Report For DataNeuron Data Science Internship

Part A:

In Part A, the core approach involves utilizing the Sentence Transformers library to calculate the semantic textual similarity between pairs of texts. First, the necessary libraries are imported, including Pandas for data manipulation and NumPy for numerical operations. The Sentence Transformers library is imported to leverage pre-trained models for encoding text into fixed-dimensional embeddings that capture semantic information.

The dataset containing pairs of texts is loaded using Pandas. Basic data exploration is performed to check for missing values and understand the structure of the dataset. The Sentence Transformers model is then initialized, specifically using the "all-MiniLM-L6-v2" model, which is a pre-trained model known for its efficiency and effectiveness.

Text embeddings are computed for each text in the dataset using the Sentence Transformers model. Cosine similarity scores are then calculated between the embeddings of corresponding pairs of texts. Additionally, a mapping function is defined to scale the cosine similarity scores between 0 and 1, as the original scores range from -1 to 1.

The calculated similarity scores are assigned to a new column in the dataset, and the dataset is exported to a CSV file named 'Solution_Task1.csv'.

Part B:

In Part B, a Flask web application is created to expose an API endpoint for calculating the semantic similarity score between two input texts. The application utilizes the Sentence Transformers library to compute text embeddings and cosine similarity scores, similar to Part A.

The Flask application defines a single endpoint ("/calculate-similarity") that accepts HTTP GET requests. The endpoint expects two query parameters: 'text1' and 'text2', representing the two texts for which the similarity score needs to be calculated.

A Swagger documentation is integrated into the Flask application to provide documentation for the API endpoint. Within the endpoint implementation, the input texts are retrieved from the request, and the Similarity Score function is called to compute the semantic similarity score. The calculated score is then returned as a JSON response.

The Flask application is configured to run on port 8000 in debug mode. When executed, the application listens for incoming requests and responds with the computed similarity score for the provided texts.

Report By.
Shreyas Patil

Student At MITWPU, Pune

patilshreyas808@gmail.com

+91 9167939171