Natural Language Processing with Transformers - Assignment Report

Student Name: Daniel Muthama
Program: DATA and AI 2025
Date: 22/7/2025

1. Introduction

This assignment explores the application of BERT (Bidirectional Encoder Representations from Transformers) for semantic similarity analysis between sentence pairs. The primary objectives were:

1. Implement BERT for sentence encoding using Hugging Face Transformers.
2. Compute cosine similarity between sentence embeddings.
3. Predict similarity based on a threshold (0.7) and evaluate accuracy against manual labels.

The project highlights the advantages of contextual embeddings over traditional NLP methods like TF-IDF or Bag-of-Words.

2. Task Completion

Step 1: Environment Setup

- Installed required libraries:

```
!pip install --upgrade transformers tf-keras
```

- Resolved compatibility issues between Keras 3 and Transformers by using `tf-keras`.

Step 2: Data Preparation

- Sentence Pairs: Extended the provided 5 pairs to 10 (e.g., "How do I learn Python?" vs. "What is the best way to study Python?").
- Labels: Manually assigned binary labels (1=similar, 0=not similar).

Step 3: BERT Embeddings

- Loaded `bert-base-uncased` tokenizer and model:

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
bert_model = TFBertModel.from_pretrained('bert-base-uncased')
```

- Extracted [CLS] token embeddings for sentence-level representations:

```
def get_sentence_embedding(sentence):
    inputs = tokenizer(sentence, return_tensors='tf', padding=True, truncation=True)
    outputs = bert_model(inputs)
    return outputs.last_hidden_state[:, 0, :].numpy()
```

Step 4: Cosine Similarity & Predictions

- Computed cosine similarity between embeddings for each pair.
- Predicted similarity if `score > 0.7` (threshold).
- Example Output:

```
Sentence 1: How do I learn Python?
Sentence 2: What is the best way to study Python?
Cosine Similarity: 0.9743 → Predicted Similar: 1
```

Step 5: Evaluation

- Accuracy: 80% (8/10 correct predictions).
- Mismatch: Incorrectly predicted "What is AI?" vs. "How to cook pasta?" as similar (score: 0.9033).

3. Key Concepts Explained

1. BERT vs. Traditional NLP

- BoW/TF-IDF: Static word representations, ignoring context.
- BERT: Dynamic contextual embeddings (e.g., "bank" differs in "river bank" vs. "bank account").

2. Encoder Role in BERT

- BERT uses Transformer encoders with self-attention to capture word relationships.
- Encoder outputs contextual embeddings used for similarity tasks.

3. Contextual Embeddings

- Generated by processing sentences through BERT’s layers.
- Usage: Extracted `[CLS]` token embeddings represent aggregated sentence meaning.

4. Cosine Similarity

- Measures angle between vectors:

$$\text{cosine similarity} = \frac{A \cdot B}{\|A\| \|B\|}$$

- Why Useful: Quantifies semantic closeness (range: -1 to 1).

4. Conclusion

- Achievements: Successfully applied BERT for semantic similarity with 80% accuracy.
- Challenges: Threshold tuning (e.g., 0.7 may not generalize for all datasets).
- **Learnings:**
  - Contextual embeddings outperform static methods.
  - BERT’s computational cost vs. traditional approaches.

Notebook Link: [Google Colab]

Appendix: Code**[GITHUB]**

1. Installation: Libraries setup and compatibility fixes.
2. Embedding Extraction: Code execution for `[CLS]` token.
3. Results: Similarity scores and accuracy calculation.