Semantic Textual Similarity (STS) Assignment – Short Report

Candidate: Roshni Seth
Date: 22-Oct-2025

Part A – Model Approach

Objective

The goal is to quantify semantic similarity between two text paragraphs. The model predicts a **score between 0 and 1**:

- `1` → Highly similar
- `0` → Highly dissimilar

Model Selection

- **Model Used:** `sentence-transformers/all-MiniLM-L6-v2`
- **Reason:** Lightweight, memory-efficient, fast inference, suitable for cloud deployment.
- **Framework:** Python, `sentence-transformers` library

Methodology

1. Encode each text paragraph into embeddings using the pre-trained model.
2. Compute **cosine similarity** between the embeddings.
3. Normalize similarity to 0–1 scale:

```
normalized_score = (cosine_similarity + 1) / 2
```

4. Return the score as the similarity measure.

Part B – Deployment

Platform

- **Hugging Face Spaces** (Free tier, 1GB RAM)
- **Reason:** Publicly accessible, easy GitHub integration, supports Gradio for API deployment.

API Implementation

- **Framework:** Gradio (`gr.Interface`)
- **Endpoint:** `/run/predict`
- **Request Format:**

```
{
  "data": ["text1 paragraph", "text2 paragraph"]
}
```

- **Response Format:**

```
{
  "similarity score": 0.8456
}
```

Testing

1. Optional test_api.py script can be used locally to test the model using Python requests library.

2. Since the HF Space URL is publicly accessible, instructors can test the model **directly via browser** without needing the test script.

Files Submitted

1. **api_app.py** – Model + API code
2. **requirements.txt** – Dependencies
3. **README.md** – HF Space configuration
4. **Updated Resume** – Contact information included

   Note: test_api.py is optional. Instructors can use the Live HF Space URL for testing.

Live API Endpoint

https://huggingface.co/spaces/Roshni231123/Semantic_Textual_Similarity

- Publicly accessible for testing and evaluation.
- Correctly returns similarity score for any pair of text paragraphs.

Notes

- Model is optimized for **semantic similarity** detection.
- Lightweight model ensures deployment in **limited memory environments**.
- Assignment evaluation based on **model correctness**, **API deployment**, and **response format adherence**.