iNeuron.ai

Low-Level Document

Big Data Internship Report

Social media community using optimized clustering algorithm

Submitted by

Arbash Hussain

Project: Social Media Community Using Optimized Clustering Algorithm

Overview

This document provides a detailed low-level design for the Social Network User Clustering project. It includes detailed descriptions of modules, classes, functions, and interactions within the system.

1. Data Ingestion

Purpose: Generate synthetic user and post data for clustering analysis.

Class: `Data_Ingestion`

Methods:

- `__init__()`:
  - Initializes the `Faker` library and Spark session.
  - **Attributes:**
    - `self.faker`: An instance of the `Faker` class for generating random data.
    - `self.spark`: A Spark session for data processing.

- `initiate_data_ingestion(num_users, num_posts)`:
  - Parameters:
    - `num_users` (int): Number of synthetic users to generate.
    - `num_posts` (int): Number of synthetic posts to generate.
  - Process:
    - Defines a Spark schema for the data.
    - Generates random user IDs and post IDs.
    - Creates synthetic data including timestamps, content, and view/comment counts.
    - Returns a Spark DataFrame containing the generated data.

- `save_to_csv(df, file_path)`:
  - Parameters:
    - `df` (DataFrame): The Spark DataFrame to be saved.
    - `file_path` (str): The path where the CSV file will be saved.
  - Process:
    - Saves the DataFrame to a CSV file in the specified path.

2. Data Transformation

Purpose: Aggregate and transform data to prepare it for clustering analysis.

Class: `Data_Transformation`

Methods:

- `__init__()`:
  - Initializes the Spark session for data transformation.
  - Attributes:
    - `self.spark`: A Spark session for data processing.

- `initiate_data_transformation(spark_df)`:
  - Parameters:
    - `spark_df` (DataFrame): Input Spark DataFrame containing raw user data.
  - Process:
    - Aggregates data by `user_id` to calculate `unique_posts`, `total_views`, and `total_comments`.
    - Computes `comment_to_view_ratio`.
    - Uses `VectorAssembler` to combine features into a single vector.
    - Applies `StandardScaler` to standardize features.
    - Utilizes `PCA` to reduce dimensionality to two principal components.
    - Returns a Pandas DataFrame of PCA features and the user activity DataFrame.

3. Model Training

Purpose: Cluster users based on their activity using the K-Means algorithm.

Class: `Model_Trainer`

Methods:

- `__init__()`:
  - Initializes the model trainer.

- `initiate_model_trainer(pca_features, user_activity)`:
  - Parameters:

    - `pca_features` (DataFrame): Pandas DataFrame of PCA features.
    - `user_activity` (DataFrame): Pandas DataFrame of aggregated user activity data.
  - Process:
    - Extracts PCA features as a NumPy array.
    - Calculates WCSS (Within-Cluster Sum of Squares) for K-Means with clusters ranging from 1 to 10.
    - Uses the elbow method to determine the optimal number of clusters.
    - Trains a K-Means model with the optimal number of clusters.
    - Assigns cluster labels to `user_activity` and returns the updated DataFrame.

4. Visualization

Purpose: Visualize user clusters using an interactive plot.

Class: `Visualizer`

Methods:

- `__init__()`:
  - Initializes the visualizer.

- `visualize(pca_features, user_activity)`:
  - Parameters:
    - `pca_features` (DataFrame): Pandas DataFrame of PCA features.
    - `user_activity` (DataFrame): Pandas DataFrame of user activity with cluster assignments.
  - Process:
    - Converts PCA features to a DataFrame with `pca_x` and `pca_y` columns.
    - Merges cluster information from `user_activity`.
    - Creates an interactive scatter plot using Plotly to display clusters.
    - Exports the plot as an HTML file.

5. Web Application

Purpose: Provide a user interface for executing the clustering pipeline and displaying results.

Framework: Flask, Flask-SocketIO

Components:

- Flask App:
  - Endpoints:
    - `/run_pipeline`: Starts the clustering pipeline.
    - `/view_results`: Displays the clustering results.

Interaction Flow

1. Data Generation: Synthetic data is generated and stored as a DataFrame.
2. Data Aggregation: User activity is aggregated and transformed.
3. Clustering: PCA is applied and K-Means clustering is performed.
4. Visualization: Clusters are visualized and saved as an HTML plot.
5. Web Interface: Users initiate the pipeline and view results via a Flask web application.

Conclusion

The low-level design document provides a comprehensive breakdown of each component within the user clustering project, highlighting the purpose, methods, and interactions of various modules. This ensures clarity in understanding the system and aids in maintaining and extending the project.