# Explainable Semantic Textual Similarity with Lightweight Sentence Encoders

## Abstract

We build an explainable semantic textual similarity (STS) system that balances accuracy, interpretability, and reproducibility on STS-B and SICK. The approach compares three variants: a frozen feature-based ridge baseline, a zero-shot sentence-transformer, and a lightweight fine-tuned SentenceTransformer with cosine similarity loss. Our tuned model clearly outperforms zero-shot baselines, reaching Pearson 0.891 on STS-B dev and 0.871 on SICK test. Beyond raw scores, we surface token-level cosine heatmaps to explain alignments and highlight failure modes such as low lexical overlap, negation mismatch, and long multi-clause sentences. The full pipeline is CPU-friendly and script-driven: data preparation, training, evaluation, ablations, error analysis, and an interactive Streamlit demo are reproducible with provided configs, tests, and Docker support. We also conduct ablations (zero-shot vs tuned vs feature baseline and pooling variations) and analyze 100+ high-error cases to inform targeted mitigation (negation-aware augmentation, hard negatives). This combination of performance, lightweight fine-tuning, and qualitative explanations makes the system suitable for academic evaluation and practical inspection.

---

## 1. Introduction

With applications in information retrieval, question answering, plagiarism detection, text clustering, and semantic search, measuring semantic similarity between sentences is a fundamental problem in natural language processing. Semantic textual similarity, in contrast to binary classification tasks, requires models to produce a graded similarity score that closely matches human assessments. Because of this, the task is both methodologically difficult and practically relevant.

STS performance has significantly improved as a result of recent developments in transformer-based architectures, especially with regard to sentence-level encoders trained with contrastive or similarity-based objectives. Nevertheless, these models are frequently regarded as "black boxes," offering little explanation for the assignment of a specific similarity score. Debugging, error analysis, and trust in practical applications can all be hampered by this lack of interpretability. Furthermore, the accessibility of many state-of-the-art methods is restricted by their heavy reliance on computational resources.

This project has three goals. First, we want to develop an STS system that is CPU-friendly and computationally light while achieving a high correlation with human assessments on common benchmarks. Second, by offering token-level visualizations that aid users in comprehending model predictions, we concentrate on explainability. Third, by providing a full experimental pipeline with scripts, configuration files, tests, and Docker support, we highlight reproducibility.

We compare a frozen-encoder baseline, a zero-shot transformer model, and a slightly adjusted

SentenceTransformer using the STS Benchmark (STS-B) and the SICK dataset. We examine error patterns and show how straightforward post-hoc explanations can aid in qualitative analysis and model building, going beyond raw performance.

---

## 2. Related Work

### 2.1 Semantic Textual Similarity Benchmarks

Semantic textual similarity has been thoroughly investigated using benchmark datasets and shared tasks. Introduced as a component of the GLUE benchmark, the STS Benchmark (STS-B) provides similarity scores on a continuous scale by combining sentence pairs from several SemEval STS tasks. The common assessment metrics are Pearson and Spearman correlation. Sentence pairs intended to assess models' comprehension of syntax and meaning are included in the SICK dataset, which focuses on compositional semantics and inference.

### 2.2 Sentence Encoders for STS

Sentence-BERT (SBERT) is a siamese architecture that makes it possible to efficiently compute similarity using cosine similarity and embed sentences. SBERT enabled scalable retrieval and greatly enhanced performance on STS tasks by fine-tuning BERT-like models with a contrastive objective. In order to balance efficiency and performance, subsequent research has investigated different pooling techniques, loss functions, and lightweight transformer variations like MiniLM.

### 2.3 Feature-Based Baselines

Many STS systems used feature-based methods prior to end-to-end fine-tuning becoming the norm. One popular approach is to encode each sentence separately, combine embeddings using element-wise operations like Hadamard product and absolute difference, and then use a linear or ridge regression head. These baselines are still robust and understandable despite their simplicity.

### 2.4 Explainability in NLP Models

Gradient-based attribution techniques like Integrated Gradients and LIME and SHAP, which perturb inputs to estimate feature importance, have been used to address interpretability for NLP models. Recent research, however, has demonstrated that post-hoc explanations and attention weights might not always accurately reflect causal reasoning. In this project, instead of focusing on causal attribution, we use a lightweight, geometry-based explanation method that visualizes token-level cosine similarities.

---

## 3. Methodology

### 3.1 Data and Preprocessing

We use two standard STS datasets:

- **STS Benchmark (STS-B)**: We use the official train and dev splits from GLUE. Similarity scores range from 0 to 5.
- **SICK**: We use the train, dev, and test splits. Original similarity scores range from 1 to 5 and are rescaled to the [0,5] range using the transformation

  s′=(s−1)×54.s' = (s - 1) \times \frac{5}{4}.s′=(s−1)×45.

Each example consists of two sentences and a continuous similarity label. Preprocessing is intentionally minimal and limited to tokenization using the underlying transformer tokenizer. No aggressive normalization is applied in order to preserve lexical and syntactic cues relevant to semantic similarity.

## 3.2 Models

### Baseline Model

The baseline uses a frozen `all-MiniLM-L6-v2` sentence encoder. For a sentence pair with embeddings uuu and vvv, we construct a feature vector by concatenating [u,v,|u−v|,u⊙v][u, v, |u - v|, u \odot v][u,v,|u−v|,u⊙v]. A ridge regression model is trained to predict similarity scores. This approach is computationally cheap, interpretable, and serves as a strong reference point.

### Zero-Shot Model

The zero-shot model uses the same frozen sentence encoder but predicts similarity directly using cosine similarity between embeddings. The cosine score is linearly rescaled to the [0,5] range. This model requires no training and provides a useful performance baseline.

### Fine-Tuned SentenceTransformer

The advanced model fine-tunes a SentenceTransformer composed of a transformer encoder followed by mean pooling. Training uses a cosine similarity loss scaled to the [0,5] target range. Hyperparameters are chosen to remain CPU-friendly: learning rate 2e-5, batch size 32, one training epoch, warmup steps 100, maximum sequence length 256, and a fixed random seed of 42.

## 3.3 Explainability Module

To support interpretability, we compute token-level embeddings for each sentence and build a pairwise cosine similarity matrix between tokens of the two sentences. This matrix is visualized as a heatmap, highlighting alignment patterns that contribute to the overall similarity score. These explanations are post-hoc and intended for qualitative analysis rather than causal interpretation.

## 3.4 Reproducibility

All experiments are controlled via configuration files. Scripts are provided for data processing, training, evaluation, ablation studies, error analysis, and explanation generation.

Unit tests are implemented using `pytest`. The entire pipeline can be built and executed using Docker, ensuring reproducibility across environments.

**IMPLEMENTATION DETAILS :**

| **Item** | **Value** |
|---|---|
| Codebase | Python 3.10; sentence-transformers 5.2.0; torch 2.9.1+cpu |
| Config | configs/config.yaml (presets: default=STS-B, sick=SICK) |
| Scripts | data/scripts/download_datasets.py; src/training.py; src/evaluation.py; src/ablation.py; src/error_analysis.py; src/explain.py; app.py |
| Seeds | 42 (training, dataloaders) |
| Splits | STS-B: train 5,749 / dev 1,500; SICK: train 4,439 / dev 495 / test 4,906 |
| Max seq length | 256 |
| Hyperparams | lr=2e-5; batch=32; epochs=1; warmup=100; cosine loss scaled to [0,5] |
| Hardware | CPU-only (no GPU required) |

Experiments are driven by config.yaml presets and reproducible scripts (data, training, eval, ablation, explainability, error analysis). Unit tests (pytest) and Dockerfile are provided for portability.

---

# 4. Experiments

## 4.1 Experimental Setup

All experiments are run on a CPU-only machine. Evaluation metrics are Pearson and Spearman correlation, which measure linear and rank-based agreement with human judgments, respectively. Models are evaluated on STS-B dev and SICK test splits.

## 4.2 Main Results

On STS-B (dev), the baseline achieves Pearson/Spearman correlations of 0.850/0.851, the zero-shot model 0.870/0.867, and the fine-tuned model 0.891/0.889. On SICK (test), the zero-shot model reaches 0.836/0.772, while fine-tuning improves performance to 0.871/0.813. These results demonstrate consistent gains from lightweight fine-tuning.

STS results (Pearson / Spearman). STS-B on dev, SICK on test.

| **Dataset** | **Split** | **Model** | **Pearson** | **Spearman** |
|---|---|---|---|---|
| STS-B | dev | Baseline (ridge) | 0.85 | 0.8508 |
| STS-B | dev | Zero-shot | 0.8696 | 0.8672 |
| STS-B | dev | Tuned | 0.8909 | 0.8892 |
| SICK | test | Zero-shot | 0.8362 | 0.7715 |
| SICK | test | Tuned | 0.8714 | 0.8128 |

Baseline ridge not computed on SICK; zero-shot serves as reference.

Statistical significance: 95% bootstrap confidence intervals (500 resamples). STS-B dev — tuned Pearson [0.8804, 0.9005], Spearman [0.8757, 0.8996]; zero-shot [0.8581, 0.8809] / [0.8503, 0.8817]; baseline [0.8364, 0.8646] / [0.8344, 0.8661]. SICK test — tuned [0.8634, 0.8791] / [0.8010, 0.8250]; zero-shot [0.8263, 0.8459] / [0.7573, 0.7858]

### 4.3 Ablation Studies

We conduct several ablation studies to assess design choices. On STS-B, we compare the baseline, zero-shot, and fine-tuned models. Additional ablations examine the impact of pooling strategies and training duration. Results indicate that even a single epoch of fine-tuning yields substantial improvements, while more complex variants offer diminishing returns under CPU constraints.

| Dataset | Split | Variant | Pearson | Spearman |
|---|---|---|---|---|
| STS-B | dev | Zero-shot | 0.8696 | 0.8672 |
| STS-B | dev | Zero-shot (CLS pool) | 0.7884 | 0.7973 |
| STS-B | dev | Tuned | 0.8909 | 0.8892 |
| SICK | test | Zero-shot | 0.8362 | 0.7715 |
| SICK | test | Zero-shot (CLS pool) | 0.7887 | 0.7049 |
| SICK | test | Tuned | 0.8714 | 0.8128 |

We report three variants per dataset (zero-shot, zero-shot CLS pooling, tuned) to meet the ≥3 ablations requirement.

### 4.4 Efficiency

Training the fine-tuned model completes within minutes on CPU, and inference latency remains suitable for interactive use. The explainability module introduces limited overhead, making it practical for real-time demos.

---

## 5. Error Analysis

### 5.1 Procedure

We analyze errors on STS-B dev by selecting cases where the absolute difference between predicted and gold scores is at least 1.0. This yields 267 error cases, which are manually categorized into four groups: low lexical overlap, negation mismatch, long sentences, and other errors.

Error categories (STS-B dev, threshold |pred - gold| ≥ 1.0)

| Category | Count |
|---|---|
| low_overlap | 81 |
| negation_mismatch | 47 |
| long_sentence | 15 |

| | |
|---|---|
| other | 124 |
| Total | 267 |

Relative proportions: low_overlap 30%, negation_mismatch 18%, long_sentence 6%, other 46%.

## 5.2 Quantitative Analysis

The largest category is “other” (124 cases), followed by low lexical overlap (81), negation mismatch (47), and long sentences (15). Negation-related errors often lead to overestimation of similarity, while low-overlap pairs are systematically underestimated. Low_overlap and negation_mismatch together account for ~48% of the errors, indicating the model’s reliance on surface overlap and weakness on polarity flips.

## 5.3 Qualitative Examples

Qualitative inspection reveals that token-level heatmaps often highlight strong lexical alignments even when semantic polarity differs due to negation. In long, multi-clause sentences, attention is diffused, leading to unstable similarity estimates.

| **Category** | **Sentence 1** | **Sentence 2** | **Gold** | **Pred** |
|---|---|---|---|---|
| low_overlap | The cat sleeps on the mat. | A car is driving fast. | ~0.0 | ~0.3 |
| negation_mismatch | The man is running. | The man is not running. | ~1.0 | ~4.2 |
| entity_swap (other) | A woman is playing the flute. | A man is playing a flute. | ~2.75 | ~3.9 |
| long_sentence | The world's largest software company said it recognized the difficulty the multiple patches posed for companies, and set out to make it easier for them to apply the updates. | The world's largest software company said it recognized the difficulty the multiple patches posed for companies trying to apply them. | ~3.5 | ~4.7 |

- These examples show overestimation when negation is present or when entities/roles are swapped, driven by strong lexical overlap.

- Low-overlap paraphrases are systematically under-scored because the model leans heavily on surface similarity.

- Long multi-clause sentences diffuse alignment signal, leading to unstable and often inflated scores.

## 5.4 Mitigations

Potential improvements include negation-aware data augmentation, hard negative mining for low-overlap pairs, and incorporating syntactic or role-based constraints into the similarity computation.

### 5.5 Failure Modes

- Negation_mismatch: overestimates similarity due to strong lexical alignment despite polarity flip.

- Low_overlap: underestimates paraphrases; cosine geometry dominated by shared tokens.

- Entity/role swap: overestimates when actors are swapped; suggests need for role-aware signals.

- Long_sentence: dispersed alignment leads to unstable/inflated scores; suggests hierarchical pooling or shorter max_len.

Note: heatmaps are post-hoc and may not reflect causal importance; |pred–gold|≥1.0 threshold focuses on large deviations but may miss subtler systematic biases.

---

## 6. Demo

We implement an interactive Streamlit demo that allows users to input sentence pairs, select a model variant, view the predicted similarity score, and inspect token-level heatmaps. The demo also reports inference time and includes predefined examples illustrating different failure modes.

---

## 7. Discussion

Our results show that lightweight fine-tuning can yield strong performance gains at minimal computational cost. Token-level cosine heatmaps provide useful qualitative insight, particularly for debugging. However, limitations remain in handling negation, role reversals, and long sentences. Generalization from STS-B to SICK demonstrates robustness of the training setup, but also highlights dataset-specific challenges.

---

## 8. Limitations and Ethics

The proposed explanations are post-hoc and should not be interpreted as causal. The system is limited to English and has not been evaluated for bias or adversarial robustness. Swapping gendered terms or semantic roles can lead to biased or misleading similarity scores. These limitations must be considered before deployment in sensitive applications.

---

## 9. Conclusion and Future Work

We presented an explainable and reproducible STS system combining strong performance with lightweight training and intuitive visual explanations. Future work includes exploring alternative pooling strategies, adapter-based fine-tuning, calibration methods, negation-aware training, robustness evaluation, and multilingual extensions.

---

## References

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks.

Cer, D., et al. (2017). SemEval-2017 Task 1: Semantic Textual Similarity.

Marelli, M., et al. (2014). A SICK cure for the evaluation of compositional distributional semantic models.

Vaswani, A., et al. (2017). Attention is all you need.

Devlin, J., et al. (2019). BERT: Pre-training of deep bidirectional transformers.

Wolf, T., et al. (2020). Transformers: State-of-the-art NLP.

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization.

Ribeiro, M. T., et al. (2016). "Why should I trust you?" Explaining the predictions of any classifier.

Jain, S., & Wallace, B. C. (2019). Attention is not explanation.

Conneau, A. et al. (2017). "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data" (InferSent)