# Sentiment Analysis

**Ben Ellis** [1] **Bruce Balfour** [1] **Florian Zogaj** [1] **Jakob Hütteneder** [1]
Department of Computer Science, ETH Zurich, Switzerland

## Abstract

We address the task of three-class sentiment classification on a dataset of 102,000 user-generated reviews, characterized by informal language and class imbalance. Our approach combines transformer-based encoders, minimal preprocessing, and lightweight ensembling. We investigate data augmentation, loss functions aligned with task objectives, and active learning. The final system—a fine-tuned DeBERTa-v3-large ensemble with a CNN classification head—achieves strong and consistent performance across validation and test sets.

## 1 Introduction

Sentiment classification is a foundational task in natural language processing (NLP), playing a crucial role in applications such as customer review analysis, brand monitoring, and automated moderation. While recent transformer-based models have advanced state-of-the-art performance on benchmark datasets, real-world sentiment tasks still pose unique challenges. User-generated text is often informal, noisy, and imbalanced—factors that can undermine the reliability of standard classification pipelines.

In this work, we explore sentiment classification over a moderately imbalanced dataset of approximately 102,000 single-sentence English reviews. Unlike synthetic or curated benchmarks, this dataset reflects realistic conditions: class skew, informal phrasing, and minimal text structure. Our goal is to develop a robust and efficient modeling pipeline using modern transformer architectures. Along the way, we evaluate preprocessing techniques, data augmentation, active learning, loss design, and model ensembling to determine their practical value in this constrained setting.

github.com/bennellis/CIL_Sentiment_analysis

[1]Department of Computer Science, ETH Zurich, Switzerland. Correspondence to: Ben Ellis <beellis@ethz.ch>, Bruce Balfour <balfourb@ethz.ch>, Florian Zogaj <zogajf@ethz.ch>, Jakob Hütteneder <jhuetteneder@ethz.ch>.

## 2 Related Work

Early sentiment classification methods used linear models with hand-crafted features such as n-grams and POS tags (6), later improved by neural networks like CNNs and RNNs (10). The introduction of pre-trained transformers—e.g., BERT (1), RoBERTa (5), and DeBERTa (2)—marked a major shift, enabling powerful context-aware representations.

Class imbalance remains a persistent challenge in NLP. Strategies such as resampling, weighted losses, and active learning (9) have been proposed, though their integration into transformer-based sentiment models remains limited.

Our work builds on these foundations, focusing on practical effectiveness over architectural novelty, and evaluating modern techniques like ensembling, custom loss functions, and active learning in a realistic classification setting.

## 3 Data

We conduct sentiment classification on a dataset of approximately $102,000$ labeled user reviews, each consisting of a single sentence. The sentiment labels are derived from associated star ratings: high-star reviews are labeled as positive $(+1)$, low-star reviews as negative $(-1)$, and mid-star reviews as neutral $(0)$. The resulting distribution is moderately imbalanced, with $30.5\%$ positive, $48.0\%$ neutral, and $21.5\%$ negative samples.

The reviews are drawn from multiple sources and predominantly $(> 99.5\%)$ in English. However, many include informal text artifacts—such as typos, elongated words (e.g., "soooo good"), emojis, and embedded URLs—which present unique challenges for NLP models and make preprocessing a potentially high-leverage step.

### 3.1 Preprocessing & Text Cleaning

To standardize and clean the text input, we experimented with several common preprocessing steps: lowercasing, contraction expansion (e.g., "can't" $\rightarrow$ "cannot"), removal of URLs and numbers, normalization of repeated characters (e.g., "soooo" $\rightarrow$ "soo"), punctuation and stopword removal, lemmatization, and translation to English. Each method was applied in isolation and evaluated using a frozen DistilBERT encoder with an MLP classification head.

Interestingly, most preprocessing techniques had little to no positive impact on validation performance. Some, such as removing punctuation or stopwords, even degraded performance—likely due to discarding syntactic or affective markers important for sentiment interpretation. Table summarizes the observed effects on validation accuracy.

| Pre-processing Technique | Validation |
|---|---|
| None | **0.855** |
| Expand Contractions | **0.855** |
| Remove URLs | **0.855** |
| Remove Numbers | **0.855** |
| Normalize Repeated Characters | 0.854 |
| Lemmatize | 0.854 |
| Remove Punctuation | 0.850 |
| Remove Stopwords | 0.828 |

*Table 1.* Model validation scores under different pre-processing techniques

In some cases, aggressive preprocessing (e.g., punctuation removal) produced empty or meaningless text samples. These were filtered out prior to training and assigned the neutral class during inference to avoid invalid predictions. Additionally, 18 duplicate entries were detected in the dataset. Only one pair had conflicting labels, so deduplication had no measurable impact on model performance.

### 3.2 Addressing Class Imbalance

Given the moderate skew in label distribution, we tested dynamic undersampling to counteract class imbalance. In this setup, a class-balanced subset of the training data was sampled anew at the start of each epoch. This allowed the model to eventually see the full dataset while maintaining per-epoch balance across classes.

Initial results suggested more stable training dynamics, but overall performance suffered. As shown in Table 2, models trained on the full, imbalanced dataset consistently outperformed those trained on dynamically rebalanced subsets—regardless of whether evaluation was done on balanced or unbalanced validation data. This suggests that the undersampling introduced distributional artifacts that hindered generalization.

| Train Distribution | Balanced Val. | Unbalanced Val. |
|---|---|---|
| Balanced Subset | 0.841 | 0.841 |
| Original (Imbalanced) | **0.855** | **0.854** |

*Table 2.* Validation score with different training and validation distributions (DistilBERT + MLP).

Based on these results, all subsequent models—including those used in the final ensemble—were trained on the full, imbalanced dataset without resampling.

### 3.3 Data Augmentation via Back Translation

To expand training data and reduce overfitting, we experimented with back-translation, where English sentences are translated into another language and then back to English to generate paraphrases. We evaluated multiple target languages and found that French produced the most natural and semantically faithful outputs.

Translation was performed using the MarianMT model (4) via the Hugging Face Transformers library. After deduplication, this process nearly doubled the dataset to $\sim 192,000$ samples.

Although back-translation introduced lexical variety, it also carried a risk of sentiment distortion. As shown in Table 3, some translations altered the underlying sentiment, which could introduce label noise and impair learning.

| | |
|---|---|
| **Original$_1$** | I wish I could give my barber zero stars... |
| **Result$_1$** | I wish I could give my hairdresser zero stars... |
| **Original$_2$** | The tickets were fairly priced |
| **Result$_2$** | The tickets were quite expensive |

*Table 3.* Examples of successful and failed backtranslations. While the first preserves the sentiment, the second significantly alters it.

Quantitatively, training a DistilBERT model on the augmented dataset led to a modest improvement of $+0.4\%$ in validation score. Despite the potential benefits, we ultimately chose not to use the augmented data in final models due to concerns over sentiment drift and label integrity.

## 4 Methods

In this section, we describe the modeling approaches and training strategies used in our sentiment classification pipeline. We began by establishing simple baselines, then transitioned to transformer-based models with varying levels of fine-tuning and architectural extensions. We also investigated the impact of different classification heads, loss functions aligned with the evaluation metric, and active learning as a strategy for improved label efficiency and robustness. Finally, we combined our strongest models into an ensemble to maximize final performance.

### 4.1 Baselines

We use two lightweight baselines, as well as a human labeling baseline, to establish reference performance levels and evaluate pre-processing and augmentation strategies.

**Bag of Words + Logistic Regression** The simplest baseline used a bag-of-words (BoW) representation with unigram and bigram features, followed by a logistic regression classifier. Despite its simplicity, this model enabled rapid prototyping and confirmed early observations—such as the limited benefit of preprocessing or back-translation. Performance plateaued around 80% validation accuracy.

**DistilBert + MLP** As a stronger baseline, we used DistilBERT (8) encoder and a simple MLP classification head attached to the [CLS] token output. This model established a fast, stable reference for comparing architectural variations and training strategies. With minimal tuning, it achieved a validation score of $0.855$.

**Human Labeling** To better understand the performance of our models in comparison to human judgment, we manually labeled a random sample of 500 sentences, achieving a score of $0.87$. This provides a stronger baseline and shows the complexity of the dataset, where even a human struggles to be completely accurate. This establishes the objective of creating a model that can exceed human-level performance.

## 4.2 Transformer Encoders

Building on the DistilBERT baseline, we evaluated multiple transformer backbones, including BERT (1), RoBERTa (5), ModernBERT (11), and DeBERTa-v3 (3). For each architecture, we evaluated three training regimes: *frozen encoders*, where only the classification head is trained; *partial fine-tuning*, in which a subset of encoder layers is updated; and *full fine-tuning*, where all parameters are trained end-to-end.

Freezing the encoder significantly reduced training time and hardware requirements, but also limited model performance, especially in more complex architectures. Full fine-tuning consistently led to the highest scores—often improving validation accuracy by 3–4 percentage points—though at the cost of increased training instability and overfitting risk. Partial fine-tuning consistently underperformed, with accuracy degrading as more layers were kept frozen.

To address early instability from randomly initialized heads in full fine-tuning, we adopted a two-phase training strategy: first training only the classification head while keeping the encoder frozen, then unfreezing the full model and continuing training. This yielded mild but consistent improvements across multiple architectures.

## 4.3 Classification Heads

To explore the impact of downstream architectures, we evaluated three classification heads on top of the transformer encoders. The *MLP head*, which applies a fully connected layer to the [CLS] token, served as the simplest and fastest option. The *LSTM head* applied a bidirectional recurrent layer over all token embeddings, allowing the model to aggregate sequence-level features. Finally, the *CNN head* used multiple one-dimensional convolutional filters of varying kernel sizes, following the architecture proposed by Kim (2014), to extract local n-gram patterns from token representations.

When used with frozen transformers, both the LSTM and CNN heads outperformed the MLP, indicating their ability to capture richer contextual signals from the full sequence. However, under full fine-tuning, the performance gap narrowed: the CNN continued to offer slight improvements, while the LSTM often degraded performance—possibly due to conflicting gradient dynamics between the pre-trained encoder and the recurrent layer. Overall, the CNN head offered the most consistent results across both frozen and fine-tuned setups.

## 4.4 Active Learning Loop

One method we used to address class imbalance and improve model efficiency was active learning. We implemented an entropy-based sampling loop, where a model using frozen BERT embeddings and a trainable classification head selected high-uncertainty samples for training. This strategy replaced the standard random sampling used in traditional training loops.

Compared to baseline training, active learning led to higher validation accuracy and, more importantly, improved class-wise consistency: models trained with active learning maintained balanced validation performance across all three classes, whereas models trained normally tended to favor the majority class.

Due to computational constraints, we did not apply active learning to fully fine-tuned transformer models. Nonetheless, we expect similar gains in robustness and generalization in that setting.

## 4.5 Loss Function

The evaluation metric for this task was a custom score function $L(\hat{y}, y) = 0.5 \cdot \left(2 - \frac{1}{n}\sum_i |y_i - \hat{y}_i|\right)$ equivalent to $1 - \text{MAE}$ scaled to $[0, 1]$. To optimize directly for this objective, we implemented a differentiable approximation of the mean absolute error over softmax probabilities:

$$L(y, \hat{p}) = \frac{1}{2}\sum_{c \in \mathcal{C}} \hat{p}_c \cdot |c - y|, \qquad \hat{p}_c = \frac{e^{z_c}}{\sum_{j \in \mathcal{C}} e^{z_j}} \tag{1}$$

However, this loss frequently caused models to collapse to the neutral class. Gradient analysis revealed a vanishing signal when predictions approached confidence extremes:

$$\frac{\partial L}{\partial z_j} = \frac{1}{2}\left(\hat{p}_j \cdot \Big|j - y\Big| - \sum_c \hat{p}_c \cdot |c - y|\right) \tag{2}$$

To prevent collapse, we experimented with hybrid objectives that mixed in a small cross-entropy (CE) term. While CE stabilized training, it counteracted the metric-alignment of MAE. To address this trade-off, we ultimately adopted the Class Distance Weighted Cross-Entropy (CDW-CE) loss (7), defined as:

$$\textbf{CDW-CE} = -\sum_{c \in C} \log(1 - \hat{p}_c) \cdot |c - y| \tag{3}$$

This loss penalizes off-by-2 errors more heavily than off-by-1, aligning well with the evaluation objective while maintaining useful gradients. CDW-CE was used for all final models unless otherwise stated.

### 4.6 Ensembling

To improve generalization and reduce variance, we combined our strongest individual models using both *soft voting* (averaging softmax outputs) and *hard voting* (majority class prediction). In tie cases under hard voting, the neutral class was selected.

Both ensemble types improved over single models, with soft voting performing slightly better $(+0.3–0.5\%)$. Our final submission used a soft ensemble of four DeBERTa-v3-large models fine-tuned from different initializations.

### 4.7 Experimental Setup

All experiments were conducted using PyTorch, Hugging Face Transformers, and Optuna for hyperparameter tuning. We used MLflow to track model configurations, training metrics, and validation performance. Early experimentation was performed on baseline models to identify effective learning rates, dropout values, and optimizer settings using Optuna. These hyperparameters were then transferred to larger transformer models to avoid excessive tuning overhead.

Training was performed on a single NVIDIA RTX 3080 GPU. Fine-tuning large models, such as DeBERTa-v3-large, required between 2–4 hours per epoch. However, performance typically peaked after just 3–4 epochs, with additional training leading to overfitting. We employed a two-phase training scheme—initially training the classification head with the encoder frozen, followed by full fine-tuning—along with early stopping based on validation score. Checkpoints were evaluated 10 times per epoch, with the best-scoring model saved for final testing.

Our setup used a fixed 90/10 train/validation split. We did not explore parameter-efficient fine-tuning methods (e.g., LoRA, prefix-tuning), as our simpler approach proved sufficient to achieve high performance on the task.

## 5 Results

After extensive experimentation across multiple model families and training strategies, our best-performing system was a soft-voting ensemble of four independently fine-tuned DeBERTa-v3-large models. This ensemble achieved a test score of $0.9084$, surpassing both the strongest individual model (single DeBERTa-v3-large, $0.905$) and human-level labeling accuracy $(0.87)$.

Ensembling was performed using soft-voting over the final softmax distributions. Each model in the ensemble was fine-tuned from a different initialization but used the same architecture and training configuration.

Table 4 summarizes validation and test scores across key models. Notably, the largest performance jumps occur when moving from frozen transformer baselines to full fine-tuning, and again when using DeBERTa-v3 over earlier architectures. Smaller gains were observed from changes to classification heads and training objectives, but these were important for stabilizing training.

| Models | Validation | Test |
|---|---|---|
| Human Labeling | 0.870 | |
| BoW + Logistic Regression | 0.804 | 0.740 |
| Frozen RoBerta-Base + CNN | 0.856 | 0.810 |
| Frozen RoBerta-Base + RNN | 0.853 | 0.816 |
| DistilBert + MLP | 0.855 | 0.805 |
| Bert-Base + MLP | 0.864 | 0.819 |
| RoBerta-Base + MLP | 0.882 | 0.851 |
| ModernBert-Base + MLP | 0.889 | 0.848 |
| DeBerta-V3-Base + MLP | 0.902 | 0.888 |
| DeBerta-V3-Base + CNN | 0.904 | 0.888 |
| DeBerta-V3-Large + MLP | 0.908 | 0.905 |
| hard-Ensemble | 0.9234 | 0.9079 |
| soft-Ensemble | **0.9288** | **0.9084** |

*Table 4.* Model scores evaluated using the custom score metric

## 6 Discussion & Conclusion

Our experiments highlight several factors that contributed to building a strong sentiment classifier on informal, imbalanced data. Contrary to expectations, standard preprocessing techniques such as stopword or punctuation removal often harmed performance, underscoring the value of preserving the original input structure when using transformers.

Efforts to mitigate class imbalance through dynamic undersampling and back-translation proved ineffective or unstable, with the latter introducing sentiment drift. As a result, we opted to train on the original distribution using only original data.

The most impactful modeling decisions included adopting DeBERTa-v3-large with full fine-tuning, using a CNN classification head, and employing a custom class-distance-weighted loss aligned with the task metric. These choices improved both training stability and class-wise performance. Finally, ensembling multiple fine-tuned models offered a straightforward and effective way to further boost performance and reduce variance.

While we did not explore more advanced fine-tuning methods or large-scale semi-supervised setups, our simple pipeline proved sufficient to reach strong performance on this task. Future work could build on these results by integrating more efficient training strategies or improved data sampling techniques.

## References

[1] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

[2] He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *ICLR*, 2021.

[3] He, P., Gao, J., and Chen, W. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2023. URL https://arxiv.org/abs/2111.09543.

[4] Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A. F., Bogoychev, N., et al. Marian: Fast neural machine translation in c++. *arXiv preprint arXiv:1804.00344*, 2018.

[5] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

[6] Pang, B., Lee, L., and Vaithyanathan, S. Thumbs up? sentiment classification using machine learning techniques. In *EMNLP*, 2002.

[7] Polat, G., Ergenc, I., Kani, H. T., Alahdab, Y. O., Atug, O., and Temizel, A. Class distance weighted cross-entropy loss for ulcerative colitis severity estimation, 2022. URL https://arxiv.org/abs/2202.05167.

[8] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

[9] Settles, B. Active learning literature survey. *University of Wisconsin-Madison Department of Computer Sciences*, 2009.

[10] Tang, D., Qin, B., and Liu, T. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, 2015.

[11] Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., Gallagher, A., Biswas, R., Ladhak, F., Aarsen, T., Cooper, N., Adams, G., Howard, J., and Poli, I. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference, 2024. URL https://arxiv.org/abs/2412.13663.

## A Appendix - Supplemental Graphs

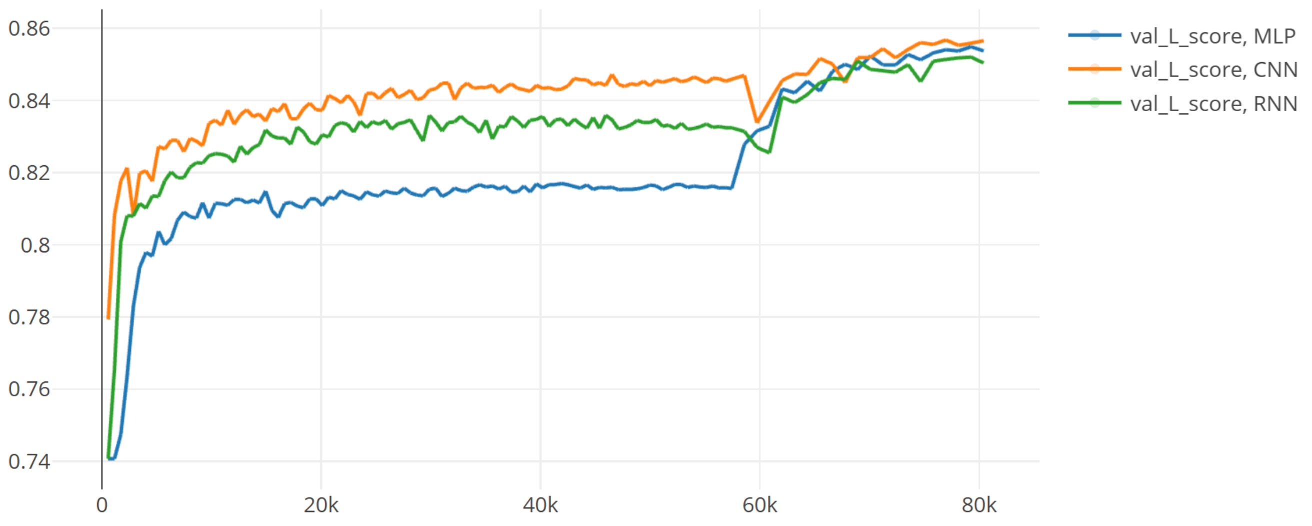

*Figure 1.* Validation scores of DistilBERT with various classification heads. Trained for 10 epochs with the encoder frozen, then 2 epochs with fine-tuning. X-axis shows update steps

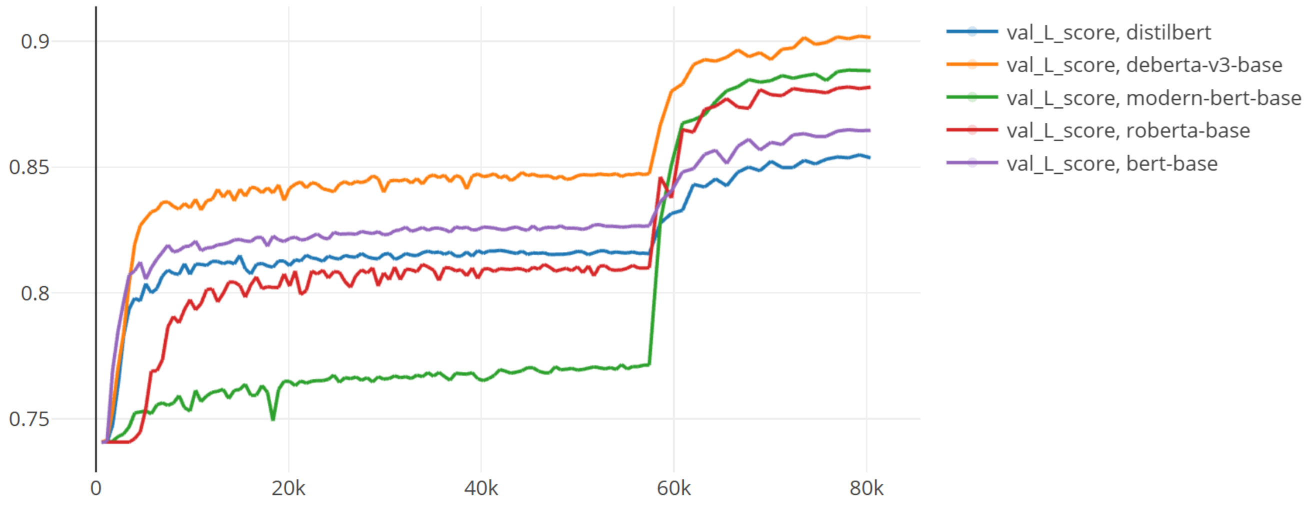

*Figure 2.* Validation scores of transformer encoders trained for 10 epochs frozen, then 2 epochs unfrozen. X-axis shows update steps

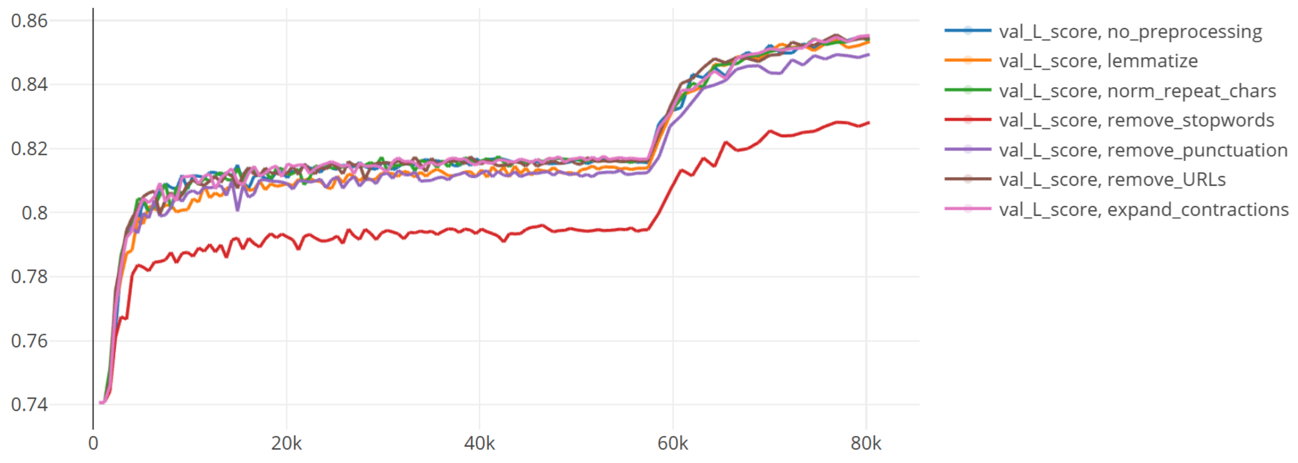

*Figure 3.* Validation scores of DistilBERT with various pre-processing techniques applied. Trained for 10 epochs with the encoder frozen, then 2 epochs with fine-tuning. X-axis shows update steps

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

## Declaration of originality

---

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

**Title of work** (in block letters):

Sentiment Analysis

**Authored by** (in block letters):
*For papers written by groups the names of all authors are required.*

| **Name(s):** | **First name(s):** |
|---|---|
| Ellis | Ben |
| Balfour | Bruce |
| Hütteneder | Jakob |
| Zogaj | Florian |
| | |

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

| **Place, date** | **Signature(s)** |
|---|---|
| Zurich, 30.05.2025 | [illegible] |
| | Bruce Balfour |
| | Jakob Hütteneder |
| | Zogaj Florian |
| | |

*For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.*