Report: Semantic Textual Similarity Task

Part A: Model Building - Semantic Similarity

Objective:
Build an algorithm that quantifies the semantic similarity between two paragraphs and predicts a score between 0 (completely dissimilar) and 1 (completely similar).

Approach:

- **Preprocessing:**
  The provided dataset contained pairs of paragraphs (`text1` and `text2`). We loaded the dataset using `pandas` and validated its structure.

- **Embedding Generation:**
  To convert text into meaningful numerical form, we used a **pre-trained Transformer model** from HuggingFace (`bert-base-uncased`).
  Each paragraph was tokenized and passed through the model to obtain embeddings. We averaged the token embeddings to represent the paragraph as a fixed-size vector.

- **Similarity Calculation:**
  We computed the **cosine similarity** between the two embedding vectors to determine how similar the two paragraphs are semantically.
  Cosine similarity score was scaled between 0 and 1 (rounded to 4 decimal places) to match the required output format.

- **Output:**
  For each text pair, the final output included `text1`, `text2`, and their corresponding `similarity_score`.

Part B: Deployment on Cloud

Objective:
Deploy the model as a live API service where users can send paragraph pairs via an HTTP request and receive a similarity score.

Approach:

- **API Development:**
  We used **FastAPI**, a lightweight and high-performance web framework for building APIs in Python.

- **API Endpoint:**

    - **Route:** `/predict`

Request Body:

json
CopyEdit

```
{
  "text1": "paragraph 1 text",
  "text2": "paragraph 2 text"
}
```

    -

Response Body:

json
CopyEdit

```
{
  "similarity score": 0.85
}
```

    -
    - The API accepts two text inputs, computes the embeddings, calculates cosine similarity, and returns the result.

- **Deployment:**
  The API was containerized and deployed on **Render** (a cloud service provider).
  A `requirements.txt` file listed all the dependencies, and a `start.sh` script was used to launch the application with Uvicorn server on port 10000.

- **Live Endpoint:**
  The API is publicly accessible and ready to handle requests.

Summary:

This solution efficiently quantifies semantic similarity using a pre-trained BERT model and exposes the logic through a simple, scalable cloud API, fulfilling all the given task requirements.