processing (NLP) methods to preprocess textual descriptions of CCSS standards, applies the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization, and builds a Nearest Neighbors model to find and recommend the most relevant standards based on user input.

Dataset
The dataset contains textual descriptions of various CCSS standards. Each record includes:

- ID: A unique identifier for the CCSS standard.
- Description: A textual description of the CCSS standard.

Code Implementation

1. Loading the Dataset

```
import pandas as pd

def load_dataset(file_path):
    df = pd.read_csv(file_path)
    print("Step 1: Load Dataset")
    print(df.head())  # Display the first few rows of the dataframe
    return df[['id', 'description']]

file_path = 'C:/Users/acer/Downloads/ccss.csv'
ccss_df = load_dataset(file_path)
```

Description:
The dataset is loaded from a CSV file and read into a pandas DataFrame. We extract only the columns id and description for further processing.

```
Step 1: Load Dataset
```

	id	content_type	category_id	category_name	grade_id	grade_name	item	description
0	CCSS.ELA-LITERACY.L.K.1	ELA-LITERACY	L	Language	K	Kindergarten	1	Demonstrate command of the conventions of stan...
1	CCSS.ELA-LITERACY.L.K.1.a	ELA-LITERACY	L	Language	K	Kindergarten	1a	Print many upper- and lowercase letters.
2	CCSS.ELA-LITERACY.L.K.1.b	ELA-LITERACY	L	Language	K	Kindergarten	1b	Use frequently occurring nouns and verbs.
3	CCSS.ELA-LITERACY.L.K.1.c	ELA-LITERACY	L	Language	K	Kindergarten	1c	Form regular plural nouns orally by adding /s/...
4	CCSS.ELA-LITERACY.L.K.1.d	ELA-LITERACY	L	Language	K	Kindergarten	1d	Understand and use question words (interrogati...

```
Step 1: Load Dataset
                         id  content_type category_id category_name grade_id  \
0    CCSS.ELA-LITERACY.L.K.1  ELA-LITERACY           L      Language        K
1  CCSS.ELA-LITERACY.L.K.1.a  ELA-LITERACY           L      Language        K
2  CCSS.ELA-LITERACY.L.K.1.b  ELA-LITERACY           L      Language        K
3  CCSS.ELA-LITERACY.L.K.1.c  ELA-LITERACY           L      Language        K
4  CCSS.ELA-LITERACY.L.K.1.d  ELA-LITERACY           L      Language        K

     grade_name item                                   description
0  Kindergarten    1  Demonstrate command of the conventions of stan...
1  Kindergarten   1a            Print many upper- and lowercase letters.
2  Kindergarten   1b          Use frequently occurring nouns and verbs.
3  Kindergarten   1c  Form regular plural nouns orally by adding /s/...
4  Kindergarten   1d  Understand and use question words (interrogati...
```

2. Data Preprocessing

```
from sklearn.feature_extraction.text import TfidfVectorizer

def preprocess_data(df):
    vectorizer = TfidfVectorizer(stop_words='english')
    tfidf_matrix = vectorizer.fit_transform(df['description'])
    print("Step 2: Data Preprocessing")
    print("TF-IDF Matrix Shape:", tfidf_matrix.shape)
    print("Feature Names:", vectorizer.get_feature_names_out()[:10])  # Display the first 10
feature names
    return vectorizer, tfidf_matrix

vectorizer, tfidf_matrix = preprocess_data(ccss_df)
```

Description:
The TfidfVectorizer is used to convert the textual descriptions into TF-IDF vectors, which capture the importance of each word in the context of the entire dataset. This step transforms the text data into a numerical format suitable for model building.

```
Step 2: Data Preprocessing
TF-IDF Matrix Shape: (1554, 2886)
Feature Names: ['01' '01212t' '02' '05' '05a' '0s' '10' '100' '1000' '100s']
```

3. Model Building

```
from sklearn.neighbors import NearestNeighbors

def build_model(tfidf_matrix):
    model = NearestNeighbors(n_neighbors=5, algorithm='auto')
    model.fit(tfidf_matrix)
    print("Step 3: Model Building")
    print("Model built with n_neighbors=5")
    return model

model = build_model(tfidf_matrix)
```

Description:
A Nearest Neighbors model is trained on the TF-IDF matrix. This model is used to find the nearest (most similar) CCSS standards based on their textual descriptions.

```
Step 3: Model Building
Model built with n_neighbors=5
```

4. Making Predictions

```
def find_closest_ccss(input_text, vectorizer, model, df, n=5):
    input_tfidf = vectorizer.transform([input_text])
    distances, indices = model.kneighbors(input_tfidf, n_neighbors=n)
    closest_ids = df.iloc[indices[0]]['id'].values
    print("Step 4: Making Predictions")
    print("Input Text:", input_text)
    print("Closest CCSS ids:", closest_ids)
    return closest_ids

input_text = "Understand and use question words"
closest_ccss_ids = find_closest_ccss(input_text, vectorizer, model, ccss_df, n=5)
```

Description:
Given an input text, the model transforms the text into a TF-IDF vector and finds the nearest CCSS standards. The function returns the IDs of the closest CCSS standards based on the similarity of descriptions.

```
Step 4: Making Predictions
Input Text: Understand and use question words
Closest CCSS ids: ['CCSS.ELA-LITERACY.L.K.1.d' 'CCSS.ELA-LITERACY.L.8.5.b'
 'CCSS.ELA-LITERACY.L.5.5.c' 'CCSS.ELA-LITERACY.L.6.5.b'
 'CCSS.ELA-LITERACY.L.7.5.b']
```

Results

The system successfully demonstrates the ability to process textual descriptions, build a similarity model, and find the most relevant CCSS standards based on input text. The closest CCSS standards are identified and displayed, providing a practical recommendation based on the similarity of descriptions.

Conclusion

The CCSS Text Similarity Recommendation System effectively leverages NLP and machine learning techniques to recommend relevant educational standards. By using TF-IDF vectorization and the Nearest Neighbors algorithm, the system can match user-provided text with the closest CCSS standards. Future work could involve expanding the dataset, refining the model, or incorporating additional features to enhance recommendation accuracy and functionality.

Future Work

Enhance Dataset: Include more comprehensive descriptions and additional features.
Improve Model: Experiment with more advanced similarity measures or deep learning techniques.
User Interface: Develop a user-friendly application for interacting with the recommendation system.