Duluth at SemEval-2026 Task 4: A Hybrid Approach to Narrative Similarity using Bi-Encoder Embeddings with Cross-Encoder Tie breaking using Learned Weights

Maxwell Bevers, Aidan Carlson, Ted Pedersen
Department of Computer Science
University of Minnesota
Duluth, MN 55812 USA
bever140@d.umn.edu, carl6746@umn.edu, tpederse@d.umn.edu

Abstract

We present a hybrid system for SemEval-2026 Task 4 on Narrative Similarity. Our approach decomposes the stories into four narrative components: theme, plot, emotion, and outcome. Each component is then encoded using a bi-encoder (all-mpnet-base-v2), and cosine similarities are combined through a learned pairwise ranking model. When similarity scores between candidate stories fall within a small margin of error, a cross-encoder (ms-marco-MiniLM-L-6-v2) is used as a tie-breaker. Our final system achieves 58.5% accuracy on the official test set, placing us at 39th overall. Error analysis reveals that the system struggles with complex themes, multiple protagonists, and contrasting outcomes.

1 Introduction

The Narrative Story Similarity and Narrative Representation Learning task was designed to expand research in narrative similarity, and it emphasizes symbolic systems. The sets of triplets provided were generated from Wikipedia story summaries (in English) with the most similar candidate story labeled. The overview paper fully outlines Task 4 for the 2026 competition (Hatzel et al., 2026).

Our system sought to determine narrative similarity between stories using prebuilt models and our extracted narrative components. We use a bi-encoder to create embeddings for each of our components. These components are then combined using a weighted sum of cosine similarities. We use a learned ranker to determine the optimal weights for this method. We incorporate a cross-encoder to deal with ties when they occur.

Against the official test data, our system had an accuracy of 58.5%, placing us at a rank of 39^{th} overall. Our qualitative analysis shows that our model performs well on stories that have singular protagonists and straightforward causal plots. However, it also shows that our model struggles with situations that have more complex themes, multiple influential characters, and contrasting outcomes.

2 Background

Task 4 of the SemEval 2026 workshop defined a concise NLP exercise, and the organizers welcomed models built in any style. We participated in Track A, and we used the included development and test sets of triplets. The development data contained 200 entries, and the test data contained 400. The data was generated from English story summaries from Wikipedia and annotated manually. Our hybrid model produced its own labels for the most similar story of each triplet.

BERT was released in 2018, and has been applied across a range of NLP problems since. It established a system built on the Transformer architecture (Vaswani et al., 2017), and its use of deep bidirectional pretraining extracts more semantic context than prior models. This feature makes the model much better across narrative similarity domains.

Other systems have been built on top of BERT to address some of its limitations. MPNet, for instance, seeks to patch the lack of dependencies among predicted tokens in BERT by combining it with XLNet. By combining the advantages of both models, MPNet was shown to outperform both individual systems in tasks such as GLUE and SQuAD (Song et al., 2020).

Cross-encoders specialize in fine grained token level comparison between two texts. The model ms-marco-MiniLM-L-6-v2 is built on a lightweight MiniLM system when compared to BERT based models (Wang et al., 2020). The variant has also been fine-tuned on the MS MARCO dataset (Bajaj et al., 2018).

Our narrative components were inspired by those outlined by the task organizers (Hatzel et al., 2026). As the SemEval-2026 Task 4 overview paper states:

Proceedings of the The 20th International Workshop on Semantic Evaluation (2026), pages 927–931

abstract theme refers to the ideas and motives of the story, course of action refers to the sequence of central events, turning points, etc, and outcomes refer to the results of a story.

3 System Overview

We are using a hybrid approach that implements both a bi-encoder model, all-mpnet-base-v2 (Song et al., 2020), and a lightweight MiniLM cross encoder, ms-marco-MiniLM-L-6-v2 (Wang et al., 2020), fine-tuned on MS MARCO using the Sentence-Transformers framework (Bajaj et al., 2018; Devlin et al., 2019). In order for our system to predict whether the provided story A or B is closer to the anchor text in a provided triplet, the following methods are applied. Chunking is applied to format the data for our bi-encoder. Then, embeddings are created for narrative components and some basic thematic roles. A learned ranker assigns weights to all narrative components based on optimized values after 10 epochs. Finally, the learned weights are multiplied with the cosine similarity of each narrative component. Ties are broken by the cross-encoder.

3.1 Embeddings

For the embeddings used by the bi-encoder, we first split the sentences using a simple regex boundary method, looking for `(?<=[.!?])\s+`. Our bi-encoder model has a maximum token buffer size of 388 tokens. To avoid truncation issues, we implemented a strategy for partitioning the stories into chunks. We iteratively add sentences into the chunk, checking to see if the addition of a sentence will push us above the token ceiling. In the cases where there is still token space available, we insert padding to make the size of each chunk uniform.

Each chunk is then independently encoded with our bi-encoder, and masked mean pooling is used to eliminate the influence of the padding on our final representation. Since each story consists of multiple chunks, we average the chunks to create a single embedding per narrative component that we then normalize. Formally, we can write this as:

$$\mathbf{h}_{story} = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{N_k}\sum_{i=1}^{N_k} h_{ik}\right) \quad (1)$$

where K is the number of chunks that were created from each story and N_k is the number of valid tokens in chunk k, and h_{ik} is the hidden embedding of token i in chunk k. This strategy preserves sentence-level structure while enabling efficient similarity computation over long narratives.

3.2 Narrative Components

Stories are multifaceted. To account for this, we define three major narrative components, namely: events, outcomes, and themes. We also introduce a fourth component, emotions, related to the theme to help distinguish between negative and positive views.

We chose to use simple heuristics to find the emotion, outcome, and plot components using a word list that we expanded using VerbNet (Kipper-Schuler, 2005). To get a start on the specific hit words, we queried the ChatGPT 4.0 model for common words in each category, getting the following:

- EVENT WORDS: kill, die, attack, leave, arrive, fight, escape, lose, win, discover, decide, confront, confess, betray, save
- OUTCOME WORDS: because, therefore, suddenly, finally, as a result, after that, no longer, never again, decided to
- EMOTION WORDS: love, hate, fear, panic, hope, despair, joy, anger, grief, shock, relief, terror, excitement.

Our four narrative components were derived from their relation to these lists. The theme is inferred from the area with the highest narrative intensity. Narrative intensity, in this case, is defined as being the points that have the highest concentration of the other three narrative components. The plot becomes the sentences that contain a word from the expanded Event Words. We also extracted shallow thematic roles from the plot. Emotion is the sentence that contains a word from the expanded Emotion Words. Finally, the outcome was defined as the resulting sentences from taking the last two sentences and assigning the sentences that contain an event word or a causal word. Then, we created embeddings for each of these narrative components.

3.3 Thematic Roles

Due to the simplicity of our method for determining similarities, we wanted to remove some of the bias that was created by proper names and places. To reduce sensitivity to surface-level lexical variation, we introduce shallow thematic role labeling within the plot component. This seeks to provide an abstracted encoding of the entities and actions of

a story . The shallow thematic roles were assigned with spaCy (**?**).

- AGENT: nsubj or nsubjpass,
- PATIENT: dobj, obj, or pobj
- ACTION: VERB

3.4 Learned Ranker

To combine the cosine similarity scores that are computed for each narrative component, we use a learned ranker. Each of these component similarities is given an equal initial weighting that is optimized through a pairwise logistic ranking loss over 10 epochs on the development dataset. The ranker then assigns the optimized weighting for each component based on the given true closer values of our datasets.

3.5 Decision Strategy

Using the weights that were deduced by using the learned ranker, we multiply them by the cosine similarities of their respective narrative component embeddings for both A and B. We formally define this as the following:

$$S_C = \sum_{f \in \mathcal{F}} w_f \cos(\mathbf{h}_{Anchor,f}, \mathbf{h}_{C,f}) \quad (2)$$

Where S_C denotes the final weighted similarity score between the anchor and the candidate story, C, A or B. Our index f is iterating over the elements of the set of narrative components, $\mathcal{F}$, being emotions, outcomes, events, and themes. In the event that the difference between S_A and S_B is within a predefined threshold $\varepsilon > 0$, we compute cross-encoder scores of the anchor and each candidate, choosing the candidate with the higher cross-encoder score.

$$|S_A - S_B| \leq \varepsilon \quad (3)$$

For the sake of clarity, we state the exact decision algorithm.

If $|S_A - S_B| > \varepsilon = 0.01$, we use our prediction $\hat{C} = \arg\max_{C \in \{A,B\}} S_C$. Otherwise, the cross-encoder score determines the final prediction.

4 Experimental Setup

Using the system that we describe above, we targeted our approach on the guidelines set for Track A of the SemEval 2026 Task 4 (Hatzel et al., 2026).

4.1 Data

While the synthetic data had 1897 usable data samples, we decided not to use the data due to the high scores we saw when running the dataset on our base models. For our models, a preliminary run gave us 99.7%, which to us implied that there was probably a structural pattern to the triplets that were generated using LLMs. While there could have been useful information here, we were worried that the structural bias being inserted into our embeddings could skew our results by overfitting to synthetic.

Instead of using the synthetic, we focused on refining our technique using the development data and the sample data. The development data consisted of 200 entries and the sample data consisted of 39 entries. We used the development data for training and the test data as the test set, which had 400 entries. Each of these entries consisted of an anchor narrative and two candidate narratives. The datasets themselves were generated from Wikipedia articles.

4.2 Hardware

All of our experiments were ran on an HP Envy x360 Laptop, which has a 13th Gen Intel Core i5-1335U processor and 8 GB RAM. Due to the limitations of our device, we did all of computations through CPU.

4.3 Model

Having limited experience with Narrative Similarity and NLP-type problems, we decided to use popular, prebuilt models as the core of our system. After exploring some popular models, we chose the bi-encoder, all-mpnet-base-v2 for its ease of generating embeddings that allow us to make cosine similarity comparisons. For our cross-encoder, we decided to use ms-marco-MiniLM-L-6-v2.

4.4 Implementation Details

All experiments were implemented in Python using Hugging Face Transformers[1] for model loading and inference. Neural computation was performed with PyTorch[2], and numerical operations used NumPy[3]. For linguistic preprocessing, we used spaCy[4] with the `en_core_web_sm` model[5].

[1] Hugging Face Transformers v4.56.2, https://github.com/huggingface/transformers

[2] PyTorch v2.8.0, https://pytorch.org

[3] NumPy v2.3.3, https://numpy.org

[4] spaCy v3.8.7, https://spacy.io

[5] spaCy English model en_core_web_sm v3.8.0, https://spacy.io/models/en

5 Results

The result of our model for track A was 58.5%, and we ranked 39^{th} overall. We show an ablation table in table 1, which shows that our model itself did not perform better than the baseline model. However, using the cross-encoder did improve the accuracy of the overall model, but failed to get above the bi-encoder with our narrative components.

System Variant	Accuracy (%)
Bi-encoder (theme only)	60.50
+ Narrative Components	59.75
+ Learned Ranker	56.00
+ Cross-Encoder (Full System)	58.5

Table 1: Ablation results showing the accuracy of the system as components are added.

In an attempt to increase the accuracy of our model post-competition, we tried varying the number of epochs that the learned-ranker went through, which we show in Table 2. We did not see any monumental improvements as we increased epochs, but our model did get above the baseline when we had 25 epochs.

Epochs	Correct (Out of 400)	Accuracy (%)
5	229	57.25
10	**234**	**58.50**
15	239	59.75
20	240	60.00
25	246	61.25

Table 2: In this table we show the results of post submission experiments. We list the accuracies that our model gave and the number of correct answers for varying epoch levels running on our learned ranker. We show the official submitted results in bold in the table.

Exploring what happens at 25 epochs, we recompute a new ablation table, which can be seen in table 3. It is worth noting that for higher epochs, the learned ranker system does show improvement over the baseline bi-encoder. Interestingly, there is a minor reduction in accuracy when we implement our cross-encoder, which may be due to the model not being fine-tuned.

5.1 Error Analysis

We explored the successes and pitfalls of our model qualitatively by focusing our attention on the first 25 triplet pairs. In these cases we noticed that our model does well when there is a single main character, simpler themes, and causal story lines. Our model accurately predicted 15 out of 25 anchors. While our model did a decent job with stories that had these characteristics, we noticed some patterns in the incorrect decisions. Our model appears to struggle with more complex themes, where the theme diverges from the general plot, situations where there are multiple main characters, and contrasting outcomes. We showcase the particular stories of these 25 that we believe suffered from these issues in Table 4

System Variant	Accuracy (%)
Bi-encoder (theme only)	60.50
+ Narrative Components	59.75
+ Learned Ranker	61.50
+ Cross-Encoder (Full System)	61.25

Table 3: Ablation results showing the accuracy of the system with 25 epochs on the learned ranker.

5.1.1 Plot vs Theme

One of the biggest things we noticed was the inability of our model to distinguish between some of the underlying themes and the overall structure of the story. The way that we use our heuristics, the model gets stuck on situations where there is high overlap in the actual words present in each story, or the plots follow a similar flow. It mistakes the structural similarities for thematic similarities.

5.1.2 Multiple Characters

Another issue that our model had was with stories was when there were multiple agents with their own goals. We believe that due to the shallow thematic roles we use, we are not able to differentiate between the side characters and the main characters, which can have an impact on the narrative similarity scores. This is especially the case when the model fails to differentiate between certain outcomes affecting the antagonist vs the protagonist of a story

5.1.3 Contrasting Outcomes

Our model did not put enough weight on the situations where, for example: The narrative structure of story A is very close to the anchor's structure, but the outcome is opposite. In story B, however, it lacks some of the same narrative structure that A has with the anchor, but has an outcome that matches the anchor. We notice in these cases, a hu-

Triplets	2	4	7	8	12	15	17	18	20	21	23
Plot vs Theme	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Multiple Characters	✓		✓			✓	✓				
Contrasting Outcomes	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓

Table 4: Major problem areas in the first 25 Triplets (Anchor, A, B). The checkmarks showcase the areas that we believe were the largest source of inconsistencies for each corresponding triplet in the columns

man reviewer appears to more often go with story B rather than story A. This shows that our model puts too much emphasis on the structure of a story and not enough on the outcomes.

6 Conclusion

Our system utilizes a hybrid approach using a bi-encoder, a learn ranking system for the weights of each component, and a cross-encoder that is used in the event of a tie. Our results show that using a small number of epochs is not beneficial when it comes to implementing a learned ranker. Our results also show us that our methods, are about equal to the modern base models that exist for this style of task.

One of the major ways that we believe we could have improved our model is to take our models and fine-tune them using our own data. In addition to fine-tuning, we also believe that our approach could benefit from a model that is more aligned with what we are doing, as opposed to the base models we used. In addition to this, we want to explore how symbolic linguistic ideas can be further tied to how we can compute narrative similarity. We started tying in some of these ideas using shallow thematic roles and our VerbNet expanded heuristics. We would like to implement a more complex theta role-like mechanism that could further help with some of the bias towards structural similarities. We also noticed, during analysis, that there was some variability in the results of each independent model. In our case, the variability was relatively minor, but we would like to explore averaged voting estimates across multiple runs or even multiple models to increase narrative similarity.

7 Acknowledgments

We would like to thank the organizers of this task for their efforts in making this possible.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hans Ole Hatzel, Ekaterina Artemova, Haimo Stiemer, Evelyn Gius, and Chris Biemann. 2026. SemEval-2026 Task 4: Narrative similarity and narrative representation learning. In *Proceedings of the 20th International Workshop on Semantic Evaluation (SemEval-2026)*, San Diego, CA, USA. Association for Computational Linguistics.

Karin Kipper-Schuler. 2005. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*.