Stance-Aware Transformers for Opinion Mining

1 Introduction

Opinion mining in NLP is a crucial task that involves analyzing and understanding different viewpoints on a topic. Sentence transformers, based on self-attention encoder architectures, perform well in grouping semantically similar texts. However, they struggle to differentiate between opposing perspectives within the same topic. This study explores this limitation and proposes fine-tuning strategies using contrastive learning techniques to enhance the model's ability to distinguish between contrasting stances.

To address this challenge, we implemented and compared two contrastive learning-based architectures: **Siamese Network** and **Triplet Network**. Both models were fine-tuned using the training dataset created from given text files, which consists of structured debates containing Pro and Con arguments. The models were evaluated by analyzing cosine similarity distributions before and after fine-tuning.

2 Model Architecture

2.1 Siamese Network

A Siamese Network consists of two identical subnetworks that process two input sentences simultaneously and compute their similarity using a distance metric. We trained the Siamese model using **cosine similarity loss**, which learns to bring semantically similar sentences closer and push dissimilar ones apart.

Implementation:

- **Input:** Sentence pairs (X1, X2) labeled as either agreeing (1) or disagreeing (0).
- **Base Model:** sentence-transformers/all-mpnet-base-v2
- **Loss Function:** Cosine Similarity Loss
- **Training Objective:** Reduce distance for similar pairs and increase it for dissimilar pairs.

2.2 Triplet Network

The Triplet Network extends the Siamese architecture by incorporating an additional negative example. It consists of three inputs:

- **Anchor:** A claim or discussion title.
- **Positive Sample:** A Pro argument supporting the anchor.
- **Negative Sample:** A Con argument opposing the anchor.

The model is trained using **triplet loss**, which ensures that the anchor is closer to the positive sample than the negative sample in the embedding space.

Implementation:

- **Input:** Triplets (Anchor, Pro, Con)
- **Base Model:** sentence-transformers/all-mpnet-base-v2
- **Loss Function:** Triplet Loss
- **Training Objective:** Maximize the distance between the anchor and the negative sample while minimizing the distance between the anchor and the positive sample.

3 Training Procedure

3.1 Dataset Preparation

The training and evaluation datasets were created from the **Kialo** debate corpus:

- **SiameseTrainingDataCreation.py**: Processed raw text files to extract Anchor-Pro/Anchor-Con arguments.
- **TripletTrainingDataCreation.py**: Processed raw text files to extract Anchor-Pro-Con arguments.
- **SiameseTestDataCreation.py**: Generated test and validation datasets for the Siamese Network.
- **TripletTestDataCreation.py**: Generated test and validation datasets for the Triplet Network.

3.2 Siamese Network Training

- **Training Data:** SiameseTraining.csv
- **Batch Size:** 8
- **Epochs:** 3

- **Loss Function:** Cosine Similarity Loss
- **Optimizer:** Adam with a learning rate of $2e-5$
- **Warmup Steps:** 50
- **Hardware:** GPU-enabled (CUDA)

3.3 Triplet Network Training

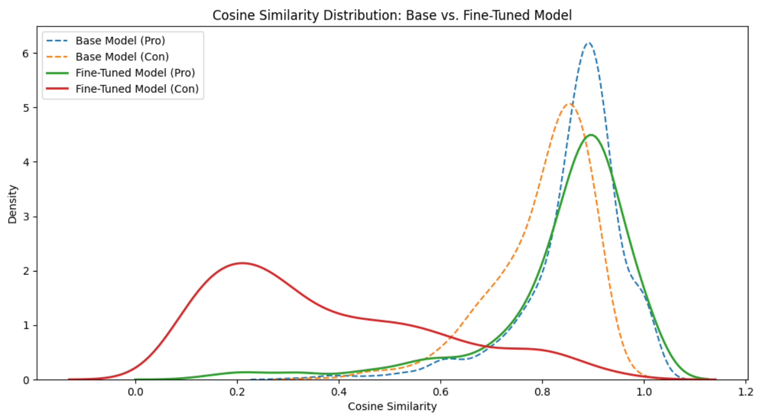
- **Training Data:** TripletTraining.csv
- **Batch Size:** 8
- **Epochs:** 3
- **Loss Function:** Triplet Loss
- **Optimizer:** Adam with a learning rate of $2e-5$
- **Warmup Steps:** 50
- **Hardware:** GPU-enabled (CUDA)

4 Results

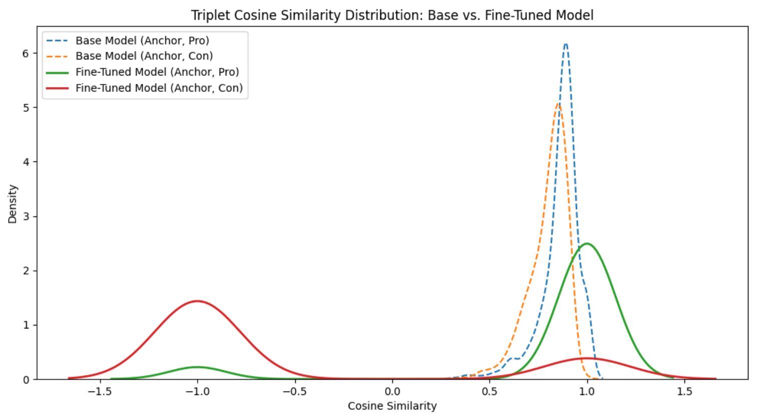
4.1 Graphical Comparison

The following plots visualize the cosine similarity distribution for both networks. The graphical analysis suggests that both fine-tuned models learned to separate opposing stances more effectively than the base model, with the Triplet network showing a more distinct separation.

- **Siamese Network:**
  - The KDE plot illustrates that before fine-tuning, Pro and Con arguments had overlapping cosine similarity distributions with the claim.
  - After fine-tuning, the similarity between Pro-Claim pairs increased, while Con-Claim pairs had a noticeable decrease in similarity.



- **Triplet Network:**
  - The KDE plots demonstrate a sharper distinction after fine-tuning, with Pro-Claim pairs clustering closer to 1 and Con-Claim pairs shifting toward lower similarity scores.
  - The fine-tuned model achieved a better separation between Pro and Con arguments compared to the Siamese model.



4.2 Quantitative Metrics

To provide a more rigorous evaluation, we calculated the accuracy, precision, recall, and F1-score for both models on a test set. The results in Table 1 and Table 2 confirm the effectiveness of fine-tuning.

Table 1: Siamese Model Performance Comparison (Threshold = 0.5)

Metric	**Base Model**	**Fine-Tuned Siamese Model**
Accuracy	0.5021	0.8331
Precision	0.5011	0.7699
Recall	0.9886	0.9503
F1-Score	0.6651	0.8506

Table 2: Triplet Model Performance Comparison (Threshold = 0.0)

Metric	**Base Model**	**Fine-Tuned Triplet Model**
Accuracy	0.5000	0.8544
Precision	0.5000	0.8138
Recall	1.0000	0.9190
F1-Score	0.6667	0.8632

5 Conclusion

This study demonstrated the limitations of pre-trained sentence transformers in distinguishing opposing viewpoints and how fine-tuning with contrastive learning improves stance awareness.

Key findings:

- Both the Siamese and Triplet fine-tuning methods substantially improved performance over the base model, as confirmed by quantitative metrics.
- The Triplet model performed better than the Siamese model, achieving a higher Accuracy (0.8544 vs. 0.8331) and F1-Score (0.8632 vs. 0.8506). This supports the conclusion from the graphical analysis that explicitly forcing separation between similar and dissimilar arguments is a more effective strategy.
- Cosine similarity distributions and quantitative metrics showed clear improvements after fine-tuning.

Future Improvements:

- Increasing dataset diversity: Training on more diverse opinionated texts.
- Hyperparameter tuning: Experimenting with different learning rates and optimizers.