SRCB at SemEval-2026 Task 5: A Multi-Target Finetuning Framework for Large Language Models with Joint Regression and Text Generation

Yuming Zhang, Junyu Zhou, Hongyu Li, Yongwei Zhang, Shanshan Jiang, Bin Dong
Ricoh Software Research Center (Beijing) Co., Ltd
{Yuming.Zhang1, Junyu.Zhou, Hongyu.Li}@cn.ricoh.com
{Yongwei.Zhang, Shanshan.Jiang, Bin.Dong}@cn.ricoh.com

Abstract

This paper presents our winning system for SemEval-2026 Task 5 on rating the plausibility of word senses in ambiguous stories. Unlike traditional Word Sense Disambiguation, the task requires predicting continuous plausibility scores that reflect human variability rather than selecting a single correct sense. We propose a multi-target fine-tuning framework for decoder-only large language models that jointly optimizes regression module for score prediction and text generation module for that produces both interpretable explanations and the predicted score. To address inter-annotator variability, we adopt objective-level strategies to enhance robustness. Our system achieves first place, demonstrating the effectiveness of unified regressive–generative modeling for fine-grained plausibility estimation.

1 Introduction

SemEval-2026 Task 5-Rating Plausibility of Word Senses in Ambiguous Stories through Narrative Understanding aims to address a key limitation of traditional Word Sense Disambiguation (WSD), which typically assumes a single "correct" sense despite real-world interpretation often being shaped by ambiguity and individual judgment. To capture this complexity, the task introduces AmbiStory, an English dataset of five-sentence stories where a target homonym appears in one sentence. Given a candidate sense, rather than predicting a binary label, systems are required to estimate its plausibility as perceived by humans, aggregated from multiple annotators on a 1–5 scale. Performance is evaluated via Spearman correlation with average human judgments and accuracy within one standard deviation of annotator ratings (Gehring et al., 2026).

Recent advances in WSD have been driven largely by encoder-based models, which leverage contextual representations and lexical knowledge to infer word meanings from context like GlossBERT (Huang et al., 2019), SenseBERT (Levine et al., 2020) and PolyBERT (Xia et al., 2025). SenseBERT is designed for "masked token prediction (classification)". PolyBERT and GlossBERT share the same core idea — using glosses (sense definitions) to assist disambiguation. However, all of them are not inherently designed to produce continuous plausibility scores—a key requirement addressed in this work. PolyBERT explicitly incorporates the target word embedding, which in theory enables more precise capture of polysemy and token-level differences. However, the performance of PolyBERT is still not comparable with LLM-based methods in our experiments.

In this paper, we present a multi-target fine-tuning framework for decoder-only LLMs (Figure 1) that jointly optimizes: (1) Regression Module, predicting a continuous plausibility score, and (2) Text Generation Module, producing both the plausibility score and its interpretable explanations. Both objectives are integrated within a unified framework, allowing the regression and generation components to jointly contribute to the loss and produce a calibrated numerical prediction in a single forward pass.

Task 5 presents a significant challenge due to substantial inter-annotator variability: identical target words and precontexts may receive divergent plausibility scores depending on the associated endings. To mitigate this issue, we adopt a series of strategies designed to enhance robustness:

Data level We employ Qwen3-Next-80B-A3B (Team, 2025a) to analyze why identical homonyms and precontexts receive different plausibility scores under different story endings. These analyses are then incorporated as auxiliary training tasks to improve the model's ability to distinguish and evaluate context-dependent plausibility.

Training level We treat high-variance samples as potentially noisy and assign them lower loss

Proceedings of the The 20th International Workshop on Semantic Evaluation (2026), pages 1957–1964

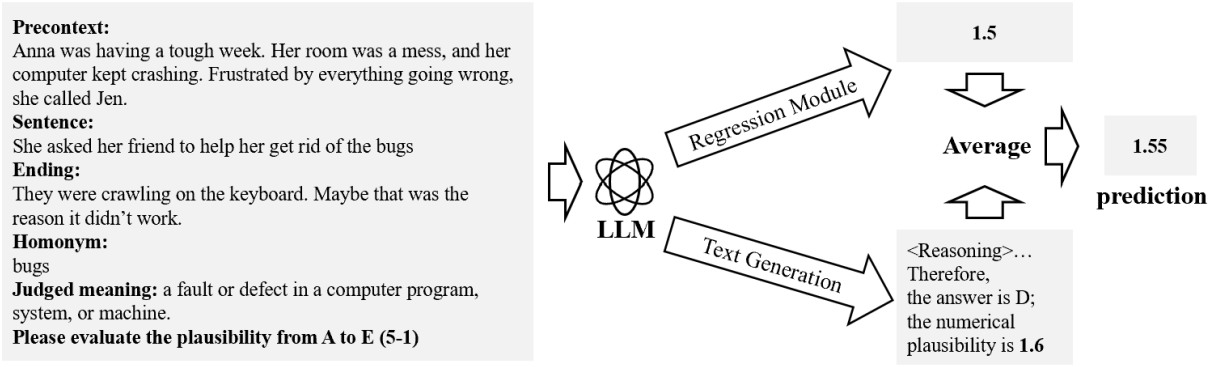

Figure 1: Overview of our system framework. The left panel presents a simplified example instance, and the full prompt template is described in Appendix A.2.

weights during training or apply outlier processing methods to reduce their impact on optimization.

Model level Beyond a single regression head for score prediction, we introduce multiple regression heads to simulate diverse annotator behaviors and better capture inter-annotator variability.

In addition, we adopt several strategies to further improve performance with respect to the task evaluation metrics. Through extensive experiments, we make the following key observations that mainly contribute to the performance improvements:

1. Incorporating generated auxiliary supervision data into the training set significantly improves overall system performance.
2. Optimizing a batch-wise spearman correlation–based loss leads to consistent improvements in the final Spearman evaluation score.
3. Our proposed framework outperforms the variants that rely solely on either the Regression Module or the Text Generation Module.

2 Background

AmbiStory dataset samples consist of three main components: homonym-related information, complete story, and annotator evaluation. Table 1 presents a concrete example.

Each complete story consists of: (1) a three-sentence precontext that establishes the narrative background; (2) an ambiguous sentence containing a homonym that allows two distinct yet plausible interpretations; and (3) optionally, one of two alternative endings, each biasing interpretation toward a specific sense.

Annotators rate the plausibility of a given judged meaning in context. A "Nonsensical" field indicates whether an ending is considered meaningless, illogical, or invalid. In the illustrated example, all five annotators agree the ending is not nonsensical, yet plausibility scores vary substantially (1, 1, 2, 1, 5)—revealing considerable disagreement despite consensus on semantic coherence.

Homonym: bugs **Judged meaning:** a fault or defect in a computer program, system, or machine. **Example sentence:** There's a bug in the software.
Precontext: Anna was having a tough week. Her room was a mess, and her computer kept crashing. Frustrated by everything going wrong, she called Jen. **Sentence:** She asked her friend to help her get rid of the bugs. **Ending:** They were crawling on the keyboard. Maybe that was the reason it didn't work.
Choices: 1, 1, 2, 1, 5 **Average:** 2.0 **Stdev:** 1.73 **Nonsensical:** false, false, false, false, false

Table 1: An example from the AmbiStory dataset.

Provided AmbiStory dataset is organized into standard training, development, and test splits. All data used in this task are in English, comprising 2280, 588, 930 instances respectively.

3 Data

3.1 Data Preprocessing

We construct two types of labels from the annotation data. (1) Direct Prediction, the label is the mean of all choices for a given instance. (2) Annotator Prediction, we assume there are only 5 annotators, each maintaining a consistent scoring distribution, and the n-th choice corresponds to the n-th annotator. Since most data contains 5 choices, we convert this task from predicting a single averaged score into predicting all 5 choices separately. We

first exclude instances with six scores, then use the n-th choice as the label for the n-th annotator.

To mitigate the impact of high stdev annotations, we apply two preprocessing strategies: 1) For Direct Prediction: Exclude any choice outside the average±stdev range; recompute the mean of the remaining choices as the reconstructed label. 2) For Annotator Prediction: Reconstruct labels to achieve centralization using:

$$choice_{reconstructed} = (choice + average)/2 \quad (1)$$

Where choice is one of the 5 original annotator choices, and average is the average value of them.

These adjustments have minimal effect on low stdev instances and remain within a reasonable range for high stdev ones. Since the evaluation metrics are elastic, any introduced discrepancy has limited impact on final evaluation. Therefore, introducing these adjustments can stabilize training without harming metrics.

3.2 Data Augmentation

To improve the model's ability in assigning context-based plausibility scores and in distinguishing subtle semantic differences among candidate endings, we introduce comparison-based auxiliary supervision and rationale-based augmentation.

Comparison-based auxiliary supervision We merge instances with the same story, homonym, and judged meaning but different endings into a single prompt (as shown in Appendix A.1). Qwen3-Next-80B-A3B then generates a comparative explanation analyzing why different endings yield varying plausibility scores. The resulting question–answer pairs serve as an auxiliary supervision signal, improving fine-grained semantic discrimination and score calibration.

Rationale-based augmentation For each instance in the training and development sets, we use Qwen3-Next-80B-A3B to generate a textual rationale explaining the assigned plausibility score. During training, this rationale is incorporated as the intermediate reasoning component within the target output, encouraging the model to learn the semantic basis underlying the scoring decision rather than merely fitting numerical labels (Shi et al., 2025).

4 System Description

As shown in Figure 1, our system comprises two modules sharing the same LLM backbone: a Regression Module that uses a feedforward neural network to predict a continuous plausibility score, and a Text Generation Module that produces the textual score. The final prediction is obtained by averaging the outputs of two modules. The Regression Module directly optimizes the distance between predicted and ground-truth plausibility scores, while the Text Generation Module leverages linguistic knowledge in LLM's vocabulary classifier. Their combination yields complementary benefits and enhances overall prediction performance.

4.1 Regression Module

For the input representation, we use the hidden state of the last token in the question, rather than the last non-padding token in the full sequence (which includes both the question and the answer). This design ensures that the Regression Module and the Text Generation Module can be trained jointly without causing data leakage.

4.2 Text Generation

As introduced in Section 3.2, we construct comparison-based auxiliary supervision data and augment the training and development sets with additional rationales. During fine-tuning, we define two training tasks which share the same token-level autoregressive objective.

Auxiliary explanation generation The input is the same as the merged data described in 3.2, and the target output is the generated explanation. This task trains the model to produce comparative semantic analyses that justify differences in plausibility scores across alternative story endings.

Plausibility classification with reasoning Each instance (Table 1) is reformulated as a prompt asking model how well a homonym matches the specified sense by first outputting a coarse-grained option from *Absolutely (5)*, *Probably (4)*, *Possibly (3)*, *Unlikely (2)*, and *Impossible (1)*, and then a fine-grained plausibility score. The prompt template is provided in Appendix A.2. For model responses, the rationale from data augmentation is inserted as "<Reasoning>", followed by the option and the plausibility score as shown in Figure 1. This encourages the model to first generate semantic justification and then produce a calibrated discrete decision aligned with the continuous human score.

4.3 Module Combination

Table 2 presents the performance of assigning different weights to the predictions of Regression Module and Text Generation Module. Specifically, the variant is "Direct Prediction + BCELoss + Text

Combinations	Acc w/in SD	Spearman	Average
Regression Module (without training Text Generation)	90.58%	82.88%	86.73%
Text Generation (without training Regression Module)	83.96%	79.16%	81.56%
1.0 * Regression Module + 0.0 * Text Generation	90.61%	**82.94%**	86.78%
0.0 * Regression Module + 1.0 * Text Generation	83.99%	79.14%	81.57%
0.5 * Regression Module + 0.5 * Text Generation	**90.92%**	82.83%	**86.88%**

Table 2: The performance of combing the plausibility score predicted by Regression Module and Text Generation in different weights.

generation". The average combination shows the best performance. It is worth noting that even if a training signal (Regression Module or Text Generation) is not used in the final scoring, it can still serve as an auxiliary objective that improves model performance. We do not further optimize these two weights (e.g. comparing 0.45 vs. 0.55), as excessive tuning would risk overfitting and hurt generalization to the test set.

4.4 Multiple Training Objectives

We propose four types of training objectives, most of which apply only to the Regression Module.

Basic Prediction We employ MSELoss to directly reflect the discrepancy between predicted and ground-truth scores. In addition, we regard the task as a probabilistic prediction problem using BCELoss, where the 1–5 scores are normalized to the 0-1 range, to capture the likelihood difference.

Spearman Improvement We incorporate a differentiable Spearman loss (Blondel et al., 2020a) to directly optimize the ranking-based evaluation metric. The detailed computation is provided in Appendix B. This loss is computed batch-wise and would align perfectly with the evaluation metric only if one batch contains the full dataset—impractical due to memory limits. To mitigate this, we randomly shuffle the entire dataset at the start of each training epoch, ensuring diverse sample correlations across batches without affecting with other objectives.

0.2-Step Prediction We observe that most average plausibility scores are derived from five annotators, therefore must be multiples of 0.2. To better align model predictions with this inherent label structure, we introduce a loss function that encourages predictions to approximate values divisible by 0.2 (or 1.0 in Annotator Prediction). The RoundLoss is formulated as:

$$loss = (label - round(label/s) * s)^2 \quad (2)$$

Where s = 0.2 for Direct Prediction and s = 1.0 for Annotator Prediction. Theoretically, the gap between the predictions will narrow, ideally resulting in only a few possible values (e.g., 1.0, 1.2, ..., 5.0 for Direct Prediction), which would improve the performance in the respect of ranking-based evaluation metrics.

Outlier Optimization We employ two strategies to reduce the influence of outlier instances: (1) Huber loss (Huber, 1992), which is more robust than mean squared error for outlier instances; and (2) Weight decay by stdev (w_std), where we down-weight stdev samples by scaling the loss as:

$$loss = loss/(stdev + 1) \quad (3)$$

4.5 Ensemble

We systematically finetune and evaluate promising model variants and their combinations, retaining all effective configurations. These effective variants are then used as candidates in a majority voting scheme to produce the final prediction.

5 Experimental Setup

We conduct experiments on the combined AmbiStory training and development sets with 5-fold cross validation (The detailed comparison with using only the training set is described in Appendix C) and augmented data (Section 3.2). To mitigate the impact of randomness, each model variant is trained with three random seeds and the reported results are averaged.

We adopt Qwen3-14B and Qwen3-30B-A3B-Instruct-2507 (Team, 2025b) (Qwen3-30B) as backbones. For Qwen3-14B, we use AdamW with a learning rate of 6×10^{-6}, no scheduler or warmup. For Qwen3-30B, we additionally apply LoRA (Hu et al., 2022) with a learning rate of 9×10^{-5}. The global batch size is 256 (32 batch size per GPU × 4 GPUs × 2 gradient accumulation steps). We apply early stopping with a patience of 5 epochs. All experiments are conducted on NVIDIA A100 80G GPUs.

BERT-based Baselines	Acc w/in SD	Spearman	Average
Regression	77.32%	68.39%	72.86%
Classification	74.72%	64.17%	69.45%
PolyBERT	79.10%	71.83%	75.47%
LLM-based Baselines			
0-shot Prompt Engineering + DeepSeek	78.60%	68.50%	73.55%
Text Generation	83.96%	79.16%	81.56%
Direct Prediction + BCELoss	90.58%	82.88%	86.73%
Annotator Prediction + BCELoss	89.29%	82.08%	85.69%
Direct Prediction + MSELoss	90.61%	82.58%	86.60%
Annotator Prediction + MSELoss	90.57%	82.68%	86.63%
LLM-based Variants			
Direct Prediction + BCELoss + SpearmanLoss	90.93%	83.12%	87.03%
Annotator Prediction + BCELoss + SpearmanLoss	90.96%	83.21%	87.09%
Direct Prediction + BCELoss + RoundLoss	90.61%	83.13%	86.87%
Annotator Prediction + BCELoss + RoundLoss	90.08%	83.31%	86.70%
Direct Prediction + HuberLoss	90.53%	82.93%	86.73%
Annotator Prediction + HuberLoss	90.86%	81.88%	86.37%
Direct Prediction + BCELoss + w_std	91.00%	**83.33%**	87.17%
Annotator Prediction + BCELoss + w_std	89.74%	83.03%	86.39%
Direct Prediction + BCELoss + Text generation	90.92%	82.83%	86.88%
+ Data augmentation	**91.50%**	83.09%	**87.30%**
Annotator Prediction + BCELoss + Text generation	90.58%	82.97%	86.78%
+ Data augmentation	91.10%	83.18%	87.14%
Direct Prediction + BCELoss + Qwen3-30B	90.69%	82.47%	86.58%
Annotator Prediction + BCELoss + Qwen3-30B	90.43%	82.45%	86.44%

Table 3: The performance of different variants. All LLM-based methods are based on Qwen3-14B except for Qwen3-30B annotated in the table. All BERT-based methods are based on Deberta-v3-large (He et al., 2021) , prompt engineering is based on DeepSeek-v3 (non-thinking) (Liu et al., 2024)

6 Results

Our system achieves the 1st place in the competition, also ranking first in each single evaluation metrics: 93.33% accuracy w/in SD and 85.57% spearman score.

6.1 The Performance Of Data Preprocessing

Although data preprocessing may reduce the ability of the model to fit difficult samples (i.e. high variance samples)—potentially suppressing useful signal from outliers—its overall impact, as evaluated through the current elastic metrics formulas and the empirical results reported in Table 4, remains beneficial to a certain extent.

6.2 The Performance Of Variants

Table 3 presents the performance of BERT-based baselines, LLM-based baselines and various LLM-based variants. Overall, almost all LLM-based variants achieve improvements to varying degrees compared to the baselines. Among these variants, the best-performing one is the combination that integrates the Regression Module, the Text Generation Module, and data augmentation.

We further conduct experiments to combine these effective variants together in order to explore their potential complementarity. The results reveal that while certain combinations lead to further performance gains, others fail to yield additional improvements, and in some cases even result in marginal degradation. We retain the effective variants and combinations in the experiments and regard them as candidates for model ensembling.

6.3 The Best Ensemble

Table 5 presents the selection variants of our best ensemble on the leaderboard, along with their performance on the 5-fold cross validation set. It also shows the comparison of our best ensemble on the 5-fold cross validation set and test set. The weight

LLM-based Variants	Acc w/in SD	Spearman	Average
Direct Prediction + BCELoss	90.58%	82.88%	86.73%
- Data Preprocessing	89.47%	82.80%	86.14%
Annotator Prediction + BCELoss	89.29%	82.08%	85.69%
- Data Preprocessing	89.13%	82.11%	85.62%

Table 4: The performance of data preprocessing

	5-fold cross validation		
Best Ensemble Selection Variants	Acc w/in SD	Spearman	Average
Direct Prediction + BCELoss + RoundLoss + Qwen3-30B	91.69%	82.97%	87.33%
Direct Prediction + BCELoss + Qwen3-30B	90.69%	82.47%	86.58%
Direct Prediction + BCELoss + w_std + Qwen3-30B	91.04%	82.85%	86.95%
Direct Prediction + BCELoss + w_std	91.00%	83.33%	87.17%
Direct Prediction + MSELoss + SpearmanLoss + Text generation + Data augmentation	90.61%	83.17%	86.89%
Direct Prediction + HuberLoss	90.53%	82.93%	86.73%
Annotator Prediction + BCELoss + SpearmanLoss	89.82%	82.23%	86.03%
Annotator Prediction + BCELoss + Qwen3-30B	90.43%	82.45%	86.44%
	5-fold cross validation		
	Acc w/in SD	Spearman	Average
Best Ensemble	93.86%	86.44%	90.15%
- Annotator Prediction + BCELoss + SpearmanLoss	93.97%	86.25%	90.11%
		Test	
	Acc w/in SD	Spearman	Average
Best Ensemble	93.33%	85.57%	89.45%

Table 5: The performance of best ensemble

assigned to each variant remains the same across all variants. The results lead to three key observations: (1) The ensemble method proves to be effective, improving performance by approximately 3% compared to individual variants. Although each variant appears to contribute to the ensemble from different perspectives, the specific interactions among them are complex and difficult to analyze in detail. This suggests that the strength of the ensemble lies in the diversity rather than the individual superiority of its components. (2) Even variants with relatively poor individual performance—such as "Annotator Prediction + BCELoss + SpearmanLoss" in the table—can still contribute positively to the final ensemble more or less. This is evidenced by the drop in performance when that variant is removed from the best ensemble, indicating that seemingly weak models may provide useful regularization or capture edge cases that other stronger models miss. Thus, excluding a variant solely based on its performance could be suboptimal. (3) Performance on the 5-fold cross validation set shows a strong correlation with that on the test set, indicating that our methods exhibit a certain degree of robustness with limited overfitting.

7 Conclusion

In this paper, we describe our multi-target finetuning framework for the rating plausibility of word senses in ambiguous stories through narrative understanding task, which wins the 1st place. Our system combines the Regression Module and Text Generation Module together and apply some variants to further improve the performance. Most of the variants show improvement, moreover, the combination of the Regression Module and the Text Generation Module shows further improvement.

Limitations

As our system needs to combine the Regression Module and Text Generation module together, the efficiency of text generation has become a limitation. On one hand, applying batch inference or inference acceleration framework in our experiments shows comparable performance difference. On the other hand, inference one by one takes a lot of time.

References

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. 2020a. Fast differentiable sorting and ranking. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. 2020b. Fast differentiable sorting and ranking. In *International conference on machine learning*, pages 950–959. PMLR.

Janosch Gehring, Selina Meyer, and Michael Roth. 2026. SemEval-2026 task 5: Rating plausibility of word senses in ambiguous stories through narrative understanding. In *Proceedings of the 20th International Workshop on Semantic Evaluation*, San Diego, California. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *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. Association for Computational Linguistics.

Peter J Huber. 1992. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pages 492–518. Springer.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online. Association for Computational Linguistics.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Wenhang Shi, Shuqing Bian, Yiren Chen, Xinyi Zhang, Zhe Zhao, Pengfei Hu, Wei Lu, and Xiaoyong Du. 2025. Investigating the impact of rationales for llms on natural language understanding. *Preprint*, arXiv:2510.16686.

Qwen Team. 2025a. Qwen3-next: Towards ultimate training & inference efficiency.

Qwen Team. 2025b. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Linhan Xia, Mingzhan Yang, Guohui Yuan, Shengnan Tao, Yujing Qiu, Guo Yu, and Kai Lei. 2025. Polybert: Fine-tuned poly encoder bert-based model for word sense disambiguation. *Preprint*, arXiv:2506.00968.

A Prompt template

A.1 Prompt A

Template for auxiliary supervision data generation

You are an expert linguist and evaluator.

Story:
{*precontext*}
{*sentence*}

Target meaning of the word "{*homonym*}":
{*judged meaning*}

Different story endings and their corresponding target meaning scores from 1 to 5:
{*ending 1, score 1*}
{*ending 2, score 2*}
{*ending 3, score 3*}

Instructions:
1. Provide a single coherent paragraph explaining why these endings receive different plausibility scores.
2. Your explanation should analyze how each ending aligns or fails to align with the specified target meaning in context.
3. Do not repeat the scores. Explain the semantic reasons.

A.2 Prompt B

Template for plausibility classification with reasoning

<Context>
{*precontext*}
{*sentence*}
{*ending*}
</Context>

Question:
Does "{*homonym*}" in "{*sentence*}" mean "{*judged meaning*}" like the meaning in "{*example sentence*}"?

Options:
A. Absolutely
B. Probably
C. Possibly
D. Unlikely
E. Impossibly

Just use one option to correctly answer the question and evaluate its degree from 1 to 5.

B Spearman loss

Step 1: Collect the predictions and labels in a batch
Step 2: Soft rank (differentiable) the predictions and labels for calculation (Blondel et al., 2020b) (we use torch.soft_rank (Paszke et al., 2019) in our experiments)

**Same as the calculation of Spearman*
Step 3: Perform centralization processing on the prediction and labels
Step 4: Calculate covariance
Step 5: Calculate the standard deviation
Step 6: Calculate the score of Spearman
Step 7: Convert the score into loss: loss = 1 - score

Table 6: The calculation process of spearman loss.

C Incorporating the Development Set

Since our final model variants are trained on the combined training and development sets, we first conduct a preliminary experiment to assess the impact of adding the development data. Table 7 compares our baseline (Qwen3-14B with Direct Prediction using BCELoss) under two setups: training on the training set alone versus on the combined dataset. Reported scores are the average of accuracy within one standard deviation and Spearman correlation.

Note that the two setups are evaluated differently: models trained on the training set are evaluated on the official development set, while those trained on the combined dataset are evaluated via 5-fold cross validation. Thus, the "Avg on own dev" columns are not directly comparable across setups, whereas the "Avg on test" columns are, as both reflect performance on the same test set.

We observe a positive correlation between "Avg on test" and "Avg on own dev." Given that the training and development sets share the same underlying distribution, this supports using development set performance as a proxy during exploration. In addition, adding development set into training shows a certain improvement.

Based on this, we adopt a two-stage pipeline: (1) Exploration: fine-tune on the training set and evaluate on the development set to validate variants; (2) Application: fine-tune promising variants on the combined dataset and evaluate via 5-fold cross validation for final submission. All experiment results are from the Application stage.

Datasets	Avg on own dev	Avg on test
Train	85.45%	83.93%
Train + Dev	86.73%	84.51%

Table 7: Impact of Incorporating Development Set