Computation of Sentence Similarity Score through Hybrid Deep Learning with a Special Focus on Negation Sentence.

Rohit Mahesh

December 2025

1 Introduction

The global education landscape is undergoing rapid expansion, with student enrollments rising faster than the capacity of institutions to support them. At the same time, a persistent shortage of qualified teachers continues to widen the gap between instructional demand and available human resources.UNESCO reports that the world will require an additional 44 million teachers by 2030 to meet universal education goals[1]. Even among Organisation for Economic Co-operation and Development (OECD) countries, the average student–teacher ratio remains around 14:1 in primary and 13:1 in lower secondary education, reflecting the challenge of providing individualized attention at scale [2]. This growing demand intensifies the workload on teachers, who already spend a significant portion of their time on non-teaching tasks such as grading and administrative work. Studies show that teachers devote nearly half of their working hours to such responsibilities, including several hours each week dedicated to manually evaluating student responses [3]-[4]. Under these conditions, manual assessment often becomes inconsistent, time-consuming, and susceptible to fatigue-induced errors. Auto-Evaluation-Systems (AES) have emerged as a practical response to these growing educational challenges by offering scalable, consistent, and efficient mechanisms for evaluating student responses. Rather than depending on simple pattern matching or keyword lookup, AES compares the students responses with teacher's answers based on semantic similarity and evaluate them accordingly.

A core requirement in AES is to develop a model to accurately measure the sentence similarity which helps AES understand textual relationships between student's response and teacher's answer and evaluate them accordingly.Existing tools to measure the sentence similarity, referred as Sentence-Similarity-Tools (SST) estimate semantic similarity by first generating embeddings (vectors) using pretrained transformer models such as BERT and Sentence-BERT (SBERT), and then calculating similarity between these embeddings using cosine similarity measure. SSTs tend to perfom well in handling sentences without negation

words,but they struggle when negation words (such as not, can't, or do not) are present.This is due to embeddings not adequately representing the impact of negation words ,where in the embeddings (of sentences) of both kinds are placed very close to each other in the representation space which high similarity among the sentences. This becomes especially problematic in AES, where a sentence like "The theory is not applicable here" should not be considered similar to "The theory is applicable here" would lead to unfair scoring through AES.In addition to this, there are cases where sentences with identical words differ sharply in meaning based solely on position of the negation word. For example, "The system is not entirely stable" expresses a minor issue, while "The system is entirely not stable" signals a major failure. Such differences arise because the position of negation word relative to other words (present in the sentence),can fundamentally change the meaning of the sentence.

Further, the meaning of negated-sentences (sentences which have negation words) is affected by factors beyond position of the words. The number of negation words that occur in a negated sentence can alter meaning the sentence,the presence of odd counts of negation words often reverse the meaning of negated-sentences, while the occurrence of even counts of negation words may neutralize or dilute the meaning of the negated-sentences. For instance, "not impossible" suggests possibility, while "not not acceptable" only partially removes the negative tone.Context also plays a key role in measuring the similarity, as observed in sentences like: "The method is not accurate for extreme cases", which describes a specific limitation, whereas "The method is not accurate" indicates a general problem.When these aspects are overlooked, SSTs inaccurately treat negation as a superficial change rather than a substantial semantic modification.To address this persistent limitation of current SSTs in capturing the semantic effect of negation,often leading to inaccurate scores through AES.We propose a novel model referred as Negation Aligned Scorer (NAS) which computes a new score in place similarity measure to capture the impact of negation sentences, with a Hybrid Semantic Similarity (HSS) framework that combines embeddings from BERT, RoBERTa, SBERT, and Word2Vec, enhanced through a Bi-Directional LSTM,fine-tuned with a dataset curated for this purpose.

The rest of the paper is organized as follows: Section 2 reviews existing State of art of SSTs with respect to negation sentences. Section 3 describes the HSS framework and Negation Aligned Scorer. Section 4 demonstrates the curation of dataset along with experimental setup and the performance of NAS with baseline approaches with a ablation study, and the paper concludes with Section 5 which focus on the Limitations and Future Directions of this work.

2 Related Works

The research related to SSTs can be categorized into the following themes: semantic similarity, embedding fusion, negation interpretation, and contextual modeling or structured encoders. In this section we review State-of-Art of semantic-similarity ,as categorized above

A. Semantic Similarity and Embedding Models

K. Zhou et al.[5] proposed Word Confusion , a classifier-confusion–based similarity measure based on a encoder based classifier instead of the cosine similarity to better capture contextual and asymmetric semantic relations . Although effective at the word level, it does not address negation or sentence-level polarity shifts

H. Wang et al.[6] introduced a embedding model fine-tuned by a dataset with soft-negative samples to have an appropriate embeddings for the soft-negative sentences. Similarly, G. Wu et al.[7] proposed a model to control false negatives through masking and positive reassignment, whereas W. Liu et al.[8] proposed a model ,to enhance semantic discrimination through fine-tuning with hard negative samples,(which are too complex to be identified). Yet all these models could not handle the negation sentences exhaustively .

Z.Qiu et al.[12] introduced a hybrid semantic embedding by fusing conceptual knowledge with weighted word embeddings to improve domain-aware semantics..F. F. Ijebu et al.[13] proposed soft-cosine and extended-cosine adaptations for transformer embeddings through integration of correlation matrix with cosine similarity measure and a weighting factor for vector magnitudes respectively to overcome cosine saturation. Lastly,X. Li et al.[14] developed an angle-optimized embedding that captures a better embedding, yet these models does not handle negation properly.

B. Negation Sentence Related Models

M. Anschütz[9] proposed NegBLEURT, a negation-aware evaluation metric fine-tuned with negation samples. While their approach improves robustness in general evaluation, it does not go cater beyond lexical overlap of negation sentences. I.Okpala [10] introduced a semantic negation detection framework leveraging word-sense disambiguation and antonymization to restore polarity for negated words. Although effective for sentiment classification, the method handles only surface-level negation and struggles with structural or multi-clause constructions. Complementing these findings, Laverghetta Jr. and Licato.[11] analyzed the transformer based model using an NLI-based diagnostic suite, revealing systematic weaknesses in transformer reasoning across different negation types. Their work, however, provides analysis rather than a modeling solution .

At the syntactic level, F. Winaya et al.[15] proposed a language-specific dependency-parser–guided negation scope detection method ,integrated with XLNet to improve sentiment prediction. Additionally, U. Lal et al.[18] developed a WordNet-based negation-handling mechanism using antonym synsets to adjust polarity, yet this lexicon-driven approach does not generalize to broader semantic tasks involving complex sentence . Finally, S. Biradar et al.[19] explored domain-general negation handling to improve cross-domain sentiment classification, but this method focuses on classification and does not deal with similarity scoring.

C. Contrastive, Siamese Learning Approaches

Siamese architectures and contrastive objectives play a central role in sentence similarity modeling, enabling pairwise comparison of sentence embeddings.Advancing this line of work, N. A. Alnajem et al.[20] proposed a Siamese neural network architecture that integrates sentence transformers with LSTM layers and backward-flow components for similarity detection. Although the model achieves high accuracy, it lacks mechanisms to capture semantic reversals that arise due to negation words. In a related effort, F. Chen et al.[21]introduced SDQKC, a Siamese BERT framework enhanced with dynamic QK co-attention and contrastive learning to estimate short-text similarity. Despite a good performance relative to baseline models,it could not handle negation sentences.

D. Contextual Modeling Using Bi-LSTMs

Bi-LSTM architectures continue to hold relevance for capturing sequential ordering, local context, and compositional semantics in sentence-level tasks. Sumanathilaka TGDK et al.[16] proposed a Bi-LSTM based model, augmented with an attention mechanism and an advanced preprocessing pipeline for robust emotion detection. Yet the approach does not help to identify semantic inversion caused by negation . Similarly, P. Lam et al.[17] proposed an LSTM-based sentence representation model combining C-RNN and MLP components for sequential sentence classification in medical abstracts. However, their architecture is tailored to discourse-level structure and does not explicitly model polarity-sensitive semantics .

M. Mohebbi et al.[22] introduced a deep graph learning framework with SRL-driven graphs and DGNN architecture on top of transformer embeddings, to refine semantic similarity computation. This integration of SRL and graph neural networks yields strong performance improvements, the model does not incorporate negation scope or polarity-aware adjustments in its similarity scoring process .

Summarizing this section, we observe three key limitations related semantic similarity models. First, negation is seldom encoded explicitly at either the embedding or similarity-scoring level, treating negation as single entity rather than a semantic operator that reverses meaning. Second,multi-representation models fail to correlate lexical polarity cues with contextual embeddings. Third, similarity metrics remain predominantly cosine-based, lacking dynamic adjustments required for negation sentences.

3 Methodology

To address the shortcomings discussed earlier, we propose a novel model, called as, "Negation-Aligned-Similarity Scorer" (NAS Scorer) to compute the negation-aligned similarity-score, instead of the usual cosine similarity score. By integrating the fusion of multi-embeddings with Bi-LSTM for contextualization, with

an additional mechanism that dynamically adjusts the similarity score, by considering negation words into account. Our NAS scorer aims to provide a robust and a reliable similarity evaluation for sentence pairs which include negation sentences as well as non-negation sentences. The methodology of NAS Scorer has two sequential phases. The Phase 1 extracts deep, harmonized semantic features for each sentence using a Siamese encoder architecture, while Phase 2 handles the impact of the negation sentences and computes a new score called "negation-aligned-similarity score" by training a Bi-LSTM with a dataset that is curated for this purpose. Together, these two phases as shown in Figure 1, enable the NAS scorer to account for negation-driven semantic shifts that traditional embedding-based SSTs frequently overlook.

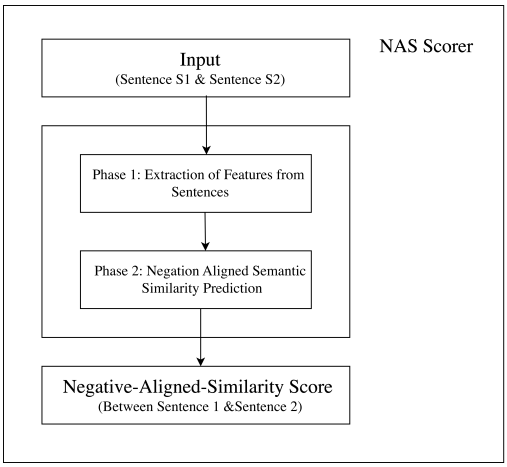

Figure 1: Flow of Negation-Aligned-Similarity Scorer

Each phase of the NAS Scorer consists of deterministic and learnable transformations that progressively compare the two input sentences S_1 and S_2 and compute a negation-aligned similarity score in the range $[0, 100]$. Because the model relies on supervised learning to capture how negation patterns influence perceived semantic similarity, it is essential to ground these transformations in a dataset that explicitly encodes such variation. We therefore first describe the Negation-Sentence-Similarity Dataset (NSSD) which was specifically curated for this purpose, which provides the annotated sentence pairs and negation-aware signals used throughout model training.

3.1 Dataset Description

Since our work is focused towards AES,we have created a dataset Negation-Sentence-Similarity-Dataset (NSSD) consists of sentence pairs spanning four conceptual domains. Each sample contains two sentences, a human annotated similarity score, negation-word counts for both sentences, and a categorical label describing the type of negation variation. Table 1 summarizes the domain

coverage.

Table 1: Domains Represented in the Negation-Sentence-Similarity Dataset (NSSD)

Domain	**Class**
Operating Systems	I
Databases	II
Computer Networks	III
Machine Learning	IV

Each instance in NSSD is represented as a structured record containing all information necessary for negation-aware similarity modelling as represented in Table 2. Every sample includes a domain label indicating whether the pair belongs to Operating Systems, Databases, Computer Networks, or Machine Learning, followed by the two sentences that form the pair. Alongside the raw text, the dataset stores a similarity score that reflects the semantic relationship after accounting for negation cues. To support explicit negation reasoning, the dataset also records the number of negation words appearing in each sentence, enabling the model to capture the asymmetricity or imbalance in polarity between the pair.

Table 2: Sample Instances from Negation-Sentence-Similarity-Dataset (NSSD)

Class	**Sentence 1**	**Sentence 2**	N_1	N_2	**Variation Type**	**Similarity**
I	The process is running in the background.	The process is not running in the background.	0	1	Single Negation	18.5
II	The query result is not incorrect.	The query result is correct.	1	0	Even Negation	95.3
III	The packet is not unlikely to be not dropped.	The packet will not be dropped.	3	1	Odd Negation	12.2
IV	The boy played because it was not raining.	The boy did not play because it was raining.	1	1	Conjunction–Negation	21.6

Finally, each entry specifies a variation type, such as single negation, sentences with, even or odd negation patterns, or conjunction–negation interactions thereby making the dataset fully transparent and interpretable for downstream analysis.The dataset incorporates several categories of negation transformations, all derived algorithmically using even–odd negation behavior and conjunction-aware meaning shifts. These categories are listed in Table 3.

Table 3: Negation Variation Types Used in NSSD Generation

Variation Type	**Semantic Effect**
Original (no negation)	Baseline meaning
Single Negation (S1 or S2)	Mild polarity shift
Even Negations	Meaning largely preserved
Odd Negations	Meaning inverted or altered
Conjunction–Negation Interaction	Contrastive or contextual shift

3.2 Phase 1: Fusion Embedding and Feature Extraction

Phase 1 converts each input sentence into a fixed-size semantic vector of dimension 256. The phase is organized into two consecutive stages: Stage 1 : Fusion and Stage 2 : Feature Extraction.

All operations described below are applied concurrently to Sentence 1 and Sentence 2; the notation below describes the computation for a generic sentence S and the resulting vectors are denoted f_1 and f_2 for the two sentences of a pair.

Stage 1 : Fusion

We employ a heterogeneous ensemble of pretrained encoders because each contributes a distinct and complementary semantic perspective. BERT [23]provides deep contextualized token-level representations that capture fine-grained, context-dependent meaning. SBERT [24], as a sentence-level transformer optimized for similarity estimation, produces stable embeddings well aligned with semantic comparison tasks. RoBERTa [26]contributes a robust masked language model driven contextual view that often complements the representational patterns learned by BERT. DistilBERT [25], through its knowledge-distillation process, yields embeddings with smoother semantic organization and reduced representational redundancy, supporting more abstract and generalized meaning extraction. Word2Vec [27]contributes classical distributional semantics based on lexical co-occurrence patterns, supplying surface-level lexical structure that transformer models may under-emphasize.

Together, these five encoders reduce the blind spots inherent to any single representation family and produce a richer, more comprehensive fused semantic embedding

For a sentence S we extract one pooled embedding from each encoder and denote them:

$$\begin{aligned} e_B &\in \mathbb{R}^{768} \quad \text{(BERT pooled embedding)}, \\ e_S &\in \mathbb{R}^{384} \quad \text{(SBERT (Mini-LM) sentence embedding)}, \\ e_D &\in \mathbb{R}^{768} \quad \text{(DistilBERT pooled embedding)}, \\ e_R &\in \mathbb{R}^{768} \quad \text{(RoBERTa pooled embedding)}, \\ e_W &\in \mathbb{R}^{200} \quad \text{(Word2Vec mean embedding)}. \end{aligned}$$

The per-model sentence embedding is produced by each encoder independently: for transformer-based encoders ,this is the mean-pooled representation, obtained by averaging token embeddings to produce one vector that captures the overall meaning of the sentence, while for Word2Vec it is the average of the word vectors that occur in the sentence. These five vectors are then combined by concatenation. Concretely,

$$x \;=\; e_B \oplus e_S \oplus e_R \oplus e_D \oplus e_W, \tag{1}$$

where $\oplus$ denotes vector concatenation, and x represents the fused embedding vector that aggregates the semantic information contributed independently by BERT, SBERT, RoBERTa, DistilBERT and Word2Vec into a single high dimensional sentence representation. The concatenation operation stacks the five vectors end-to-end, producing a single fused vector whose dimensionality equals 2888 denoted as D_{in}.

$$x \in \mathbb{R}^{D_{\text{in}}}$$

The purpose of this concatenation is twofold: (i) To preserve all embeddings produced by each encoder without discarding model-specific information, and (ii) To consolidate all semantic information into one vector that serves as input for the NAS's later learning stages. Concatenation therefore serves as a lossless fusion step, leaving it to the later layers to determine how the information should be integrated.

Stage 2 : Feature Extraction

This stage reduces the high-dimensional fused vector x into a compact, sequential representation suitable for contextual summarization, and finally produces the 256-dim sentence embedding f. This stage consists of a learnable projection followed by sequential encoding with a bidirectional LSTM.

First, the fused vector $x \in \mathbb{R}^{D_{\text{in}}}$ is passed through a two-layer latent mapping layer with a pointwise non-linearity. This latent mapping stage implements a dimensionality reduction and harmonization:

$$h \;=\; \text{ReLU}\big(W_p^{(2)}(\,\text{ReLU}(W_p^{(1)}x + b_p^{(1)})\,) + b_p^{(2)}\big). \tag{2}$$

In words, Equation (2) first applies a linear map $W_p^{(1)} : \mathbb{R}^{D_{\text{in}}} \to \mathbb{R}^p$ followed by a ReLU activation, optional dropout, and a second linear map $W_p^{(2)} : \mathbb{R}^p \to \mathbb{R}^p$ with ReLU. In our description p is the chosen projection dimension .We choose $p = 512$ for. The resulting vector h is a compact latent representation of the fused sentence embedding, preserving the key semantic information needed for later processing.Thus the projection performs the numerical reduction

$$D_{\text{in}} \;\longrightarrow\; p$$

($2888 \longrightarrow 512$), and the ReLU activations introduce non-linearity, allowing the latent mapping layer to learn rich, non-linear transformations of the concatenated features rather than merely performing a linear compression.

After passing through this latent mapping layer, the vector $h \in \mathbb{R}^p$ is reshaped into a short sequence to enable sequential modeling. Specifically, h is split into L equal-sized chunks so that

$$h \quad\longrightarrow\quad H \in \mathbb{R}^{L\times C},$$

with $C = p/L$. Here H denotes the reshaped sequence representation of the sentence S.This reshape does not create new information; it simply reorganizes

the p sized vectors of h into L pseudo-timesteps that the subsequent recurrent module can process. In our setup, typical choices are $p = 512$ and $L = 8$ (hence $C = 64$), or equivalently $p = 512$ and $L = 16$ with $C = 32$ depending on the configuration; the implementation enforces that p is divisible by L.

H is then processed by a bidirectional LSTM. The forward LSTM ingests the pseudo-timesteps in order and produces a final forward hidden state $\overrightarrow{h_T} \in \mathbb{R}^{H_s}$, while the backward LSTM ingests the sequence in reverse and produces $\overleftarrow{h_1} \in \mathbb{R}^{H_s}$; Here H_s denotes the per-direction LSTM hidden size (for example $H_s = 128$). These two directional summaries are complementary and jointly encode the full bidirectional context of the sentence. The two directional summaries are concatenated to obtain the final sentence representation:

$$\overrightarrow{h_T}: \text{ taken from the final timestep } T \text{ (forward LSTM)},$$

$$\overleftarrow{h_1}: \text{ taken from the first timestep (backward LSTM)}.$$

$$f \;=\; \overrightarrow{h_T} \oplus \overleftarrow{h_1}, \tag{3}$$

so that

$$f \in \mathbb{R}^{2H_s} \quad \text{(example: } f \in \mathbb{R}^{256} \text{ with } H_s = 128\text{)}.$$

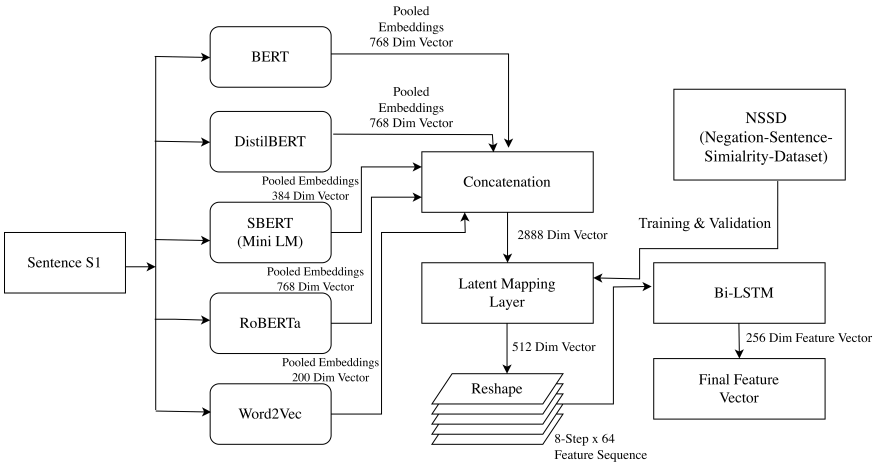

Figure 2: Phase 1: Fusion Embedding and Feature Extraction.

To summarize the numerical flow for a single sentence S : the per-model pooled vectors concatenate to a fused vector $x \in \mathbb{R}^{D_{\text{in}}}$, a learnable projection reduces and harmonizes this to $h \in \mathbb{R}^p$, h is reshaped into L pseudo-timesteps $H \in \mathbb{R}^{L \times C}$ (example $L = 8$, $C = 64$), the BiLSTM processes the sequence and produces two directional summaries of size H_s each, and their concatenation yields the 256-dimensional sentence embedding f:

$$2888 \;\longrightarrow\; 512 \;\xrightarrow{\text{reshape}}\; 8 \times 64 \;\xrightarrow{\text{BiLSTM}}\; 128 + 128 \;=\; 256.$$

The complete sequence of operations described above is illustrated in Figure 2.

Importantly, all parameters involved in this transformation from the latent mapping layer to the BiLSTM are learned supervisedly using custom-curated NSSD, ensuring that the dimensionality reduction and sequential encoding are optimized specifically for negation-sensitive semantic similarity. The entire Stage 1 (fusion) and Stage 2 (feature extraction) pipeline described above is applied concurrently to Sentence 1 and Sentence 2, producing the pair of final semantic vectors $f_1, f_2 \in \mathbb{R}^{256}$ which become the inputs to Phase 2 .

3.3 Phase 2: Negation-Aligned Similarity Estimation

Phase 2 converts the pair of 256-dimensional sentence embeddings produced in Phase 1 into a unified representation from which a negation-aware semantic similarity score is derived. The phase is organized into three sequential stages: Stage 1 : Semantic-Interaction Composition, Stage 2 : Negation Feature Construction, , Stage 3 : Feature Fusion and Projection, and Stage 4: Regression. Each stage uses the embedding pair (f_1, f_2) from Phase 1, gradually adding contrastive, alignment, and negation-based information until a final negation-aligned-similarity score is produced. All transformations described below are trained end-to-end using the custom-curated Negation-Sentence-Similarity Dataset (NSSD).

Stage 1 : Semantic-Interaction-Based Composition

Phase 1 produces two dense semantic vectors,

$$f_1, f_2 \in \mathbb{R}^{256},$$

representing the contextual interpretations of sentences S_1 and S_2. These vectors arise from a BiLSTM encoder with hidden dimensionality 128 per direction, producing a final 256-dimensional sentence representation. The vectors f_1 and f_2 encode the sentence-level semantics that form the foundation for downstream similarity estimation.

To quantify semantic contrast, the model computes the absolute difference between the two embedding vectors:

$$d = | f_1 - f_2 |.$$

Each component of d expresses how strongly the two sentences diverge along its corresponding semantic dimension, enabling the model to identify areas of disagreement in meaning.To capture semantic alignment, the model also computes the element-wise product:

$$p = f_1 \odot f_2,$$

where the operator $\odot$ denotes element-wise multiplication. Here , p vector exhibits parallel activation patterns, indicating shared semantic characteristics.Together, these four vectors: f_1, f_2, d, and p span the semantic–interaction

from which similarity is derived. Their combined dimensionality is:

$$\underbrace{256}_{f_1} + \underbrace{256}_{f_2} + \underbrace{256}_{d} + \underbrace{256}_{p} = 1024.$$

These four components together form the core similarity feature space.

Stage 2 : Negation Feature Construction

Neural embedding models frequently underrepresent the logical and structural effects of negation. To address this limitation, NAS constructs a compact three-dimensional negation-aware vector $n \in \mathbb{R}^3$ from the tokenized sentences of S1 and S2. Each component of n introduces polarity-sensitive information that semantic embeddings alone do not reliably encode.

The first component measures polarity asymmetry by comparing the negation count of the two sentences:

$$n_1 = \big|\, \mathrm{neg}(S_1) - \mathrm{neg}(S_2) \,\big|.$$

Here, $\mathrm{neg}(S)$ denotes the number of negation words in sentence S. A difference in negation count often signals a difference in semantic polarity.For instance,
S_1: The system is running correctly. (0 negations)
S_2: The system is not running correctly. (1 negation)

$$n_1 = |0 - 1| = 1.$$

The second component captures a parity mismatch:

$$n_2 = \begin{cases} 1, & \text{if } (\mathrm{neg}(S_1) \bmod 2) \neq (\mathrm{neg}(S_2) \bmod 2), \\ 0, & \text{otherwise,} \end{cases}$$

reflecting whether the sentences undergo different patterns of polarity reversal. Odd negation counts typically invert polarity, whereas even counts cancel and preserve it.For instance,
S_1: The task is not complete. (1 negation, odd)
S_2: The task is not impossible to complete. (2 negations, even)

$$n_2 = 1,$$

The third component indicates whether negation-words occupy comparable structural positions in both sentences:

$$n_3 = \begin{cases} 1, & \text{if negations in } S_1 \text{ and } S_2 \text{ occur in comparable positions,} \\ 0, & \text{otherwise.} \end{cases}$$

This factor informs the model whether negation modifies comparable semantic regions across the pair.For instance,

(aligned positions)
S_1: The boy did not pass the exam.
S_2: The boy did not clear the test.

$$n_3 = 1.$$

(misaligned positions)
S_1: The boy did not pass the exam.
S_2: The boy passed the not easy exam.

$$n_3 = 0.$$

Together, the components of n supplement the semantic–interaction space with explicit polarity information that is otherwise implicit or absent in the neural embeddings.

Stage 3 : Feature Fusion and Projection

All semantic, interaction-based, and negation-aware features are merged into a single fused representation. This vector, denoted by z, is constructed by concatenating the semantic embeddings f_1 and f_2 with the interaction features d and p, along with the negation-aware vector n:

$$z = f_1 \oplus f_2 \oplus d \oplus p \oplus n,$$

where $\oplus$ denotes vector concatenation. The vector z therefore contains all information used by the model to predict similarity, bringing together sentence-specific semantics, cross-sentence relational cues, and explicit logical polarity structure.

The dimensionality of the fused representation is :

$$\underbrace{256 + 256 + 256 + 256}_{\text{semantic + interaction}} + \underbrace{3}_{\text{negation}} = \mathbf{1027}.$$

Thus:

$$z \in \mathbb{R}^{1027}.$$

This 1027-dimensional vector forms the complete feature representation to be processed by the regression module described later. The subsequent transformations refine this fused representation and progressively compress it toward a scalar similarity value.

The first transformation produces the hidden vector h_1:

$$h_1 = \mathrm{ReLU}(W_1 z + b_1),$$

where $h_1 \in \mathbb{R}^{512}$, $W_1 \in \mathbb{R}^{512\times 1027}$, and $b_1 \in \mathbb{R}^{512}$. The vector h_1 represents a nonlinear re-encoding of the fused feature space, allowing the model to learn

higher-level combinations of semantic, interaction, and negation cues.A second transformation maps h_1 into a more compact representation:

$$h_2 = \text{ReLU}(W_2 h_1 + b_2),$$

where $h_2 \in \mathbb{R}^{128}$. The vector h_2 captures higher-order feature interactions useful for similarity estimation, and serves as the final hidden representation before regression.

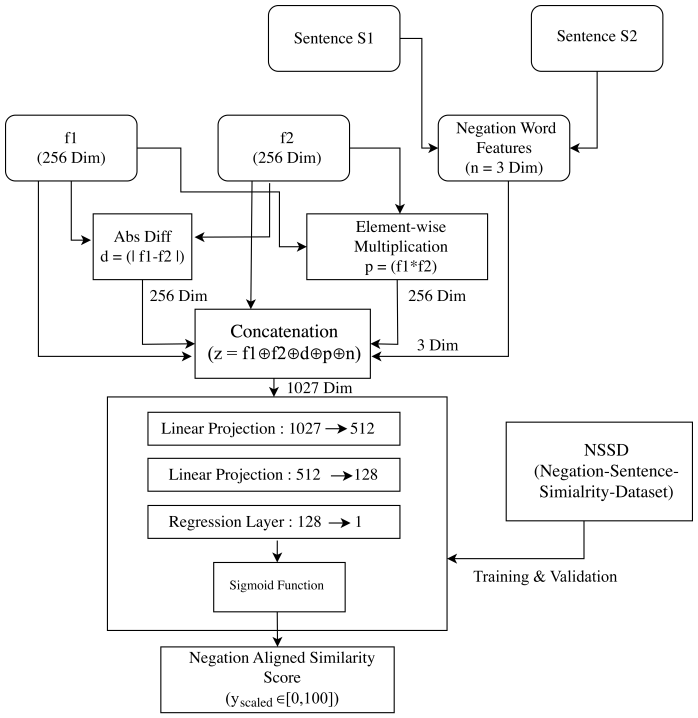

Figure 3: Phase2 : Negation-Aligned Similarity Estimation

Stage 4: Regression

This is the final stage, the contextualized feature vector h_2 is mapped to a scalar similarity prediction through a linear regression layer:

$$o = W_3 h_2 + b_3,$$

where $o \in \mathbb{R}$ represents the model's raw similarity estimate. To obtain a bounded and interpretable output, this value is passed through the sigmoid activation function $\sigma(\cdot)$, which maps real-valued inputs to the range $[0, 1]$:

$$\hat{y} = \sigma(o),$$

where $\hat{y} \in [0, 1]$ denotes the normalized similarity score.

Since the Negation-Sentence-Similarity- Dataset (NSSD) provides human-annotated similarity scores in the range $[0, 100]$, the model output is rescaled as:

$$\hat{y}_{\text{scaled}} = 100 \cdot \hat{y}.$$

The quantity $\hat{y}_{\text{scaled}}$ represents the final negation-aware semantic similarity predicted by the NAS Scorer for a given sentence pair. During training, this predicted value is directly compared with the gold similarity scores provided by NSSD.Thus, NSSD is used to supervise the training of the regression module and all preceding layers. Through this end-to-end learning process, the model learns to jointly weight semantic proximity, contrastive differences, multiplicative alignment patterns, and explicit negation cues to produce the final NAS score, as summarized in the Phase 2 architecture shown in Figure 3.

4 Experimentation

This section explains the computational environment, the frameworks and libraries used at each stage of the proposed Negation-Aligned-Similarity Scorer (NAS), and the hyperparameters adopted during training. All experiments were carried out in Google Colab, which provides a cloud-based Python environment with access to an NVIDIA GPU. GPU acceleration was enabled through CUDA support in PyTorch, allowing faster execution of the multi-encoder embedding extraction, BiLSTM operations, and the training loop.

4.1 Dataset Curation Process

The Negation-Sentence-Similarity Dataset (NSSD) was curated using a structured hybrid pipeline that integrates rule-based automation, large language model (LLM) assistance, and human-in-the-loop validation. The curation pipeline was implemented entirely in Python, with core data handling and CSV input–output operations performed using the Pandas library. The original seed dataset was first loaded and organized using Pandas DataFrames, which enabled systematic generation of new negation-based samples and consistent management of sentence pairs, class labels, and similarity scores.

Negation-aware sentence generation was driven by a combination of linguistic rule systems and LLM-guided rewriting. Regular expression-based pattern matching from Python's re module was used to detect existing negation words (not, no, never) and conjunction markers (and, but, although), as well as to identify valid insertion points for controlled negation placement. For linguistic validation and structural consistency during transformation, lightweight natural language processing support was employed using spaCy or NLTK for tokenization and basic syntactic verification. Large language models were used selectively to ensure that the generated negation variants remained grammatically fluent and semantically coherent after polarity modification.

Similarity score adjustment during curation was implemented using deterministic heuristic functions written in NumPy-based numerical operations. Even-odd negation parity rules and conjunction-negation interaction logic were encoded as conditional transformations that modified the base similarity scores by fixed proportional ranges. These automated score adjustments were subsequently verified through human inspection to ensure alignment with perceived semantic similarity.

Following automated augmentation, every generated sample was manually reviewed to validate negation placement, logical polarity behavior, and the correctness of the adjusted similarity score. This human-machine collaborative workflow ensured that the final dataset maintained both linguistic naturalness and reliable negation-sensitive semantics. Through this controlled expansion process, the dataset was systematically scaled from its original seed set into the final large-scale NSSD. The resulting expansion statistics and overall distribution characteristics are summarized in Table 4.

Table 4: Summary Statistics of the Negation-Sensitive Similarity Dataset (NSSD)

Statistic	**Value**
Total sentence pairs	8,575
Number of domains	4
Average sentence length	13.4 words
Pairs containing negation	6,900
Pairs without negation	1,675

4.2 Model Development and Implementation

The implementation was done in Python 3.10. Core deep learning components, including linear layers, recurrent layers, activation functions, loss functions and the optimization routine, were implemented using the PyTorch library (torch and torch.nn). Pretrained transformer encoders such as BERT, RoBERTa and DistilBERT were loaded using the Hugging Face Transformers library , while sentence-level encoders such as SBERT were instantiated from the Sentence-Transformers framework (sentence_transformers). Classical lexical embeddings were obtained using the Gensim library (gensim.models). Dataset handling, numerical preprocessing and feature manipulation were performed using NumPy and Pandas , while train-validation splits and scaling utilities were provided by scikit-learn (sklearn.model_selection and sklearn.preprocessing). All plotting and diagnostic visualizations were generated using Matplotlib (matplotlib.pyplot).

4.2.1 Phase 1: Fusion Embedding and Feature Extraction

Phase 1 of the NAS Scorer transforms each input sentence into a 256-dimensional semantic vector through two consecutive stages: fusion of embeddings and fea-

ture extraction . For the fusion stage, five pretrained encoders were instantiated. BERT-base-uncased, RoBERTa-base and DistilBERT-base were loaded using the AutoTokenizer and AutoModel classes from the Hugging Face Transformers library. The Sentence-BERT encoder was created using the Sentence-Transformer class from the Sentence-Transformers framework, which directly provides sentence-level embeddings optimized for similarity tasks. Word2Vec embeddings were loaded using Gensim's KeyedVectors interface, giving access to pre-trained 200-dimensional word vectors.For each sentence, tokens were first converted to ids and passed through the corresponding transformer models, and the resulting token-level hidden states were mean-pooled using PyTorch tensor operations (last_hidden_state.mean(dim=1)) to obtain fixed-size sentence embeddings.

For Word2Vec, each token in the sentence was mapped to its vector using the Gensim model, and the sentence-level embedding was obtained by averaging these word vectors using NumPy or PyTorch reduction operations. All five embeddings were finally concatenated along the feature dimension using PyTorch's torch.cat function to form a single fused vector of dimension 2888 for each sentence.

The feature extraction stage in Phase 1 was implemented as a learnable projection followed by sequence modeling using a BiLSTM, all defined in a custom PyTorch nn.Module class. The projection was realised using fully connected layers from the torch.nn module. Concretely, the fused 2888-dimensional vector was first passed through a linear layer implemented with nn.Linear(2888, 512). The projection block included a Rectified Linear Unit activation, implemented explicitly using nn.ReLU() from torch.nn. The projected vector was reshaped using PyTorch's tensor view or reshape operations into a tensor of shape $(\text{batch\_size}, L, C)$ and passed into a bidirectional LSTM, instantiated as nn.LSTM(input_size=C, hidden_size=128, batch_first=True, bidirectional=True).

The final forward and backward hidden states were concatenated using torch.cat to obtain the final 256-dimensional sentence embeddings f_1 and f_2. The entire Phase 1 process, in which fusion relies on frozen pretrained encoders while the projection and BiLSTM layers were trained using the Negation-Sentence-Similarity Dataset (NSSD).

4.2.2 Phase 2: Negation-Aligned Similarity Estimation

Phase 2 takes the pair of 256-dimensional sentence embeddings and the negation features and converts them into a single scalar negation-aligned similarity score in the range $[0, 100]$. The semantic interaction features were implemented using basic PyTorch tensor arithmetic. Given the embeddings f_1 and f_2 as tensors of shape $(\text{batch\_size}, 256)$, the absolute difference vector was computed using torch.abs(f1 - f2), and the element-wise product was obtained using pointwise multiplication f1 * f2. These operations produced the vectors d and p described in the methodology.

Negation features were extracted directly from each sentence pair (S_1, S_2)

rather than being read as precomputed attributes. Each sentence was first tokenized using standard Python string processing utilities, and negation words were identified through rule-based pattern matching against a predefined negation lexicon. The total number of detected negation tokens in each sentence was counted to obtain $\text{neg}(S_1)$ and $\text{neg}(S_2)$. Using these values, the three-dimensional negation-aware vector n was constructed by computing the absolute negation-count difference, the parity mismatch based on odd–even negation counts, and the relative positional alignment of negation words across the sentence pair. The resulting vector $n \in \mathbb{R}^3$ was then converted into a PyTorch tensor using `torch.tensor` and used as an explicit polarity-sensitive feature.

The four vectors f_1, f_2, d and p and the negation vector n were concatenated into a single 1027-dimensional fused feature vector using torch.cat. This fused representation was passed to the regression subnetwork, which was implemented using PyTorch's nn.Sequential container. The first hidden layer was declared as nn.Linear(1027, 512), followed by a ReLU activation implemented as nn.ReLU() and a dropout layer implemented using nn.Dropout(p=0.3). The second hidden layer was defined as nn.Linear(512, 128), again followed by nn.ReLU(). The final output layer was implemented as nn.Linear(128, 1).

After the last linear layer, a Sigmoid activation was applied using nn.Sigmoid() to bound the output between 0 and 1. The resulting value was then multiplied by 100 using standard PyTorch arithmetic to rescale the prediction into the target range $[0, 100]$ expected by the NSSD labels.

4.2.3 Training Procedure and Hyperparameters

The hyperparameters used for training the NAS model are summarized in Table 5.The NSSD dataset was split into training and validation subsets using the train_test_split function from sklearn.model_selection. The resulting NumPy arrays were converted into PyTorch tensors and wrapped into TensorDataset objects from torch.utils.data. Mini-batches were created using the DataLoader class with a batch size of 32. The NAS model was moved to the GPU using model.to(device). The optimizer used for training was AdamW, implemented via torch.optim.AdamW, with a learning rate of 2×10^{-4}.

The loss function was implemented using Mean Squared Error (MSE) loss from the torch.nn module, and the Root Mean Squared Error (RMSE) was computed by applying a square-root operation to the MSE value during evaluation. Training was carried out for 30 epochs. During each iteration, gradients were computed using loss.backward(), and parameters were updated using optimizer.step().To ensure reproducibility, Python's built-in random.seed, NumPy's numpy.random.seed, and PyTorch's torch.manual_seed were all initialized with a fixed seed value.

Table 5: Hyperparameters for Training NAS

Hyperparameters	**Value**
Learning Rate	2×10^{-4}
Hidden Dropout	30%
Hidden Layer Sizes	256, 128
Optimizer	AdamW
Loss Function	Root Mean Squared Error (RMSE)
Epochs	30
Batch Size	32
Activation Function	ReLU
Final Activation	Sigmoid

5 Results

This section presents a comprehensive evaluation of the proposed Negation-Aligned Similarity (NAS) Scorer. The experimental analysis includes four complementary components: (i) Training behavior, (ii) Ablation Study, (iii) LoRA Fine-Tuning for Comparative Analysis, and (iv) Validation.

5.1 Training Behaviour

The training loss and validation RMSE curves shown in Figure 4 illustrate the learning behavior of the NAS Scorer over 30 training epochs and indicate both optimization quality and generalization performance. The training loss reflects how well the model fits the training data, while the validation RMSE measures how effectively it generalizes to unseen sentence pairs. The observed trend corresponds to an ideal learning scenario, with both curves decreasing smoothly and stabilizing at low error values with a small gap between them. Specifically, the validation RMSE drops from 22.67 in the first epoch to 5.5610 at convergence, demonstrating strong generalization capability. The absence of divergence between the curves shows that the model avoids both overfitting and underfitting, confirming that the fused embeddings, BiLSTM contextualization, and explicit negation-aware interaction modeling together form a stable and reliable system.

In addition to convergence analysis, the discriminative behavior of the NAS Scorer was assessed through Receiver Operating Characteristic (ROC) analysis. As shown in Figure 5, the NAS Scorer achieves an Area Under the Curve (AUC) value of 0.9875, which is very close to the ideal value of 1.0. This indicates that the proposed model possesses an exceptionally strong ability to discriminate between semantically similar and dissimilar sentence pairs across all decision thresholds. Since the AUC represents the probability that the model ranks a randomly chosen positive sample higher than a randomly chosen negative sample, a value of 0.9875 confirms that the NAS Scorer performs near optimally in ordering sentence pairs according to their true semantic similarity.

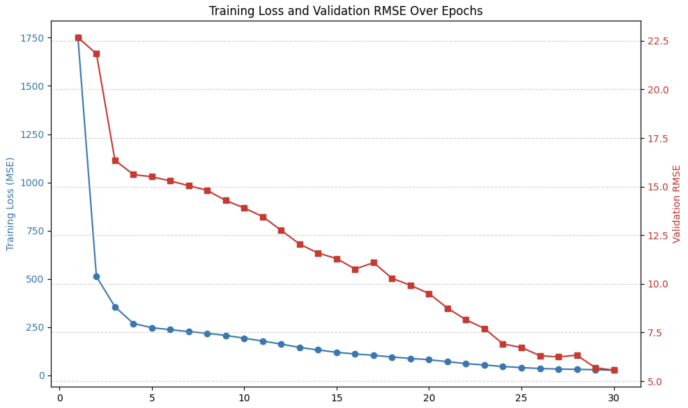

Figure 4: Training Loss vs Validation of NAS Scorer

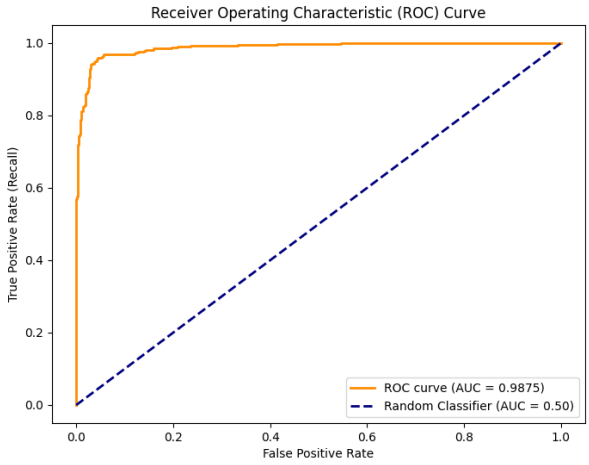

Figure 5: AUC-ROC Curve of NAS Scorer

The ROC curve consistently remains far above the diagonal random classification baseline (AUC = 0.5), demonstrating a high true positive rate even at very low false positive rates. This behavior indicates that the model is able to correctly identify genuinely similar sentence pairs while simultaneously suppressing incorrect similarity assignments. The strong ROC performance directly validates the effectiveness of the negation-aware interaction modeling and the fusion-based semantic encoding strategy, showing that explicit negation alignment significantly enhances binary similarity discrimination beyond conventional embedding-only approaches.

5.2 Ablation Study

As described in the methodology, the proposed NAS Scorer is organized into two phases. Phase 1 consists of Stage 1 (fusion embedding) and Stage 2 (feature extraction), while Phase 2 consists of Stage 1 (Semantic-Interaction-Based Composition), Stage 2 (Negation Feature Construction), and Stage 3 (Feature Fusion and Regression). Since the BiLSTM module in Phase 1-Stage 2 and the negation vector constructed in Phase 2-Stage 2 represent the two most critical learning components of NAS, we perform a stage-wise component ablation by selectively disabling each of these stages to quantify their individual contributions.

Table 6: Stage-wise Ablation Study of Core NAS Components

Model Variant	**RMSE**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
NAS without Phase 2-Stage 2 Negation Vector	10.1842	0.8946	0.9018	0.8732	0.8872
NAS without Phase 1-Stage 2 BiLSTM	12.6395	0.8371	0.8493	0.8016	0.8248
NAS (Proposed: All Phases and Stages)	5.5610	0.9659	0.9718	0.9524	0.9620

We evaluated the contribution of the core NAS stages using the custom-curated Negation-Sentence-Similarity Dataset (NSSD). When Phase 2–Stage 2, which constructs the negation feature vector, is removed, performance declines, with RMSE increasing to 10.1842 and the F1-score decreasing to 0.8872. This indicates that, polarity information contributes meaningfully to similarity estimation. When the negation vector is retained but Phase 1–Stage 2, the BiLSTM-based feature extraction module, is removed, RMSE increases further to 12.6395 and the F1-score decreases to 0.8248, showing that sequential contextual encoding plays an important role in modeling the scope and position of negation. The full NAS Scorer, which includes all stages of Phase 1 and Phase 2, attains the lowest RMSE of 5.5610 and an F1-score of 0.9620, reflecting the combined effect of contextual encoding and negation modeling.

5.3 LoRA Fine-Tuning for Comparative Analysis

To evaluate the effectiveness of the proposed NAS Scorer against strong transformer based baselines, we perform a comparative analysis using LoRA-fine-tuned DistilBERT and LoRA-fine-tuned SBERT models. DistilBERT is selected due to its lightweight architecture and computational efficiency, making it particularly well-suited for parameter-efficient fine-tuning, while SBERT is chosen as the standard vanilla transformer model widely adopted in traditional Sentence Similarity Tasks (SSTs). Both baselines are first fine-tuned on the NSSD dataset using parameter-efficient Low-Rank Adaptation (LoRA), where only a small set of adapter parameters is updated while the base transformer weights remain frozen. This ensures a fair and stable adaptation of the baselines to the negation-sensitive training distribution.

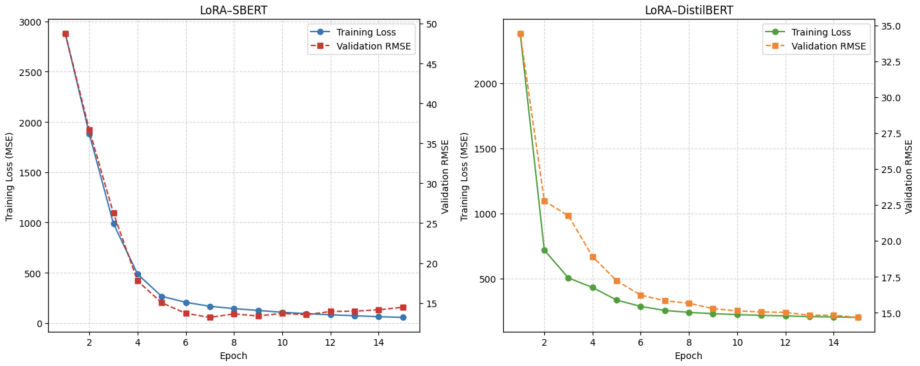

Figure 6: Training Loss and Validation RMSE for LoRA Baselines

The training and validation behavior of the LoRA–SBERT and LoRA–DistilBERT models on NSSD is jointly illustrated in Figure 6. In both cases, the training loss decreases sharply during the initial epochs, indicating effective parameter-efficient optimization. However, the corresponding validation RMSE curves saturate early and converge to relatively high final values when compared to the NAS Scorer. This trend indicates that although LoRA fine-tuning stabilizes training, these transformer-only baselines remain limited in modeling negation-driven semantic shifts due to the absence of explicit negation-aware reasoning mechanisms.

5.4 Validation

Validation of the NAS Scorer was carried out on the STS Benchmark [28] dataset by comparing its performance against LoRA-fine-tuned DistilBERT and LoRA-fine-tuned SBERT, and the corresponding quantitative results are summarized in Table 7.Since the STS Benchmark provides similarity scores on a 0–5 scale, all scores were linearly rescaled to the 0–100 range for consistency with NAS outputs. The proposed NAS Scorer achieves a markedly lower RMSE of 5.5610, substantially outperforming the LoRA–DistilBERT model, which records an RMSE of 14.6680, and the LoRA–SBERT model with an RMSE of 14.4327. This clearly demonstrates the superior regression accuracy of the NAS framework.

Table 7: Performance Comparison of NAS

Model	**RMSE**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
NAS Scorer	5.5610	0.9759	0.9807	0.9604	0.9704
LoRA DistilBERT	14.6680	0.7218	0.6031	0.9491	0.7375
LoRA SBERT	14.4327	0.8353	0.7760	0.8434	0.8083

In terms of classification performance after thresholding, the NAS Scorer attains the highest accuracy of 0.9759 and an F1-score of 0.9704, indicating

highly reliable similarity discrimination. Although the LoRA–DistilBERT baseline achieves a high recall of 0.9491, its significantly lower precision of 0.6031 reveals a strong tendency to overestimate similarity by misclassifying dissimilar sentence pairs as similar. The LoRA–SBERT model exhibits a more balanced precision–recall profile with values of 0.7760 and 0.8434, respectively, but still remains consistently inferior to the NAS Scorer across all reported evaluation metrics.

To further analyze the behavior of different models under structured negation, we conduct a sentence-level similarity comparison using carefully selected sentence pairs from the STS Benchmark dataset that explicitly exhibit different types of negation variations. The similarity scores for NAS, LoRA–DistilBERT, LoRA–SBERT, and vanilla SBERT are reported in Table 8. For the transformer-based baseline models (LoRA–SBERT, LoRA–DistilBERT, and vanilla SBERT), sentence embeddings are first extracted and the similarity is then computed using the cosine similarity between these embedding vectors, which is the standard procedure followed in traditional SSTs. This setup enables a direct and fair comparison between the proposed negation-aware scoring mechanism in NAS and conventional embedding-based similarity estimation.

Table 8: Negation-Aware Similarity Comparison Across Models

Variation Type	**Sentence 1**	**Sentence 2**	**NAS**	**LoRA–DistilBERT**	**LoRA–SBERT**	**SBERT**
Original(No Negation)	A man is playing the piano on the stage.	A person is performing music on a stage.	94.6	94.8	93.2	96.4
Single Negation	The child is eating the ice cream.	The child is not eating the ice cream.	12.3	34.7	41.2	80.9
Even Negation	The task is not impossible to complete.	The task is possible to complete.	91.5	77.2	81.6	84.3
Odd Negation	The proposal is not unlikely to be not rejected.	The proposal will be rejected.	9.8	39.1	46.7	67.4
Conjunction–Negation	The boy played because it was not raining.	The boy did not play because it was raining.	18.7	42.3	47.9	73.5

In the original (no negation) case, all models assign high similarity scores, indicating correct semantic alignment under standard conditions. In the presence of single and odd negation, NAS produces very low similarity scores (12.3 and 9.8), accurately reflecting polarity reversal, whereas SBERT and the LoRA-based models continue to assign relatively high scores, indicating systematic overestimation under negation. For the even negation case ("not impossible" vs. "possible"), NAS assigns a high similarity score (91.5), correctly capturing negation cancellation, while the baselines show only partial sensitivity. In the conjunction-negation example, NAS again assigns a low similarity score (18.7), capturing the combined effect of negation and context, whereas the other models remain inflated. Overall, these results indicate that while cosine similarity serves as a standard mechanism for measuring coherence between sentence embeddings, the primary limitation lies in the embeddings themselves, which fail to adequately capture the contextual scope, positional influence, and count-based effects of negation, whereas NAS is designed to account for these factors to achieve robust negation-aware semantic behavior.

6 Conclusion

In conclusion, this work presents a Negation-Aligned Scorer (NAS) implemented through Hybrid Semantic Similarity (HSS) framework for robust similarity estimation under negation-rich conditions. By combining multi-embedding fusion, BiLSTM-based contextual encoding, and explicit negation-aware interaction features, the proposed model achieves high regression accuracy and strong classification performance, consistently outperforming LoRA-fine-tuned DistilBERT and SBERT baselines across all evaluation metrics. These results confirm the importance of explicitly modeling negation and contextual sequencing for reliable semantic similarity estimation.Despite its effectiveness, the proposed approach has certain limitations. The reliance on multiple transformer encoders introduces higher computational and memory overhead, which may constrain real-time deployment in resource-limited environments. In addition, although the NSSD dataset is systematically curated and human-verified, it may not fully capture the wide range of pragmatic and discourse-level negation phenomena present in naturally occurring text.

Future work will focus on improving scalability through knowledge distillation and lightweight fusion strategies to reduce inference cost, expanding the dataset to include more nuanced and context-dependent forms of negation, and evaluating cross-domain generalization to assess robustness across diverse linguistic settings. These directions aim to advance practical, scalable, and semantically faithful negation-aware similarity modeling for real-world NLP applications.

Data and Code Availability: All code and Dataset used in this study are publicly accessible at: https://github.com/RohitMahesh2004/Hybrid-Deep-Learning-for-Sentence-Similarity-with-Emphasis-on-Negation-Handling.

For reference and secondary evaluation, we also used the STS Benchmark dataset, which is available at https://huggingface.co/datasets/mteb/stsbenchmark-sts.

Acknowledgement

This work was supported by the Ministry of Electronics and Information Technology (MeitY),Government of India,under the project sanctioned with number L-14011/4/2022-HRD.

References

[1] UNESCO, *Global Report on Teachers: What You Need to Know*. Paris, France: UNESCO, 2025.

[2] OECD, *Class Size and Student–Teacher Ratios*. Paris, France: OECD Publishing, 2024.

[3] OECD, "How much time do teachers spend on teaching and non-teaching activities?," OECD Publishing, Paris, France, 2015.

[4] Education Week Research Center, *How Teachers Spend Their Time*. Bethesda, MD, USA: Editorial Projects in Education, 2022.

[5] K. Zhou, H. Gao, S. L. Chen, D. Edelstein, D. Jurafsky, and C. Shani,"Rethinking Word Similarity: Semantic Similarity through Classification Confusion,"in *Proc. 2025 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Albuquerque, NM, USA, 2025, pp. 5803–5817.

[6] H. Wang, Y. Li, Z. Huang, Y. Dou, L. Kong, and J. Shao, "SNCSE: Contrastive Learning for Unsupervised Sentence Embedding with Soft Negative Samples," 2022. doi: 10.48550/arXiv.2201.05979.

[7] G. Wu, "FNCSE: Contrastive Learning for Unsupervised Sentence Embedding with False Negative Samples," in *Proc. SPIE 12719, Second International Conference on Electronic Information Technology (EIT 2023)*, 15 Aug. 2023, Art. no. 127194R, doi: 10.1117/12.2685528.

[8] W. Liu, Z. Yang, C. Li, Z. Hong, J. Ma, Z. Liu, L. Zhang, and F. Huang, "HNCSE: Advancing Sentence Embeddings via Hybrid Contrastive Learning with Hard Negatives," 2024. doi: 10.48550/arXiv.2411.12156.

[9] M. Anschütz, D. M. Lozano, and G. Groh, "This is not correct! Negation-aware Evaluation of Language Generation Systems," in *Proc. 16th Int. Natural Language Generation Conf. (INLG)*, Prague, Czechia, 2023, pp. 163–175.

[10] I. Okpala, G. R. Rodriguez, A. Tapia, S. Halse, and J. Kropczynski, "A Semantic Approach to Negation Detection and Word Disambiguation with Natural Language Processing," in *Proc. 2022 6th Int. Conf. on Natural Language Processing and Information Retrieval (NLPIR '22)*, New York, NY, USA, 2023, pp. 36–43, doi: 10.1145/3582768.3582789.

[11] A. Laverghetta Jr. and J. Licato, "Developmental Negation Processing in Transformer Language Models," in *Proc. 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vol. 2: Short Papers, Dublin, Ireland, 2022, pp. 545–551.

[12] Z. Qiu, G. Huang, X. Qin, Y. Wang, J. Wang, and Y. Zhou, "A Hybrid Semantic Representation Method Based on Fusion Conceptual Knowledge and Weighted Word Embeddings for English Texts," *Information*, vol. 15, no. 11, p. 708, 2024, doi: 10.3390/info15110708.

[13] F. F. Ijebu, Y. Liu, C. Sun, and P. U. Usip, "Soft Cosine and Extended Cosine Adaptation for Pre-trained Language Model Semantic Vector Analysis," *Applied Soft Computing*, vol. 169, Jan. 2025, doi: 10.1016/j.asoc.2024.112551.

[14] X. Li and J. Li, "AoE: Angle-optimized Embeddings for Semantic Textual Similarity," in *Proc. 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Bangkok, Thailand, 2024, pp. 1825–1839.

[15] F. Winaya and A. S. Girsang, "Negation Handling on XLNet Using Dependency Parser for Sentiment Analysis," in *Proc. 2024 Int. Conf. on Intelligent Cybernetics Technology & Applications (ICICyTA)*, Bali, Indonesia, 2024, pp. 251–256, doi: 10.1109/ICICYTA64807.2024.10913246.

[16] Sumanathilaka TGDK, V. Selvarai, U. Raj, V. P. Raiu, and J. Prakash, "Emotion Detection Using Bi-directional LSTM with an Effective Text Preprocessing Method," in *Proc. 2021 12th Int. Conf. on Computing Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, 2021, pp. 1–4, doi: 10.1109/ICCCNT51525.2021.9579844.

[17] P. Lam, L. Pham, H. Tang, M. Seidl, M. Andresel, and A. Schindler, "LSTM-based Deep Neural Network With a Focus on Sentence Representation for Sequential Sentence Classification in Medical Scientific Abstracts," 2024, pp. 219–224, doi: 10.15439/2024F5872.

[18] U. Lal and P. Kamath, "Effective Negation Handling Approach for Sentiment Classification Using Synsets in the WordNet Lexical Database," in *Proc. 2022 First Int. Conf. on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, Trichy, India, 2022, pp. 01–07, doi: 10.1109/ICEEICT53079.2022.9768641.

[19] S. Biradar, G. T. Raju, and K. M. Divakar, "Negation Handling and Domain Generalization in Sentiment Analysis using Machine Learning Models," in *Proc. 2024 Int. Conf. on Knowledge Engineering and Communication Systems (ICKECS)*, Chikkaballapur, India, 2024, pp. 1–4, doi: 10.1109/ICKECS61492.2024.10616885.

[20] N. A. Alnajem, M. Binkhonain, and M. S. Hossain, "Siamese Neural Networks Method for Semantic Requirements Similarity Detection," *IEEE Access*, vol. 12, pp. 140932–140947, 2024, doi: 10.1109/ACCESS.2024.3469636.

[21] F. Chen, Z. Zhang, Y. Jia, *et al.*, "Research on the Similarity Calculation of Short Text in the Terminology Domain Based on Siamese BERT Model," *Scientific Reports*, vol. 15, p. 36954, 2025, doi: 10.1038/s41598-025-20908-8.

[22] M. Mohebbi, S. N. Razavi, and M. A. Balafar, "Computing Semantic Similarity of Texts Based on Deep Graph Learning with Ability to Use Semantic Role Label Information," *Scientific Reports*, vol. 12, p. 14777, 2022, doi: 10.1038/s41598-022-19259-5.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc.*

2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Minneapolis, MN, USA, June 2019, pp. 4171–4186, doi: 10.18653/V1/N19-1423.

[24] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, 2019, pp. 3982–3992.

[25] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter," *Proc. NeurIPS Workshop on Energy Efficient Machine Learning and Cognitive Computing*, Vancouver, Canada, 2019.

[26] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv preprint arXiv:1907.11692*, 2019, doi: 10.48550/arXiv.1907.11692.

[27] Mikolov, Tomas Chen, Kai Corrado, G.s Dean, Jeffrey. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of Workshop at ICLR. 2013.

[28] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 Task 1: Semantic Textual Similarity—Multilingual and Cross-Lingual Focused Evaluation," *in Proc. 11th Int. Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, BC, Canada, Aug. 2017, pp. 1–14.