Sentiment Analysis of Multilingual Amazon Reviews

AI509/508/DM894 Final Project Report

Imanshu Sharma Alberto Aiello Luca Anzaldi

January 22, 2026

Abstract

This project focuses on sentiment analysis of Amazon product reviews in multiple languages. The goal is to train and evaluate neural language models to predict the sentiment expressed in customer reviews. We analyze how different factors influence model performance, including the amount of training data available for each language and the use of multilingual versus monolingual models. In particular, we study (i) how performance changes as the size of the training data increases, (ii) whether a single multilingual model can effectively handle multiple languages, and (iii) how performance varies across languages. Our experiments use a compact multilingual transformer model and provide a baseline evaluation of sentiment prediction on Amazon reviews.

1 Introduction and Problem Presentation

1.1 Task description

Given an Amazon review text, the goal is to predict its sentiment. We consider two related formulations:

- **Classification:** predict discrete sentiment labels (e.g., negative / neutral / positive or 1–5 stars).
- **Regression:** predict a continuous sentiment score derived from the 1–5 star rating.

From an NLP perspective, this task requires transforming raw review text into meaningful numerical representations that can be processed by neural models. Each review is first tokenized into subword units and mapped into embeddings that capture semantic information. These representations are then processed by transformer-based architectures, which use attention mechanisms to model relationships between words and capture sentiment cues across the entire review. This approach allows the model to handle long texts, negation, and context-dependent sentiment, which are common in customer reviews.

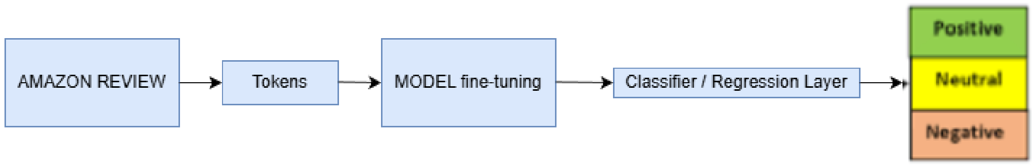


Figure 1: End-to-end sentiment analysis pipeline. An Amazon review is processed by the model tokenizer, mapped to embeddings, encoded by a transformer model, and passed to a task-specific prediction head.

1.2 Project goal and research questions

Our project investigates the following questions:

1. **How much target-language data is needed?** We vary training set size (e.g., 10%, 30%, 50%, 70%, 100%) and identify where performance saturataes.
2. **One multilingual model vs. monolingual models: which works better?** We compare a shared multilingual model trained on all languages against separate models trained per language.
3. **Per-language performance ranking:** We compute and compare evaluation metrics for each language to understand which languages are easier/harder.

1.3 Why the problem is challenging

- **Noisy and subjective labels:** Star ratings do not always reflect the sentiment expressed in the text. Users may assign high ratings while writing neutral reviews, or low ratings for reasons that are not directly related to the product quality.
- **Ordinal nature of the target:** Sentiment ratings from 1 to 5 stars are ordered, which means that prediction errors have different importance. For example, predicting 4 instead of 5 stars is less severe than predicting 1 instead of 5, which motivates the use of evaluation metrics that take ordering into account.
- **Context-dependent sentiment:** The sentiment of a word or sentence often depends on its context. Elements such as negation or mixed opinions make it difficult to determine

sentiment based on individual words, and require models that can capture information across the entire review.

- **Multilingual and cross-lingual variability:** Reviews are written in different languages, each with its own grammar, vocabulary, and writing conventions. This creates challenges for tokenization and representation learning, especially when using a single model across multiple languages.
- **Domain and style variation:** Amazon reviews cover many product categories and writing styles, ranging from very short comments to long and detailed descriptions. This diversity makes it harder to learn sentiment representations that generalize well.

2 Method

2.1 Computing resources (Hardware)

All experiments were executed on the SDU/University GPU cluster. In the table below are showed the *hardware specification.*

Component	**Specification**
Compute node	`u1-gpu-h-1`
CPU	Intel Xeon Gold 6230 (20 vCPUs)
RAM	46 GB
GPU	1 × NVIDIA V100
OS / Environment	Linux + Conda

Table 1: Hardware configuration used for model training and evaluation.

2.2 Software stack (Frameworks and tools)

We implemented the pipeline in Python using the following libraries:

- **PyTorch**: backend for model training and automatic differentiation.
- **Hugging Face Transformers**: model loading, tokenization, and the training loop (`Trainer`).
- **Hugging Face Datasets**: loading CSV splits and preprocessing with efficient mapping.
- **Evaluate**: metric computation during evaluation.
- **Weights & Biases (wandb)**: experiment tracking (loss curves, metrics, run metadata).
- **NumPy**: metric utilities and post-processing.

2.3 Dataset overview

We use a multilingual Amazon Reviews dataset [5] split into `train.csv`, `validation.csv`, and `test.csv`. Each review includes a 1–5 star rating and textual fields (title/body), along with metadata such as language and product category. The dataset covers multiple languages (e.g., EN, ES, DE, FR, JA, ZH).

2.3.1 Data fields

Field	**Description**
`review_id`	Unique identifier of the review.
`product_id`	Identifier of the product being reviewed.
`reviewer_id`	Identifier of the user who wrote the review.
`stars`	Integer rating from 1 to 5 stars assigned by the user.
`review_title`	Short title of the review.
`review_body`	Main textual content of the review.
`language`	Language in which the review is written.
`product_category`	Category of the reviewed product.

Table 2: Overview of the main fields in the Amazon Reviews dataset.

Non-essential metadata (e.g., review IDs, product identifiers, language tags, and product category fields) are removed for the baseline experiments.

2.3.2 Label distribution

The training split is perfectly balanced across the five star ratings, as shown in Table 3.

Star rating	Number of samples
1	240,000
2	240,000
3	240,000
4	240,000
5	240,000

Table 3: Label distribution in the training set (balanced).

The validation and test sets follow the same balanced distribution, with 6,000 samples per star rating.

2.3.3 Dataset split sizes

Table 4 summarizes the number of samples used in each split.

Split	Samples per class	Total samples
Training	240,000	1,200,000
Validation	6,000	30,000
Test	6,000	30,000

Table 4: Overview of dataset splits used in the experiments.

This balanced setup allows a fair comparison between regression and classification approaches and ensures that evaluation metrics are not biased by class imbalance.

2.4 Model choice

We use `distilbert-base-multilingual-cased`, an **encoder-only** Transformer model. It is a distilled variant of multilingual BERT, providing a good trade-off between computational efficiency and multilingual coverage [2, 3].

Feature	**Specification**
Model name	`distilbert-base-multilingual-cased`
Architecture	Encoder-only Transformer (DistilBERT)
Number of layers	6 Transformer encoder layers
Hidden size	768
Attention heads	12
Total parameters	$\sim$134 million
Vocabulary size	$\sim$119,547
Maximum sequence length	512 tokens
Pretraining objective	Masked Language Modeling (MLM)
Languages covered	100+ languages
Case sensitive	Yes (cased)
Disk size (weights)	$\sim$517 MB

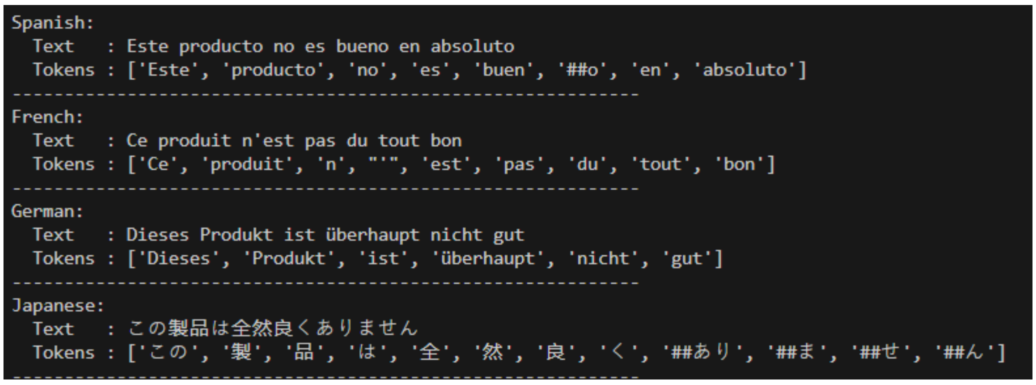
Table 5: Main characteristics of the `distilbert-base-multilingual-cased` model [6]

2.5 Preprocessing and input representation

Column selection. We rename `stars` to `label` and construct the input text by concatenating the review title and body (`text` = `review_title` + `review_body`). Non-essential metadata (e.g., review IDs, product identifiers, language tags, and product category fields) are removed for the baseline experiments.

Tokenization. The input text is processed using the tokenizer associated with the pre-trained model. This tokenizer applies a subword tokenization strategy, which splits the text into smaller units that belong to a shared multilingual vocabulary. Subword tokenization helps handle rare words, spelling variations, and text written in different languages, and ensures compatibility with the model vocabulary. Truncation is applied to limit the maximum sequence length, while dynamic padding is used to efficiently batch inputs during training.

Model vocabulary. The model uses a shared multilingual subword vocabulary rather than separate vocabularies for each language. This design allows the same tokens or subword units to be reused across different languages, enabling cross-lingual transfer. As a result, the model can process reviews written in multiple languages using a single tokenizer and a unified representation space.



```
Spanish:
  Text   : Este producto no es bueno en absoluto
  Tokens : ['Este', 'producto', 'no', 'es', 'buen', '##o', 'en', 'absoluto']
----------------------------------------------------------
French:
  Text   : Ce produit n'est pas du tout bon
  Tokens : ['Ce', 'produit', 'n', "'", 'est', 'pas', 'du', 'tout', 'bon']
----------------------------------------------------------
German:
  Text   : Dieses Produkt ist überhaupt nicht gut
  Tokens : ['Dieses', 'Produkt', 'ist', 'überhaupt', 'nicht', 'gut']
----------------------------------------------------------
Japanese:
  Text   : この製品は全然良くありません
  Tokens : ['この', '製', '品', 'は', '全', '然', '良', 'く', '##あり', '##ま', '##せ', '##ん']
----------------------------------------------------------
```

Figure 2: Example tokenization of Amazon review sentences in different languages using the `distilbert-base-multilingual-cased` tokenizer. The figure shows how a shared multilingual subword vocabulary is used across languages.

Label normalization. In the *regression* setting, the original star ratings $y \in \{1, 2, 3, 4, 5\}$ are linearly normalized to the interval $[0, 1]$:

$$y_{\text{norm}} = \frac{y - 1}{4}.$$

For evaluation and interpretability, model predictions are mapped back to the original rating scale using the inverse transformation:

$$\hat{y} = 4 \cdot \hat{y}_{\text{norm}} + 1.$$

In the *classification setting*, star ratings are re-encoded from the original range $\{1, \ldots, 5\}$ to $\{0, \ldots, 4\}$, in accordance with the Hugging Face `Transformers` framework, which requires class labels to be zero-indexed.

2.6 Training setup

Training is performed using task-specific loss functions. For the *regression formulation*, we minimize the **Mean Squared Error (MSE)** loss between predicted and ground-truth star ratings. For the *classification formulation*, we optimize the **categorical cross-entropy** loss, as standard for multi-class classification in the Hugging Face `Transformers` framework.

Hyperparameter	**Value**
Training batch size	32
Gradient accumulation steps	4
Effective batch size	128
Number of epochs	8
Weight decay	0.01
Learning rate scheduler	Cosine
Warmup ratio	0.06
Maximum gradient norm	1.0

Table 6: Key training hyperparameters used in the experiments.

Hyperparameter selection.

- The training batch size was set to 32 to ensure stable optimization while avoiding GPU memory constraints.

- Gradient accumulation with four steps was used to simulate a larger effective batch size of 128 without increasing memory usage.
- The model was trained for eight epochs to allow sufficient convergence while limiting the risk of overfitting.
- Weight decay was applied to regularize model parameters and mitigate overfitting on noisy sentiment labels.
- A cosine learning rate scheduler with a warmup ratio of 0.06 was adopted to ensure smooth learning rate decay and stable early training.
- Gradient clipping with a maximum norm of 1.0 was employed to prevent exploding gradients and improve overall training stability.

2.7 Evaluation metrics

Evaluation is performed using different metrics depending on whether the task is formulated as **regression** or **classification**.

2.7.1 Regression setting.

When modeling sentiment as a continuous variable, we report the following metrics:

- **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE)**, computed in the original 1–5 star scale, to measure average prediction error.
- **Rounded Accuracy (5-class)**, obtained by rounding the continuous prediction to the nearest integer star rating.
- **Rounded Accuracy (3-class)**, computed after mapping 1–2→negative, 3→neutral, and 4–5→positive.
- **Spearman's rank correlation coefficient** (ρ), to assess how well the model preserves the ordinal ranking of sentiment.
- **Quadratic Weighted Kappa (QWK)**, which measures ordinal agreement while penalizing larger rating discrepancies more heavily.

2.7.2 Classification setting.

When sentiment prediction is formulated as a classification task, we report:

- **Accuracy (5-class)** and **Macro F1-score (5-class)**, to evaluate exact label prediction performance across all star ratings.
- **Quadratic Weighted Kappa (QWK)**, to account for the ordinal structure of the 1–5 star labels.
- **Accuracy (3-class)** and **Macro F1-score (3-class)**, computed after mapping ratings to negative, neutral, and positive sentiment.

2.8 Experiment protocol

2.8.1 Regression vs Classification

The first set of experiments was designed to assess whether sentiment prediction should be formulated as a *regression* or a *classification* problem. Since the Amazon review ratings are ordinal in nature, both approaches are viable: classification treats each star rating as a discrete category, while regression models sentiment as a continuous variable that preserves the ordering between ratings.

2.8.2 How much target-language data is needed

To investigate how model performance scales with the amount of target-language data, we trained the regression model using progressively larger subsets of the training set, namely 100k, 200k, 600k, and 1.2M samples.

This experiment aims to identify potential performance plateaus, i.e., points beyond which adding more training data yields diminishing returns. Understanding this behavior is essential to balance predictive performance and computational cost, and to inform the choice of dataset size for subsequent experiments. In particular, detecting an early plateau allows us to reduce training time and resource usage while maintaining competitive performance in later cross-lingual analyses.

2.8.3 One multilingual model vs. monolingual models: Which works better?

One of the main challenges in multilingual sentiment analysis is deciding whether to use one multilingual model or separate models for each language. Both options have their pros and cons.

A multilingual model is trained on reviews in many languages at once, which helps it learn common patterns across different languages. The big advantage of this approach is that it can handle multiple languages with just one model, saving resources. Also, multilingual models can benefit from cross-lingual transfer, meaning that knowledge learned in one language can help improve performance in other languages, especially for languages with fewer resources.

On the other hand, monolingual models are trained separately for each language, so they can focus on the specific features of each language. This might lead to better performance in languages with lots of data, because the model can specialize in each language's unique structure, grammar, and vocabulary. In our experiments, we compare a multilingual model trained on reviews from all languages with separate monolingual models trained on reviews from individual languages. By using the same amount of data for both, we can see how well the multilingual model handles the differences between languages compared to monolingual models that focus on just one language.

2.8.4 How much target-language data is needed

To investigate how model performance scales with the amount of target-language data, we trained the regression model using progressively larger subsets of the training set, namely 100k, 200k, 600k, and 1.2M samples.

This experiment aims to identify potential performance plateaus, i.e., points beyond which adding more training data yields diminishing returns. Understanding this behavior is essential to balance predictive performance and computational cost, and to inform the choice of dataset size for subsequent experiments. In particular, detecting an early plateau allows us to reduce training time and resource usage while maintaining competitive performance in later cross-lingual analyses.

2.8.5 One multilingual model vs. monolingual models: Which works better?

One of the main challenges in multilingual sentiment analysis is deciding whether to use one multilingual model or separate models for each language. Both options have their pros and cons.

A multilingual model is trained on reviews in many languages at once, which helps it learn common patterns across different languages. The big advantage of this approach is that it can handle multiple languages with just one model, saving resources. Also, multilingual models can benefit from cross-lingual transfer, meaning that knowledge learned in one language can help improve performance in other languages, especially for languages with fewer resources.

On the other hand, monolingual models are trained separately for each language, so they can focus on the specific features of each language. This might lead to better performance in languages with lots of data, because the model can specialize in each language's unique structure, grammar, and vocabulary. In our experiments, we compare a multilingual model trained on reviews from all languages with separate monolingual models trained on reviews from individual languages. By using the same amount of data for both, we can see how well the multilingual model handles the differences between languages compared to monolingual models that focus on just one language. The data is split as follows:
- For the monolingual model, 500k reviews are selected from a single language (e.g., English) and used to train the model.
- For the multilingual model, 500k reviews are distributed across multiple languages (e.g., English, Spanish, French, and German) to train the model.
The same model architecture is used for both types of models: distilbert-base-multilingual-cased, a transformer-based model. This model has been pre-trained on a multilingual corpus and fine-tuned for the task of sentiment analysis.

2.8.6 Per-language performance ranking

HIMANSH
or AL-
BERTO

3 Results

3.0.1 Regression vs Classification

Table 7 shows that the overall performance of the regression and classification approaches is very similar across most evaluation metrics. In particular, metrics that capture ordinal consistency and ranking quality, such as Quadratic Weighted Kappa (QWK) and Spearman correlation, exhibit only marginal differences between the two settings.

These similarities are also reflected during inference, where both models generally predict comparable star ratings for clearly positive or negative reviews. However, regression achieves slightly higher QWK and Spearman scores, suggesting a better preservation of the ordinal structure of the rating scale.

For this reason, despite the comparable classification-oriented metrics (Accuracy and F1), we ultimately chose the regression formulation. This choice is motivated by its ability to produce continuous scores that better capture sentiment intensity and reduce the impact of small ordinal errors (e.g., predicting 4 instead of 5 stars), which are penalized less severely by ordinal metrics.

Metric	**Regression**	**Classification**
Dataset size	1.2M	1.2M
MAE ↓	0.508	–
RMSE ↓	0.692	–
QWK ↑	0.854	0.853
Spearman ↑	0.855	0.854
F1 (5-class) ↑	0.740	0.614
F1 (3-class) ↑	0.603	0.746
Accuracy (5-class) ↑	0.596	0.616
Accuracy (3-class) ↑	0.784	0.792

Table 7: Comparison between regression and classification

3.0.2 Qualitative Comparison

The following examples illustrate the behavior of the two approaches at inference time.

Regression

- **Review: VALID** — "This product is very valid, I suggest this to everyone" Predicted score (normalized): 4.83 Predicted stars (rounded): 5/5
- **Review: NOT VALID** — "Pls don't buy this product, it is a scam" Predicted score (normalized): 1.01 Predicted stars (rounded): 1/5
- **Review: IT OKS** — "This product has a nice cost/quality rate even if it is not the best one" Predicted score (normalized): 3.68 Predicted stars (rounded): 4/5
- **Review: DISSAPOINTED** — "This product is not good. I could say that it is acceptable only because it is very cheap." Predicted score (normalized): 1.27 Predicted stars (rounded): 1/5

Classification

- **Review: VALID** Predicted stars: 5/5 Confidence (softmax): 0.66
- **Review: NOT VALID** Predicted stars: 1/5 Confidence (softmax): 0.99
- **Review: IT OKS** Predicted stars: 3/5 Confidence (softmax): 0.47
- **Review: DISSAPOINTED** Predicted stars: 1/5 Confidence (softmax): 0.80

These examples highlight how regression naturally captures intermediate sentiment levels (e.g., borderline positive reviews), while classification tends to force discrete decisions even when confidence is relatively low.

3.0.3 Results about dataset size

From Table 8, a consistent improvement across all metrics can be observed as the dataset size increases. No clear performance plateau is reached within the explored range, indicating that reducing the dataset size leads to faster training but inevitably results in a loss of predictive quality.

Dataset size	Epochs	MAE ↓	RMSE ↓	QWK ↑	Acc (5) ↑	Acc (3) ↑
100k	4	0.600	0.791	0.813	0.527	0.743
200k	5	0.554	0.754	0.835	0.568	0.766
600k	4	0.526	0.710	0.848	0.585	0.778
1.2M	4	0.508	0.692	0.854	0.596	0.784

Table 8: Impact of dataset size on regression performance

Moreover, the best values for all major metrics are already achieved between 4 and 5 epochs. Extending training beyond this range does not provide additional benefits and may increase the risk of overfitting while unnecessarily increasing computational cost. For this reason, early stopping around 4 epochs is recommended, rather than training up to 8 epochs (Figure 3).

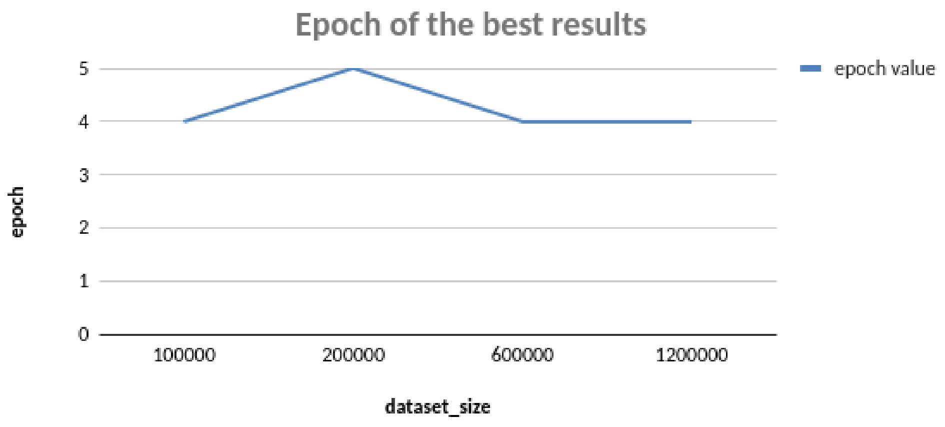


Figure 3: Epoch related to the dataset size

Overall, these results suggest a trade-off between efficiency and performance: smaller datasets significantly reduce training time but degrade ordinal consistency and accuracy, while larger datasets continue to yield measurable gains without exhibiting saturation.

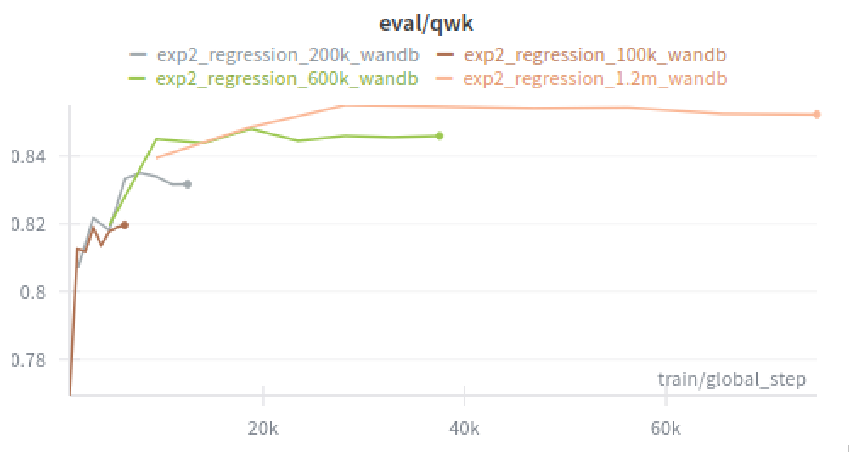


Figure 4: QWK metrics

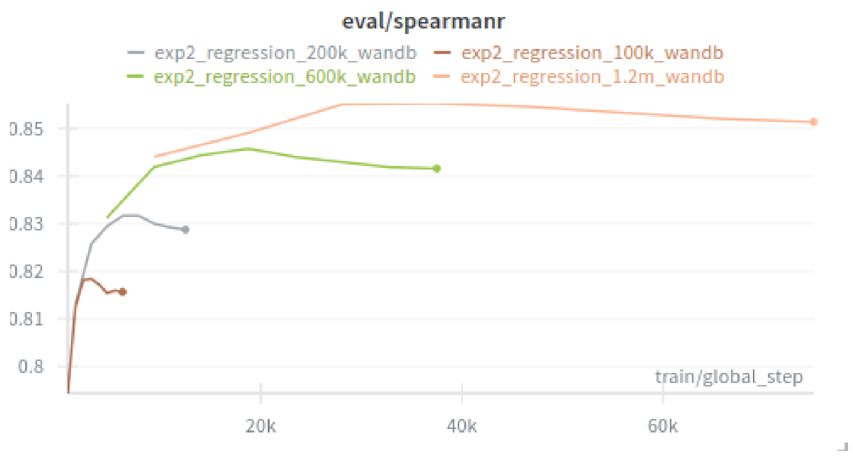


Figure 5: Spearman metrics

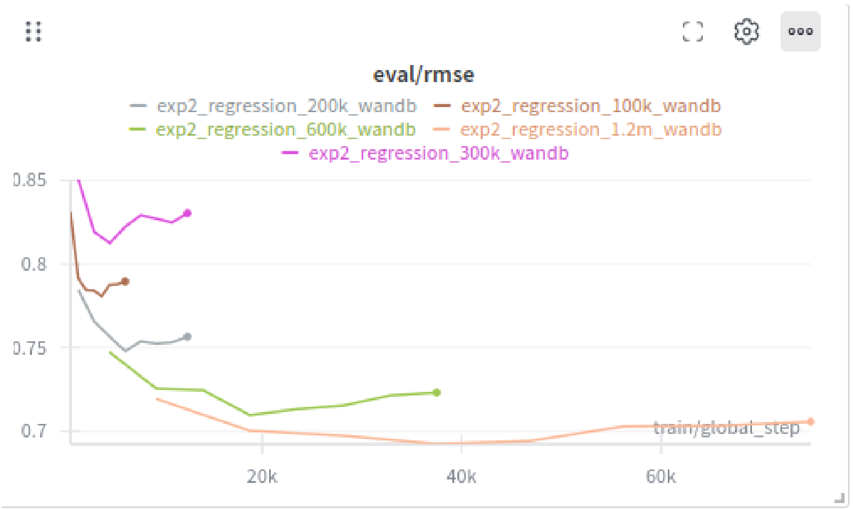


Figure 6: RMSE metrics

4 Discussion

4.1 Key findings

Our baseline confirms that a multilingual encoder fine-tuned as a regressor can capture sentiment intensity and ordinal structure...

Finish this chapter using the result of question2 and question3

5 Conclusion

We presented a baseline pipeline for multilingual sentiment prediction on Amazon Reviews and validated training efficiency on a GPU cluster. Using a multilingual DistilBERT encoder, the regression setup achieves strong ordinal agreement (QWK) and rank preservation (Spearman), supporting the feasibility of larger-scale cross-lingual experiments. Future work will systematically evaluate data requirements per language and compare multilingual vs monolingual training strategies.

References

[1] Devlin J., Chang M.-W., Lee K., Toutanova K., *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[2] Sanh V., Debut L., Chaumond J., Wolf T., *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*, *arXiv preprint* arXiv:1910.01108, 2019.

[3] Wolf T., Debut L., Sanh V., et al., *Transformers: State-of-the-Art Natural Language Processing*, *Proceedings of EMNLP: System Demonstrations*, 2020, pp. 38–45.

[4] Biewald L., *Experiment Tracking with Weights and Biases*, *Software available from wandb.com*, 2020.

[5] Mexwell (Kaggle uploader), *Amazon Reviews Multi (Kaggle dataset)*, *Kaggle*, year unknown. `https://www.kaggle.com/datasets/mexwell/amazon-reviews-multi/data`

[6] Hugging Face, *distilbert-base-multilingual-cased: Model card*, Hugging Face Model Hub, 2019. Available at: `https://huggingface.co/distilbert-base-multilingual-cased`

A Reflection with respect to learning objectives

This project aligns with the course learning objectives by requiring:

- Understanding and application of core NLP building blocks, including tokenization, embeddings, and Transformer architectures (attention mechanisms) as the foundation of modern language understanding systems;
- Implementation and fine-tuning of pre-trained language models (e.g., with Hugging Face and/or PyTorch) to solve practical NLP tasks such as text classification and sentiment analysis (and, where applicable, sequence-to-sequence tasks like translation);
- Comparison of alternative modeling choices by analyzing evaluation metrics and qualitative outputs, assessing their effectiveness for language modeling and sequence generation behavior where relevant;
- Critical assessment of strengths and limitations of different NLP approaches, considering model capacity, scalability, computational cost, and robustness to real-world, domain-specific language variation;
- Development of an end-to-end, scalable NLP solution that integrates preprocessing, embeddings, and advanced architectures, with attention to deployment considerations and reproducible experimentation.

B Work tasks distribution

Member	**Main contributions**
Imanshu Sharma	TODO
Alberto Aiello	TODO
Luca Anzaldi	Regression setup, metrics (QWK/Spearman), training pipeline, test on different dataset size, report

Table 9: Work distribution