Django Application for Recipe Analysis

Overview

This Django application performs clustering and similarity analysis on a collection of recipes. It uses various techniques to process the data and generate visual insights. The main functionalities include:

- Reading recipe data from a CSV file and saving it to a database.
- Clustering recipes based on their ingredients.
- Calculating centroids for clusters.
- Finding recipes similar and dissimilar to a given recipe or centroid.
- Generating visualizations for data analysis.

Documentation and Justifications

Data Ingestion and Cleaning

- **Function:** `read_csv_and_save_to_db(file_path)`
    - **Purpose**: Reads recipe data from a CSV file and stores it in the database.
    - **Justification**: This function ensures that the application can load initial data from a CSV file, which is a common format for data storage and exchange.
- **Function:** `clean_all_recipes()`
    - **Purpose**: Deletes all records from the `Recipe` model.
    - **Justification**: Provides a way to reset the database, which is useful for testing or restarting the application with fresh data.

Data Processing

- **Function:** `tokenize(recipes)`

    - **Purpose**: Converts a list of recipes into a binary matrix indicating the presence or absence of ingredients.
    - **Justification**: This step transforms textual data into a numerical format that can be used by machine learning algorithms.
- **Function:** `get_PCA_data(recipes)`
    - **Purpose**: Applies Principal Component Analysis (PCA) to reduce the dimensionality of the data to two dimensions.
    - **Justification**: PCA is used to simplify the data and make it easier to visualize while retaining as much information as possible.
- **Function:** `get_point_data(recipes)`
    - **Purpose**: Generates the x and y coordinates for plotting the recipes in a 2D space after PCA transformation.
    - **Justification**: This function aids in visualizing the recipes in a scatter plot, making it easier to analyze the clustering of recipes.

Clustering and Centroid Calculation

- **Function:** `get_centroid(recipes)`
    - **Purpose**: Calculates the centroids of the clusters formed by the recipes.
    - **Justification**: Centroids represent the center of clusters, which helps in identifying the average characteristics of recipes within the same dish category.
- **Function:** `get_centroid_plot(centroids, recipes, closer_recipes, far_recipes)`
    - **Purpose**: Prepares data for plotting the centroids and recipes on a scatter plot.
    - **Justification**: This function visualizes the relationship between recipes and their cluster centroids, highlighting similar and dissimilar recipes.

Similarity and Dissimilarity Analysis

- **Function:** `get_similarity_recipes(recipes)`
    - **Purpose**: Computes the cosine similarity matrix for the recipes.
    - **Justification**: Cosine similarity measures the similarity between recipes based on their ingredients, which is crucial for finding similar

recipes.

- **Function:** `find_similar_recipes(recipes, similarity_matrix)`
    - **Purpose**: Identifies recipes that are most similar to each other.
    - **Justification**: Finding similar recipes helps users discover recipes that share common ingredients and possibly similar flavors.
- **Function:** `find_dissimilar_recipes(recipes, similarity_matrix)`
    - **Purpose**: Identifies recipes that are most dissimilar to each other.
    - **Justification**: Identifying dissimilar recipes can help users explore diverse recipes with different ingredients.

Visualizations

- **Function:** `get_bar_data(recipes)`
    - **Purpose**: Prepares data for a bar chart showing the frequency of ingredients.
    - **Justification**: This visualization provides insights into the most commonly used ingredients across recipes, helping to understand ingredient popularity.

View Logic

- **View:** `index(request)`
    - **Purpose**: The main view that handles user interactions, processes data, and renders the template.
    - **Justification**: This view integrates all functionalities, allowing users to reset the database, filter recipes by dish or recipe name, and visualize data through charts.

Justifications for Algorithm Choices

- **PCA (Principal Component Analysis)**
    - **Chosen for**: Dimensionality reduction.
    - **Justification**: PCA reduces the complexity of the data by projecting it into a lower-dimensional space, which is easier to visualize and analyze while retaining significant variance.
- **Cosine Similarity**

    - **Chosen for**: Calculating recipe similarity.
    - **Justification**: Cosine similarity is effective in measuring the similarity between high-dimensional vectors (binary vectors of ingredients) and is less affected by the number of ingredients than Euclidean distance.
- **Euclidean Distance**
    - **Chosen for**: Finding recipes closer and farther from the centroid.
    - **Justification**: Euclidean distance is straightforward and interpretable for calculating the distance between points in the reduced PCA space.

Template Logic

- **Template: `index.html`**
    - **Purpose**: Provides the front-end structure for displaying charts and handling user inputs.
    - **Justification**: The template integrates CanvasJS for rendering charts, enhancing the user interface with interactive and exportable visualizations.

Visualization Tools

- **CanvasJS**
    - **Justification**: CanvasJS is used for rendering interactive charts. It supports features like zooming, animation, and exporting, making the visualizations more engaging and user-friendly.

Summary

The application leverages various algorithm techniques and visual tools to provide insights into a collection of recipes. Each step of the process, from data ingestion to visualization, is designed to ensure accuracy, efficiency, and usability. The choices of algorithms and libraries are justified based on their effectiveness and suitability for the tasks at hand.