Pushing the Limits of Sentiment Analysis on SST5 with Smaller Models

Vyacheslav Shishaev

May 2025

Abstract

Sentiment analysis in natural language processing and deep learning is an important field, but it is still underdeveloped in many ways. The main problems, such as the similarity of close emotions, ambiguity of context, lack of labeled data for rare sentiment categories, and subjectivity in human annotations are underexplored. In this project, the main focus will be on applying modern techniques to get a good accuracy score on the SST5 [3] dataset. Link to Github: `https://github.com/InfinitasFish/sst5_ods_proj`.

1 Introduction

Sentiment analysis is a crucial task in natural language processing (NLP) with applications ranging from customer feedback analysis to social media monitoring. While significant progress has been made in binary and ternary sentiment classification, fine-grained sentiment analysis—such as the Stanford Sentiment Treebank (SST-5) dataset with five distinct classes — remains challenging. The key difficulties include distinguishing between closely related sentiment intensities (e.g., somewhat positive vs. positive) and handling contextual ambiguity, such as sarcasm or implicit sentiment cues.

Most SOTA approaches rely on standard fine-tuning of big pre-trained language models like BERT or RoBERTa. However, these models often struggle with subtle sentiment differences due to their general-purpose training objectives. To address this, I propose a hybrid training strategy combining:

1) Triplet loss – Enhances embedding separation between sentiment classes by explicitly optimizing relative distances between anchor, positive, and negative samples.

2) Gradual unfreezing – Mitigates forgetting pre-training knowledge during fine-tuning by incrementally unfreezing layers (either layer-wise or after a fixed epoch).

To standardize input formatting and improve model interpretability, I use a task-specific prompt to each text sample during preprocessing. That approach reinforces the classification schema, and showed its effectiveness in predictions' accuracy.

1.1 Team

The only team member is myself: **Vyacheslav Shishaev** - prepared this document and wrote the code.

2 Related Work

Recent advances in fine-grained sentiment analysis on SST-5 have focused on three key directions: prompt optimization, Heinsen Sequence Routing, and self-explanatory modeling.

CAPO [5] – address the high computational cost of prompt engineering in LLMs by proposing an evolutionary algorithm that integrates AutoML techniques with multi-objective optimization, while balancing accuracy versus prompt length. It jointly optimizes instructions and few-shot examples while leveraging task descriptions for improved robustness. Though primarily designed for LLMs, its cost-efficiency principles are adaptable to smaller models.

Heisen Routing Algorithm [2] – introduces a novel vector routing mechanism that reformulates sequence processing as credit assignment problem, tries to maximize "bang per bit" by dynamically weighting input vectors and enables interpretable routing decisions through geometric latent variables, has most effect on visual classification tasks and long sequences.

Self-Explaining Structures [4] – propose an framework that adds an interpretation layer on top of any nlp model to compute span-level weights, eliminates need for probing models by providing direct phrase/sentence importance scores while using weighted span combinations for final prediction.

These methods are exeeding in prediction accuracy on test split of SST5 [3], but they use large models, from 355M roberta up to 70B Llama-3.3.

3 Model Description

The main idea behind the model – applying SOTA methods but in simplified way to avoid large computational overhead and be able to fine-tune with limited resources. To not train the model from the scratch, pre-trained on Twitter posts roberta-base was used [1]. Model has backbone that specializes on sentiment analysis of Twitter posts, that is pretty close to SST5 [3] movie reviews domain and custom head for outputting 5 different labels. Backbone and head combined have only 125M parameters. To achieve good accuracy, different techniques were applied:

1) Simple initialization prompt is added to each text, defined as follow: "Movie review sentiment classification task: From the following five options - very negative, negative, neutral, positive, or very positive (0-4) – which best describes this review? text";

2) Gradual unfreezing module, which helps to optimize fine-tuning for new dataset while not losing any pre-training knowledge. From the start only classification head is trained, and after some amount of epochs backbone is being unfreezed completely or gradually layer by layer. It also prevents fast overfitting on small datasets.

3) To make model distinguish close classes (label pairs 0-1, 1-2, etc.) better, triplet loss was applied on backbone layers for making close sentiments more contrastive in terms of their embeddings. The triplet loss function is defined as:

$$\mathcal{L}_{triplet} = \max\left(0, \|f(a) - f(p)\|_2^2 - \|f(a) - f(n)\|_2^2 + \alpha\right) \tag{1}$$

where:

- a, p, n are anchor, positive, and negative samples
- $f(\cdot)$ is the embedding function
- α is the margin hyperparameter (default $\alpha = 1.0$)

As a result, I have a relatively small model that achieves competitive results compared to large SOTA solutions. Described model is shown in figure 1.

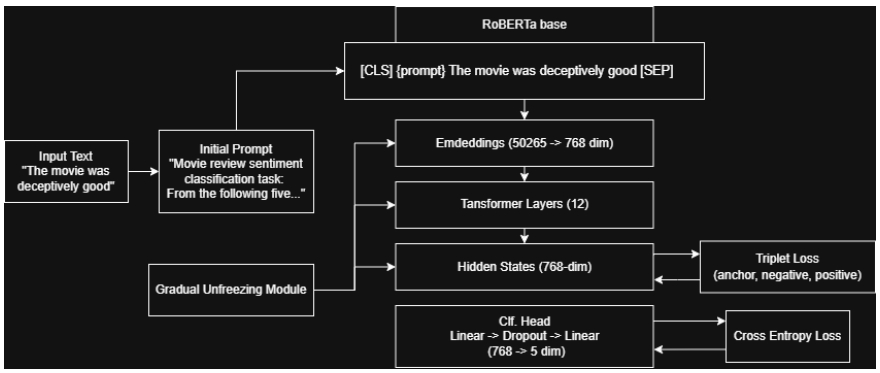

Figure 1: Full model architecture.

4 Dataset

SST5 [3] is pretty old benchmark, but it fits well for researching because top-1 accuracy on test split is pretty low to this day – 62.27 accuracy – a lot of

room for improvement. Also this dataset is pretty small, that's convenient in terms of computational cost. SST5 can be downloaded from `https://nlp.stanford.edu/sentiment/` or from kagglehub with command `kagglehub.dataset_download('haoshaoyang/sst5-data')`

On the Tab. 1 you can see the minimalistic statistics for SST5.

	Train	Valid	Test
Phrases	8544	1101	2210
Class Distribution (0-4)			
12% 19% 25% 23% 21%			

Table 1: Statistics of the SST5.

5 Experiments

Before combining all techniques in one model, each of them was tested one by one, and then added to the final model if shown increase in accuracy on test split.

5.1 Metrics

For evaluating and model-selecting two main metrics were selected: F1 with macro weightening and accuracy. Macro F1 enables us to select model that can perform great on all classes, including rare ones, because macro weightening gives equal weight to all classes. Accuracy is also used for comparing to other SOTA approaches, because current leaderboard `https://paperswithcode.com/sota/sentiment-analysis-on-sst-5-fine-grained` uses this metrics.

5.2 Experiment Setup

Each experiment contains a sequential addition of techniques to each other. For each technique, 1-2 trainings were conducted, within 40 epochs for each experiment. Different learning rates were set for the backbone and the classification head (α and $\alpha/10$ roughly) to prevent overfitting. Dataset splits were constant in all experiments. GPU P100 on kaggle was used for training.

5.3 Baselines

For the baseline original pre-trained model was used without unfreezing backbone, only custom classification head was trained with learning rate $2e-5$. In result model has 0.463 f1-macro and 0.485 accuracy on test split, which is pretty low, that's why it is used as a baseline.

6 Results

The first improvement was experimenting with optimal backbone utilization. To utilize it properly, an unfreezing module was proposed with two different techniques: "after-n" and "layerwise". First one unfreezes full backbone after fixed amount of epochs, second one unfreezes backbone layers one by one also after fixed amount of epochs.

For the first strategy unfreeze epoch was 7, classification head with learning rate $8e^{-}5$ and backbone with learning rate $8e^{-}6$, model was trained for 30 epochs, though the best metrics were shown shortly after unfreezing. It has shown great improvement compared to baseline: f1-macro 0.463 -> 0.561 and accuracy 0.485 -> 0.5669 on test split.

"Layerwise" strategy has been shown a bit unstable and hard to train properly. To compensate slow unfreezing, the backbone learning rate was set equal to $8e^{-}5$, equal to head's, and model was trained for 40 epochs. Result metrics are not that impressive: f1-macro 0.528 and accuracy 0.5425. So for the next improvements "after-n" strategy was used.

The next step is adding described above initialization prompt to each text, that should theoretically help with reducing variance in transformer representations by explicitly defining the label space (0–4) and classification goal, though adding some computational overhead and slowing training. Unfreeze epoch with "after-n" strategy was set to 5, head learning rate is $5e^{-}5$ and backbone learning rate is $5e^{-}6$, model was trained for 25 epochs, hitting the best metrics at epoch 11. Compared to previous model, prompt approach achieves a bit better accuracy, but not f1: f1-macro 0.561 -> 0.5556 and accuracy 0.5669 -> 0.5747 on test set. Leaderboard is using accuracy, so prompt was also used for training next iteration.

The last improvement – using triplet loss for better backbone training, by generating triplets with close to each other classes: 0 with 1, 1 with 2, 2 with 3, 3 with 4. But in a way, that doesn't inflate amount of datapoints in training set – for each phrase I randomly choose one index for positive example and one index for negative example. Unfreeze epoch is 7 with "after-n" strategy, prompt is used for each text, triplet loss weight is 0.2, head learning rate is $5e^{-}5$ and backbone learning rate is $5e^{-}6$, model was trained for 23 epoch, and training took much more time, roughly 4 times longer than previous iterations, which has taken around 1-1.5 hour each. But results aren't very impressive: f1-macro 0.5505 and accuracy 0.5725 on test set.

All described results are shown in table 2.

7 Conclusion

With a model that is smaller from three to roughly five hundred times compared to SOTA models on the leaderboard `https://paperswithcode.com/sota/sentiment-analysis-on-sst-5-fine-grained`, I was able to achieve competitive accuracy on test split of SST5 and take top-5 spot.

Model	F1-macro	Accuracy
baseline	0.463	0.485
unfrez. "after-n"	0.561	0.5669
unfrez. "layerwise"	0.528	0.5425
"after-n" + prompt	0.5556	0.5747
"after-n" + prompt + t.loss	0.5505	0.5725
Llama70B + CAPO	-	0.6227
Heinsen R. + RoBERTa Large	-	0.598
RoBERTa Large + SES	-	0.591
Heinsen R. + GPT-2	-	0.585

Table 2: Experiments results and other SOTA's accuracies, metrics are calculated on test split.

The most promising idea that I would try if I would had more time and started working earlier - paraphrase augmentation and/or adding more data from other sentiment domain for training, that is labeled in similar way as SST5. In my experience, when the dataset is pretty small, such augmentations can help to get a much higher metric if done carefully without noising the domain of the existing dataset.

And it seems that using prompts with triplet loss does more harm than good, but the idea of triplet loss is very promising, and I believe, if trained without prompt, would output accuracy better, than 0.5747. So more experiments can be done to achieve even higher place in leaderboard, and my approach still has a lot of room for improvement.

References

[1] Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. Tweeteval: Unified benchmark and comparative evaluation for tweet classification, 2020. URL `https://arxiv.org/abs/2010.12421`.

[2] Franz A. Heinsen. An algorithm for routing vectors in sequences, 2022. URL `https://arxiv.org/abs/2211.11754`.

[3] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `https://aclanthology.org/D13-1170/`. Dataset: Stanford Sentiment Treebank (SST-5), available at `https://nlp.stanford.edu/sentiment/`.

[4] Zijun Sun, Chun Fan, Qinghong Han, Xiaofei Sun, Yuxian Meng, Fei Wu, and Jiwei Li. Self-explaining structures improve nlp models, 2020. URL https://arxiv.org/abs/2012.01786.

[5] Tom Zehle, Moritz Schlager, Timo Heiß, and Matthias Feurer. Capo: Cost-aware prompt optimization, 2025. URL https://arxiv.org/abs/2504.16005.