Short Approach Report: Semantic Textual Similarity Model and API Deployment

Part A: Building the Semantic Similarity Model

In response to the provided problem statement, I successfully developed a model to quantify the degree of semantic similarity between pairs of text paragraphs. The goal was to predict a value between 0 and 1, where 1 indicates high similarity, and 0 signifies high dissimilarity.

Steps:

1. Importing the relevant libraries
2. Loading the dataset
3. Text Cleaning and Preprocessing
4. Jaccard Similarity Technique
5. Text Vectorization - Count Vectorizer - TF(Term Frequency)
6. TF-IDF Transformer(Term Frequency-Inverse Document Frequency)
7. Cosine Similarity
8. Leveraging Google News Word 2 Vector library
9. Saving and loading the cosine similarity model for deployment

Part B: Deployment as a Server API Endpoint

I successfully deployed the trained model as a server API endpoint on the cloud using Amazon Web Services (AWS).

Deployment Strategy:

1. **API Development**: I created a RESTful API using the Flask web framework, which provided a user-friendly interface for making similarity score predictions. The API exposed a single endpoint that accepted POST requests with the text pairs to be compared.
2. **Model Integration**: Within the API, I loaded the Jaccard similarity and pre-processing logic for text pairs. When a request was received, the API processed the input text pairs, passed them through the model, and returned the predicted similarity score.
3. **Cloud Hosting**: I chose AWS as our cloud service provider due to its scalability and ease of use. The API was hosted on an AWS EC2 instance. I configured security groups and network settings to ensure the API was secure and accessible.