Rating Plausibility of Word Senses in Ambiguous Narratives

Cioată Ioana-Larisa
Vlad Adriana

January 2026

Abstract

We address SemEval 2026 Task 5, which focuses on predicting graded plausibility scores for word senses in ambiguous narrative contexts. We started from a BERT-based regression baseline enhanced with textual data augmentation and proposed an alternative formulation inspired by Natural Language Inference (NLI), explicitly modeling semantic compatibility between a narrative and a candidate sense definition. Experimental results on the development set show consistent improvements over the baseline across the official evaluation metrics.

1 Introduction

Lexical ambiguity represents a challenge for natural language understanding systems, especially in narrative contexts where the interpretation of a word's meaning is often not something categorical, but graded, with uncertainty also present at the level of human judgment. Task 5 of SemEval 2026 focuses on this problem, requiring the estimation of the plausibility of a particular sense within a given narrative context.

In this work, we explored neural regression models for this task. We first established a BERT-based baseline with augmented data. Building on this, we investigated an alternative NLI-style formulation, where the compatibility between a narrative context and a candidate sense definition was used.

2 Task Description

The task consisted of predicting a plausibility score between 1 and 5 for a candidate sense of an ambiguous word appearing in a short narrative. Each instance included a narrative context, a candidate meaning, and multiple human ratings reflecting graded plausibility rather than a boolean label. Evaluation was performed using Spearman correlation and accuracy within the standard deviation of the human ratings.

More specifically, each instance contains the following fields:

- **homonym**: the ambiguous word appearing in the narrative.
- **judged_meaning**: the candidate sense definition whose plausibility is being evaluated.
- **precontext**: three sentences preceding the sentence containing the homonym.
- **sentence**: the sentence in which the homonym occurs.
- **ending**: the narrative continuation following the target sentence.
- **choices**: individual plausibility ratings assigned by human annotators on a 1-5 scale.
- **average**: the mean of the human plausibility ratings.
- **stdev**: the standard deviation of the human ratings.

- **nonsensical**: a list of boolean values indicating whether annotators considered the sense nonsensical in context.
- **sample_id**: a unique identifier for the instance.
- **example_sentence**: an example sentence illustrating the candidate sense.

Of which the test data excludes the choices, average, stdev and nonsensical fields.

The main official rating is an accuracy score which counts predictions within stdev of average, or less than 1 away from the average, as correct. Spearman correlation is used as a secondary metric.

There are 2280 train instances, many of which use the same homonym, but with different meaning, or the same homonym and meaning but with different contexts. The validation set contains 588 instances, and the test 930.

3 Related Work

One related area of research is gloss-based word sense disambiguation, where each candidate sense of a word is represented by a textual definition (gloss) and evaluated with respect to its surrounding context. A representative example is GlossBERT [2], which reformulates word sense disambiguation as a sentence-pair classification task. For each candidate sense, the model jointly encodes the sentence containing the target word and the corresponding gloss using a BERT-based cross-encoder, and predicts whether the gloss matches the contextual usage of the word.

By converting sense selection into a context–gloss comparison problem, this approach enables direct interaction between the two inputs through self-attention within a single Transformer encoder. While GlossBERT focuses on categorical sense selection, the same modeling strategy can be naturally extended to settings where sense interpretations are evaluated on a graded scale rather than selecting a single correct label.

Our approach is also related to sentence-pair modeling methods developed in the context of Natural Language Inference, particularly those based on Transformer encoders such as BERT [1]. BERT was pretrained and evaluated on NLI tasks, where models jointly encode a premise and a hypothesis to reason about their semantic relationship. This premise–hypothesis structure provides a convenient template for comparing a narrative context with a candidate sense description, while differing from standard NLI in that our model predicts continuous plausibility scores instead of discrete entailment labels.

4 Methodology

4.1 Baseline Model

Our baseline employed *bert-base-uncased* as a regression model for predicting plausibility scores. The narrative context and the candidate sense definition were encoded as a sentence pair, enabling the model to learn implicit semantic associations between the story and the proposed meaning. Training used a Huber loss to mitigate the impact of outliers in noisy human annotations and was optimized using AdamW. Seeing how limited the training data was, we decided to incorporate data augmentation from the very first versions. To augment the data, we incorporated English-German-English back-translation and Easy Data Augmentation (EDA) techniques, namely synonym replacement, word deletion, and word swapping. These augmentations were only performed on precontext and ending, as to not impact the appearance of the homonym in the core sentence. Additionally, in this version, the example sentences associated with meanings were not used. The best checkpoint was selected based on Acc@SD/1 on the development set.

Back-translation was performed using Google Translate, via the python library *translators*. This part of the augmentation was done in a separate script, and the resulting new data saved in a separate file to avoid recomputation every run.

The exact hyperparameters were:

Hyperparameter	Value	Description
Model	bert-base-uncased	Pretrained BERT model used as regression backbone
Batch size	32	Number of samples processed per training step
Epochs	10	Number of training epochs
Learning rate	2×10^{-5}	Learning rate for AdamW optimizer
Max sequence length	256	Maximum number of tokens per input pair
$p_{\text{use\_bt}}$	0.5	Probability of using back-translated data
p_{EDA}	0.5	Probability of applying EDA augmentation
EDA intensity	0.25	Proportion of tokens affected by EDA
EDA mode	random	Random selection of EDA operations
Synonym replacement	0.4	Probability of synonym replacement in EDA
Word deletion	0.3	Probability of word deletion in EDA
Word swapping	0.3	Probability of word swapping in EDA
Optimizer	AdamW	Optimizer used for training

4.2 Notable attempts

This section summarizes several alternative directions we explored during development.

4.2.1 Hyperparameters

Initially we focused on further fine-tuning our chosen bert model by exploring other hyperparameters. Below are our finding regarding the impact of these parameters.

Regarding EDA, we found the best probability distribution between the three operations to be 70% synonym replacement, 15% deletion and 15% swap. We also found that a p_eda of 20% and intensity of 10% worked best in most scenarios, though there were a few setups where lower or even no p_eda had similar results.

For learning rate, on every attempt to use a value higher than $2e-5$ we found the model struggled to converge and ultimately underfitted. Lowering the learning rate only slowed down training. Implementing a learning rate scheduler did not improve results in any case.

Increasing batch size to 64 did not improve run time, but did lower results. Due to time constraints, we did not test the effects of smaller batch size on this pipeline.

We briefly experimented with dropout, as the model seemed to be overfitting most of the time. While these experiments did result in different training accuracies, results on validation did not improve.

Regarding weight decay and label smoothing, these seemed to be the most effective small changes we could have made. More specifically, we found $1e-4$ weight decay to help slow down overfitting, and label smoothing of 0.1 to further improve validation results. Versions with these additions performed a consistent 1-5% better than those without.

We experimented both with clipping predictions to (1, 5) range and to (0.5, 5.5) range, which would allow a +- 0.5 variation for each integer label. In practice, however, this did not seem to impact results.

Increasing max length to 512 very slightly and inconsistently improved results, while significantly increasing training time. Lowering max length to 128 lead to underfitting.

Variations and on huber_delta had no significant effect.

4.2.2 Story and meaning

Initially the model used precontext+sentence+ending (full_story) - judged_meaning as story-meaning pairs to train. We decided to experiment with this as well.

Adding example_sentence to judged_meaning (full_meaning) slightly worsened results. Adding the homonym at the beginning of the story significantly worsened results.

What we thought was most promising was giving the model the chance to train on multiple types of pairs. meaning using either the full story or the example_sentence as story input, and either full_meaning or judged_meaning as meaning. Unfortunately, this only seemed to confuse the model.

After many variations on what pairs the model trains on and how they are constructed, we concluded none of them were better than what we used initially.

4.2.3 Offline augmentation

To address overfitting problem, we tried more heavy augmentations.

First we decided to expand our back translation to four languages (german, italian, spanish, french). To not perform too many calls on google translate's API, we change our translation method to a local one based on *MarianMT* models from the Helsinki-NLP project. This improved results, but only slightly, and significantly increased training time.

We also tried performing various augmentations through $sentence-transformers/all-mpnet-base-v2$. After many attempts due to how promissing this idea seemed, however, we concluded they had minimal impact on results. Another attempt, which also had only a minimal impact, involved prompt engineering for data augmentation using the microsoft/Phi-3-mini-4k-instruct model.

4.2.4 Model changes

Lastly we decided to try other bert variations.

Roberta-base required no additional changes to our code and immediately improved results to 68.5%.

Using roberta-large lead to OOM issues. Trying to solve them by reducing batch size to 16 did work, but gravely impacted results. What also worked, but did not lead to improved results, was setting max_length to 192. Our attempts to use microsoft/deberta-v3-base went similarly.

During these model changing attempts we tried to further optimize the code to accomodate the use of larger models. As such, we added automatic mixed precision. We also implemented a small warmup for models we found function well with it.

4.3 Final Method

Our final method reformulated the task as a Natural Language Inference (NLI)–style regression problem. Instead of directly predicting plausibility scores from embeddings, we explicitly modeled the semantic compatibility between a narrative context and a candidate word sense.

The premise consisted of the full narrative context, more exactly the concatenation of the precontext, ambiguous sentence, and ending, like full_story in previous version. The hypothesis was constructed as a natural language statement describing the homonym's candidate sense, combining its definition with the example sentence.

We used a *DeBERTa-v3-large-mnli-fever-anli-ling-wanli* model pretrained on multiple NLI datasets and fine-tuned it as a regression model with a single output neuron. Plausibility scores derived from aggregated human ratings were linearly normalized during training and mapped back to the original 1-5 scale at inference time. To accommodate the large model under limited GPU memory, training used mixed-precision computation, gradient accumulation, and gradient checkpointing. Optimization was performed using the Adafactor optimizer with linear learning rate warmup.

The best checkpoint was selected based on the average of the Spearman correlation and Acc@SD/1, which in our case had 71.56% spearman and 82.31% accuracy scores.

5 Experimental Setup

All models were trained on Kaggle or Google Colab using NVIDIA T4 GPUs. Data loading used PyTorch DataLoaders with 24 worker processes, persistent workers, and pinned memory.

Tokenization and model initialization were performed using Hugging Face's AutoTokenizer and AutoModel classes. A batch size of 32 was used almost everywhere. For the best model, a batch size of 8 with gradient accumulation over 2 steps was used. All models were trained with a learning rate of 2×10^{-5}. Most experiments used the AdamW optimizer, while the final model employed Adafactor.

Models were ran for 10-30 epochs, as experiments and thus convergence speed varied greatly.

6 Results

In addition to the baseline and final models described earlier, we evaluate several intermediate configurations on the development set using the official metrics. These variants explore changes along three main dimensions: the choice of pretrained backbone, the use of additional regularization, and the strength of data augmentation.

Specifically, one configuration replaces the baseline backbone with RoBERTa, while another introduces weight decay and label smoothing during training. All the models so far excluding the best one combine EDA and back-translation. A further variant explores heavier data augmentation by extending back-translation to four languages using local MarianMT models, resulting in slightly improved performance at the cost of increased training time.

Table 2 presents the performance of the two models on the development data using the official evaluation metrics. As can be observed, the DeBERTa NLI variant achieves the best results on both evaluation metrics.

Model	**Epochs**	**Time**	**Spearman**	**Acc@SD/1**
BERT baseline	10	26m 39s	0.27	0.61
RoBERTa-based	30	1h 21m 14s	0.42	0.68
BERT + weight decay + label smoothing	10	26m 12s	0.33	0.65
RoBERTa + heavy augmentation	7	1h 19m 28s	0.45	0.7
DeBERTa NLI (final)	10	45m 23s	**0.71**	**0.82**

Table 2: Development data performance of the evaluated models

The leaderboard requires integer predictions. With this added condition, our best model had a 76.7% accuracy on validation and a score of 0.72 on our test data prediction submission.

7 Ablation Study

We conducted a simple ablation study to compare the baseline regression setup with the final NLI-style formulation. The baseline needed significantly augmented data and fine tuning to improve, while the second performed very well on the raw training data. They did, however, end up using comparable amounts of memory, as a result of the second model being much bigger.

The baseline predicts plausibility scores using a standard regression head over sentence-pair embeddings, whereas the final system reformulates the task as a Natural Language Inference–style regression problem and employs a DeBERTa model pretrained on multiple NLI datasets. As shown in Table 2, the final model achieves higher Spearman correlation and Acc@SD/1 scores.

Notably, the NLI-based model reaches strong performance already in the early training epochs, suggesting a better initial alignment between the pretrained model and the task formulation. This observation indicates that the performance gains are likely influenced by both the change in task formulation and the use of an NLI-pretrained encoder, rather than by data augmentation alone.

8 Discussion and Conclusion

In this work, we compared several modeling strategies for predicting graded plausibility scores of word senses in ambiguous narratives. Starting from a BERT-based regression baseline, we analyzed the effects of regularization, data augmentation, pretrained backbone choice, and task reformulation.

For the baseline setup, small weight decay and label smoothing had the strongest positive impact on validation performance, while switching from BERT to RoBERTa-base further improved results without requiring changes to the training pipeline. Modifying the data augmentation strategies used for the baseline resulted in additional small improvements, but the gains were limited compared to changing the model.

The largest improvement resulted from both the NLI-style reformulation and the change of pretrained model. The DeBERTa-based NLI model achieved the best results across both metrics and converged quickly, even without extensive augmentation.

We would have liked to further explore data augmentation and identify which types of augmentation could improve the performance of the DeBERTa NLI model.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514, Hong Kong, China, November 2019. Association for Computational Linguistics.