Advancing an Empathy Rater System in NLP: Evaluation Architectures and Loss Functions for Text Pair Rating

Minoo Shayan

November 15, 2024

This project, undertaken as part of a **Directed Studies course in Fall 2023** and supervised by **Professor Steve DiPaola**, aims to explore and evaluate various architectures and loss functions to inform the development of an effective empathy rater system.

Contents

1 Introduction

1.1 Context and Motivation

In domains like healthcare and education, AI systems are increasingly expected to convey empathy in their interactions. Empathy in text-based AI interactions can improve user trust, provide emotional support, and make systems more effective in addressing user needs. However, empathy detection in AI faces several challenges. Empathy datasets are limited and often lack sufficient labeling, making it difficult to train AI systems on empathy recognition tasks. Moreover, empathy itself remains largely undefined and subjective, with significant variation in how it is perceived between human and AI interactions compared to human-to-human interactions.

1.2 Problem Statement

Detecting empathy in AI is challenging due to the subjective and complex nature of empathy itself. Defining empathy for AI systems is inconsistent and is further complicated by the scarcity of high-quality, labeled datasets. Existing datasets often use subjective or poorly defined annotation methods for empathy. This challenge affects how empathy can be rated—whether as a classification task (identifying discrete empathy levels) or a regression task (scoring empathy levels on a continuous scale). Other approaches view the problem as a binary classification of empathy versus no empathy. Furthermore, existing methods, such as large language models (LLMs) used as "empathy raters" in zero-shot scenarios or LLM-as-judges, lack a standardized approach, leading to ambiguous and often unreliable evaluations.

1.3 Project Objective

This project aims to develop a standardized empathy detection system for text interactions. By comparing different empathy detection models, this system seeks to establish a reliable foundation for evaluating AI empathy. With a unified approach, AI systems could be rated consistently, rather than relying on varying criteria across different systems or domains. This would help address the current gap, where any conversational agent can claim empathy without a reliable basis for assessment.

2 Dataset Selection and Preprocessing

2.1 Dataset Overview and Initial Exploration

To build a foundation of empathy-related datasets for this project, I researched papers on empathy in AI, cataloged referenced datasets, and reached out to institutions and authors. This process underscored a notable gap: empathy datasets lack a structured review, making it challenging to compare and select datasets effectively. This discovery presents an opportunity to start organizing and analyzing empathy datasets for more structured future research. See table of the empathic datasets that I have collected as a result of this project in the project plan in file `Project_Plan_for_Directed_Studies___Fall_2024.pdf` in Appendix C.

Among the datasets gathered, **Empathic Stories++ (ES++)**[1] and **EPOTIMDE (EP)**[2] were chosen for their relevance and complementary perspectives on empathy. Below are further details on each dataset, along with the preprocessing steps applied.

2.1.1 Empathic Stories++ (ES++) Dataset

The Empathic Stories++ (ES++) dataset is a multimodal collection designed to capture expressions of empathy in personal storytelling contexts. Created through a month-long in-home study, participants interacted with an AI agent and were prompted to share reflections on emotionally significant or challenging personal experiences. This setup encouraged participants to express themselves in a comfortable, natural environment, resulting in authentic examples of empathy.

	Shared_stories	Response_strories	Rate
0	My family. I had a family of four siblings, tw...	I grew up in Nanjing, China. When I was a baby...	5
1	Well, I\'m going to tell you since my last sto...	We were rolling around selling tee shirts, and...	4
2	Okay. I was married when I was young, and unfo...	I have been dating Michelle for nearly three y...	3
3	So yeah. So, I can share a story that happened...	My wife and I traveled to Italy and did an ama...	3
4	So I\'d like to tell a story about how I reall...	I started off in physics, electrical engineeri...	2
...	...	...	...
263	Yeah, I\'ve been reaching out to more people a...	I was dating this woman who was really adventu...	2
264	Like, I think one story or just like an intere...	Nineteen years ago, I started working for SLUG...	1
265	Yesterday, yesterday, we went to another frien...	I recently went on vacation with my family. We...	3
266	Hello. The story I'd like to tell is about my ...	I was treated for cancer this year, and since ...	1
267	One experience I'd like to share is that of tr...	I'm the first person out of four living genera...	3

804 rows × 3 columns

Figure 1: Empathic Stories++ Data Overview

2.1.1.1 Key Features

- **Self-Reported Empathy Scores:** After each story, participants rated the perceived empathy level on a 1–3 scale, which added a subjective dimension, reflecting individual perceptions of empathy.
- **Multimodal Data:** Although ES++ includes multiple data types, only the textual stories and empathy scores were used in this project.

[1] Shen, J., Kim, Y., Hulse, M., Zulfikar, W., Alghowinem, S., Breazeal, C., & Park, H. W. (2024). EmpathicStories++: A Multimodal Dataset for Empathy towards Personal Experiences. https://arxiv.org/pdf/2405.15708

[2] Sharma, A., Miner, A. S., Atkins, D. C., & Althoff, T. (2020). A computational approach to understanding empathy expressed in text-based mental health support. arXiv preprint arXiv:2009.08441.

- **Naturalistic Setting:** Collected in participants' homes, the dataset provides spontaneous expressions of empathy in real-life contexts, making it relevant for empathy detection tasks.

2.1.1.2 Dataset Analysis

An in-depth analysis of the dataset was conducted to understand its structure and distribution of empathy ratings. Key insights from this analysis include:

- **Empathy Score Distribution:** The original 1–3 empathy ratings are unevenly distributed, with more stories assigned lower empathy scores. This imbalance presents a challenge for training, as lower-scored examples dominate the dataset.
- **Rescaling for Regression:** For regression tasks, empathy scores were mapped to a 0–1 scale to create a continuous target variable. This transformation provided a finer-grained label, allowing the model to predict nuanced empathy levels.
- **Text Length and Content Variation:** The stories vary in length and content, with some brief entries and others more detailed. This variability offers diverse examples but may require careful handling during training to prevent model biases toward shorter or longer responses.

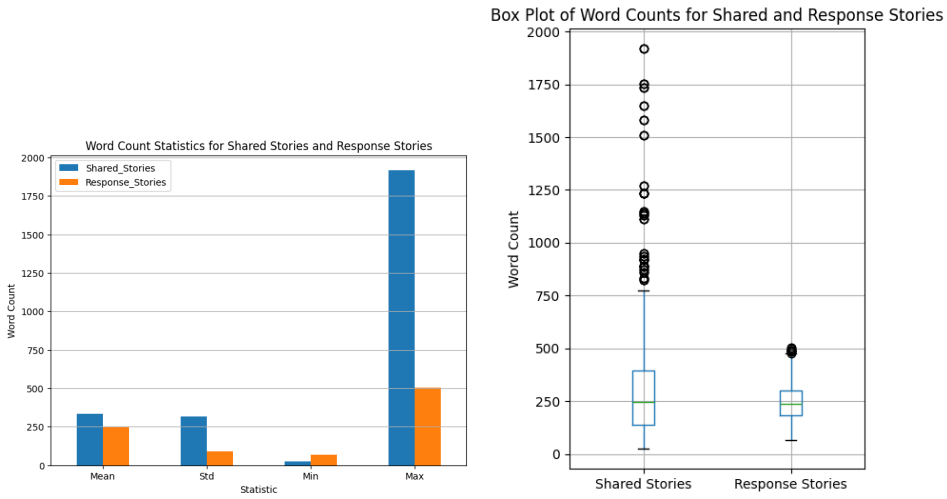

Figure 2: Word count analysis in Stories and responses in ES++.

Figure 3 and Figure 2 shows the analysis of word count and distribution of labels across this dataset. These insights guided the selection of preprocessing and training strategies to improve model performance, such as rescaling empathy ratings and managing the class imbalance.

2.1.1.3 Limitations

- **Small Dataset:** After matching stories to samples, the dataset includes around 800 entries, which limits its capacity for training large models.
- **Subjectivity:** The empathy ratings are self-assessed, introducing variability and lacking an objective standard for defining empathy.
- **Limited Scale:** The 1–3 empathy rating scale may restrict the model's ability to detect subtle differences in empathy levels.
- **Topic Bias:** The dataset primarily focuses on reflections about personal challenges, which may not generalize well to other narrative types.
- **Exclusion of Non-Textual Cues:** Despite being multimodal, only text data was utilized, omitting nonverbal cues that are often integral to conveying empathy.

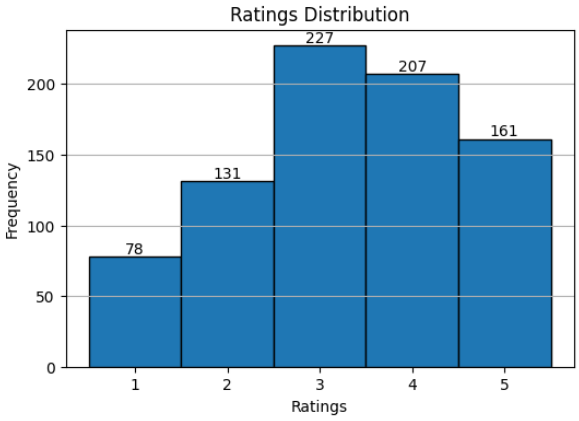

Figure 3: Distribution of Labels in ES++.

2.1.2 EPITOME Dataset

The EPITOME dataset was originally designed to support a range of affective and empathetic analysis tasks, with a specific focus on understanding emotional responses within text. Developed as part of research on natural language understanding, EPITOME includes various forms of textual data annotated for empathy, emotion, and sentiment. This dataset is distinguished by its structured empathy annotations, which categorize empathy into discrete levels based on defined guidelines. EPITOME draws from diverse sources, covering a variety of topics and emotional contexts. These sources were selected to capture a broad spectrum of affective states and empathy expressions, making the dataset versatile for analyzing empathy in varied scenarios. Empathy levels were annotated by trained experts rather than self-assessment, ensuring that empathy ratings were based on consistent criteria. Annotators assessed empathy in each text using a defined empathy framework, categorizing responses into three levels: low empathy (0), moderate empathy (1), and high empathy (2). This method of annotation minimizes subjectivity and provides a structured basis for empathy detection, although it may limit the subtleties of personal empathy perception.

	sp_id	rp_id	seeker_post	response_post	level
0	65m92s	dgbdk7z	Help. Help me. I dunno what I'm doing anymore	That's pretty vague, do you not know what you'...	2
1	9ezsfi	e5t3oxh	I'm done saying I love you to her because I do...	idk what a Red pill means exactly but my advic...	0
2	6b2cmc	dhj8tcb	Always feel like I'm being criticized and mock...	I think it's social anxiety , that creates par...	0

Figure 4: EPITOME Data Overview

2.1.2.1 Key Features

- **Expert-Annotated Empathy Levels:** By using trained annotators, the dataset provides a consistent and standardized view of empathy levels, reducing variability that could arise from personal interpretations.
- **Diversity of Emotional Topics:** The texts span a range of themes, emotions, and situations, offering a broad basis for empathy detection models to generalize across multiple contexts.
- **Structured Categories:** The empathy levels are discrete and predefined, making the dataset well-suited for classification tasks, particularly for categorical empathy detection.

2.1.2.2 Dataset Analysis

Figure 5 and Figure 6 shows the analysis of word count and distribution of labels across this dataset.Analysis of the dataset in the provided notebook revealed significant insights into label distribution and struc-

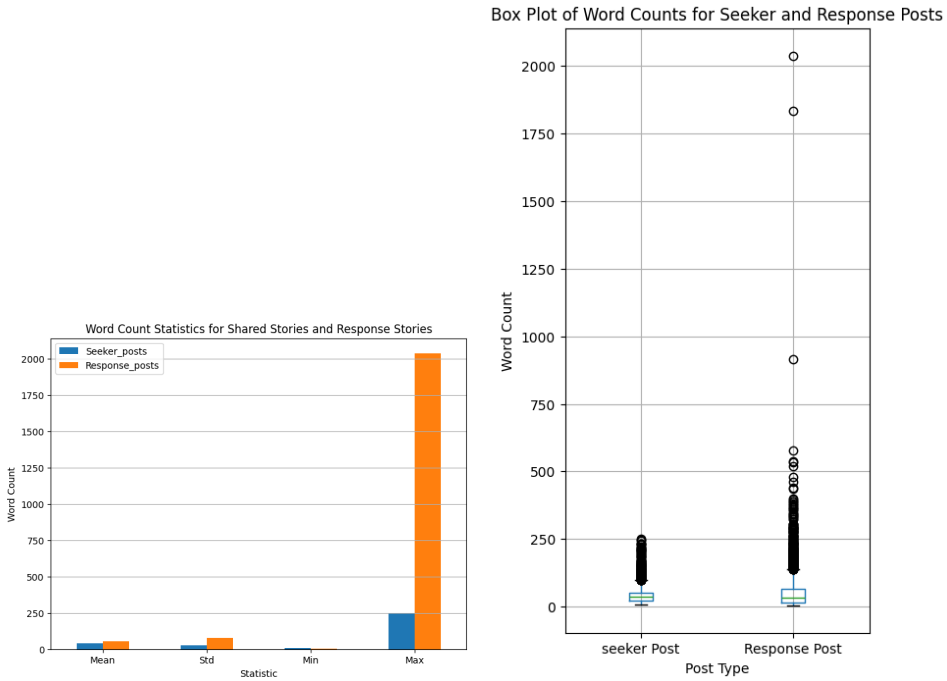

Figure 5: Word count analysis in posts and responses in EP.

ture:

- **Empathy Level Distribution:** The dataset contains an imbalanced distribution across empathy categories, with fewer samples in the higher empathy level (2). This imbalance presents challenges for model training, as the model is less exposed to high-empathy examples.
- **Label Transformation for Classification:** To address label imbalance, the initial empathy labels (0, 1, and 2) were transformed by summing them into a broader 0-6 scale. However, due to the low frequency of high empathy scores in the resulting range, all scores above 3 were grouped into a single "high-empathy" category. This final transformation resulted in a four-level scale (0–3) for empathy classification, enhancing label balance while preserving the original distinctions between empathy levels.

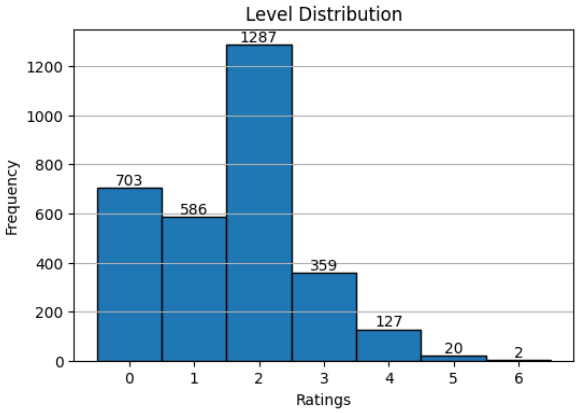

Figure 6: Distribution of Labels in EP.

2.1.2.3 Limitations

- **Small Sample Size in Higher Labels:** Even after rescaling, the high-empathy category remains underrepresented, providing limited training examples for the model in this class.
- **Class Imbalance:** The categorical distribution is not perfectly balanced, with low empathy levels more frequently appearing than high empathy levels. This imbalance could introduce bias in model predictions.
- **Objective Annotations:** Although the expert annotations provide consistency, they may not capture the subjective nuances of empathy as perceived by individuals, potentially narrowing the model's ability to recognize empathy from diverse perspectives.

The EPITOME dataset's structured empathy annotations and broad thematic range provide a foundational resource for empathy classification, particularly when used alongside more subjective datasets like Empathic Stories++. This structured approach complements subjective datasets, creating a comprehensive foundation for evaluating and training empathy detection models.

2.2 Adaptation for This Project

For this project, the empathy ratings were rescaled to a 0–1 range for a regression task, creating a continuous empathy score. This adapted dataset was then uploaded to the Hugging Face Hub for use in training and evaluation tasks. Despite its limitations, ES++ is one of the first and most recent datasets focused on empathy in text, offering a unique perspective on empathy as perceived in personal narratives. Its real-world origin and subjective ratings provide valuable insights, contributing to advancements in empathy detection within natural language processing.

In adapting the EPITOME dataset for this project, adjustments were made to better align the dataset with empathy classification goals and to manage class imbalance effectively. Initially, empathy labels were transformed from a 3-level system (0–2) into a broader 7-level scale by summing available empathy scores. However, due to underrepresentation in higher empathy levels, a final restructuring was implemented, combining scores above 3 into a single category to form a four-level system (0–3). This transformation not only improved label balance but also retained critical distinctions between low, moderate, and high empathy levels.

This preprocessing step was essential to enable a more balanced training process, where each empathy level had a more proportional representation, enhancing the model's capacity to generalize across different levels of empathy. For more details of the data analysis and data preprocessing performed on these datasets, please refer to file `DataPreparation.ipynb` in Appendix C. This modified dataset was then uploaded to Hugging Face's Hub to serve as a consistent dataset for empathy classification tasks within the project (Refer to file `DatasetHF.py` in Appendix C.)

- `minoosh/EPITOME_pairs`
- `minoosh/Annotated_story_pairs2`

3 Modeling and Implementation

3.1 Model Choice and Architecture

To address the challenge of empathy detection between text pairs, I employed transformer-based models, specifically BERT, as a foundation. This task required not only analyzing each text in isolation but also understanding the interaction between two texts, either classifying empathy or scoring it on a continuous scale. To accomplish this, I implemented both **bi-encoder** and **cross-encoder** architectures, each with a distinct approach for encoding and interpreting text pairs. Both architectures were built to enable flexibility, with the ability to swap out the base model (e.g., from BERT to RoBERTa) as required.

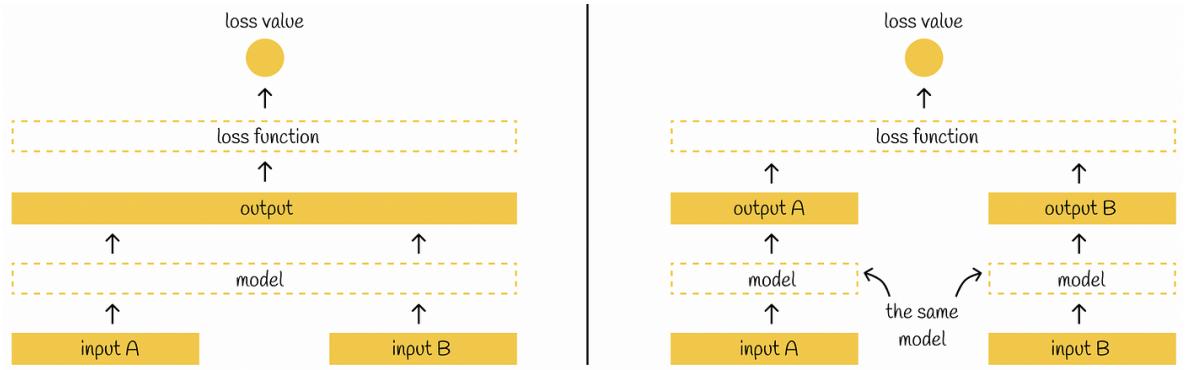

Figure 7: Bi-Encoder (right) and Cross-Encoder (left) Architectures for Processing Multiple Input Sentences. (*Image taken from https://towardsdatascience.com/sbert-deb3d4aef8a4*)

3.1.1 Bi-Encoder Architecture

In this setup, each text is encoded separately by BERT, producing individual embeddings that are combined before being passed through either a classification or regression head. This structure is effective for tasks where the comparison of separately encoded texts, like similarity tasks, is beneficial. In empathy detection, it allows the model to independently assess each text before comparing their embeddings.

3.1.2 Cross-Encoder Architecture

Here, the two texts are concatenated with a `[SEP]` token, creating a single input sequence that is jointly processed by BERT. This architecture enables the model to examine both texts in the same context, which is useful for tasks requiring close interaction, such as entailment or dialogue. The cross-encoder setup thus offers a more holistic, context-sensitive approach, ideal for evaluating empathy levels between text pairs.

3.2 Implementation Details

3.2.1 Tools

A combination of tools and libraries supported the development and fine-tuning of the models:

- **PyTorch** for constructing the custom models and handling training functions.
- **Hugging Face Transformers** for providing pre-trained BERT models, tokenizers, and training utilities.
- **Hugging Face Datasets** to facilitate efficient dataset loading and preprocessing.
- **Weights & Biases (WandB)** for tracking metrics, hyperparameters, and monitoring training progress.
- **Scikit-Learn and SciPy** for implementing various evaluation metrics, including classification and regression measures.
- **Hugging Face Hub** for storing, sharing, and retrieving models, ensuring easy accessibility.

Model training used Kaggle's dual T4 GPUs, with prediction and testing on a Colab T4. Preprocessing was handled locally, though exact processing times weren't recorded.

3.2.2 Custom Bi-Encoder and Cross-Encoder Classes

3.2.2.1 BiEncoderModel Class

- **Architecture:** This model processes each text separately, generating independent embeddings that are concatenated and passed through a final layer, depending on whether the task is classification or regression.
- **Classification Head:** For classification, the concatenated embeddings are passed through a linear layer to produce class probabilities. This setup includes several loss functions to handle class imbalance or soft-label classification, including cross-entropy, focal loss, and KL divergence.
- **Regression Head:** For regression, the model calculates cosine similarity between the embeddings, outputting a continuous score representing the empathy relation between the texts. Various regression-specific loss functions, like Mean Squared Error (MSE), Mean Absolute Error (MAE), and cosine embedding loss, were tested to identify optimal performance.

```python
class BiEncoderModel(torch.nn.Module):
    def __init__(self, base_model, num_classes=4, loss_fn="cross_entropy"):
        super(BiEncoderModel, self).__init__()
        self.base_model = base_model
        self.classifier = torch.nn.Linear(base_model.config.hidden_size * 2, num_classes)
        self.loss_fn = loss_fn

    def forward(self, input_ids_text1, attention_mask_text1, input_ids_text2, attention_mask_text2, labels=None):
        # Independent encoding of text1 and text2
        outputs_text1 = self.base_model(input_ids_text1, attention_mask=attention_mask_text1)
        outputs_text2 = self.base_model(input_ids_text2, attention_mask=attention_mask_text2)

        # Concatenation of embeddings for final scoring or classification
        concatenated_embeddings = torch.cat(
            [outputs_text1.last_hidden_state[:, 0, :], outputs_text2.last_hidden_state[:, 0, :]], dim=1
        )
        logits = self.classifier(concatenated_embeddings)
        loss = self.compute_loss(logits, labels) if labels is not None else None
        return {"loss": loss, "logits": logits}

    def compute_loss(self, logits, labels):
        # Defines loss calculation based on the chosen loss function
        pass
```

Figure 8: Overview of the Bi-Encoder Class

Figure 8 provides a sample Python code Bi-Encoder class, showcasing its implementation with a classification head for processing input data. For more details, refer to the file `ClassifierBiEncoder.py` and `RegressorBiEncoder.py` in Appendix C.

3.2.2.2 CrossEncoderModel Class

- **Architecture:** In this setup, both texts are concatenated and passed through BERT as a single sequence. This joint encoding enables the model to consider both texts in the same context, which is useful for tasks where understanding the relationship between the two is essential.
- **Classification Head:** The model produces class probabilities through a single, final linear layer, using cross-entropy for standard multi-class classification, focal loss for handling imbalanced datasets, and KL divergence for soft-label tasks.
- **Regression Head:** For continuous scoring, the model outputs a single value representing the empathy score. MSE and MAE were used as standard regression losses, while contrastive loss was also tested to encourage similarity for certain tasks.

Figure 9 provides a sample Python code Cross-Encoder class, showcasing its implementation with a regression head for processing input sentences. For more details, refer to the file `RegressorCrossEncoder.py` and `ClassifierCrossiEncoder.py` in Appendix C.

```python
class CrossEncoderModel(torch.nn.Module):
    def __init__(self, model_name, num_classes=1, loss_fn="mse"):
        super(CrossEncoderModel, self).__init__()
        self.model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=num_classes)
        self.loss_fn = loss_fn

    def forward(self, input_ids, attention_mask, labels=None):
        logits = self.model(input_ids=input_ids, attention_mask=attention_mask).logits
        loss = self.compute_loss(logits, labels) if labels is not None else None
        return {"loss": loss, "logits": logits}

    def compute_loss(self, logits, labels):
        # Custom loss function implementation
        pass
```

Figure 9: Overview of the Cross-Encoder Class

3.2.3 Loss Functions

Each architecture was tested with various loss functions tailored to classification and regression tasks to optimize model performance:

3.2.3.1 Classification Losses

- **Cross-Entropy Loss:** Standard choice for multi-class classification.
- **Focal Loss:** Adjusts loss to handle class imbalance by focusing on harder-to-predict samples.
- **KL Divergence:** Used for soft-label classification to compare predicted and actual label distributions.

3.2.3.2 Regression Losses

- **Mean Squared Error (MSE):** Commonly used for regression tasks, penalizing larger errors.
- **Mean Absolute Error (MAE):** Provides a robust alternative, focusing on absolute differences.
- **Contrastive Loss:** Encourages similarity between embeddings for similarity scoring.
- **Cosine Embedding Loss:** Evaluates similarity in embedding space; used only with the bi-encoder as it requires separate embeddings.

for more details about the loss functions, Please refer to Appendix B.

3.3 Fine-Tuning Process for Both Architectures

The fine-tuning process for the bi-encoder and cross-encoder involved a systematic approach to optimize model performance, focusing on appropriate loss functions, hyperparameters, and data management. Each model architecture was fine-tuned on the task-specific empathy datasets for both classification and regression, ensuring that the models were tuned for both discrete empathy classification and continuous scoring. Bellow is a list of all models trained in this project, along with links to their respective Hugging Face Hub repositories.

- Bi-Encoder with KL Divergence for Classification
- Bi-Encoder with Focal Loss for Classification
- Bi-Encoder with Cross-Entropy for Classification
- Bi-Encoder with MAE for Regression
- Bi-Encoder with MSE for Regression

- Bi-Encoder with Cosine Embedding Loss for Regression
- Bi-Encoder with Contrastive Loss for Regression
- Cross-Encoder with Contrastive Loss for Regression
- Cross-Encoder with MAE for Regression
- Cross-Encoder with MSE for Regression
- Cross-Encoder with Cross-Entropy for Classification
- Cross-Encoder with Focal Loss for Classification
- Cross-Encoder with KL Divergence for Classification

3.3.1 Preprocessing and Data Collation

For the bi-encoder, the texts were tokenized independently, generating separate embeddings for each. A custom data collator managed batching by aligning text pairs, allowing the model to encode each text independently before merging embeddings. For the cross-encoder, the texts were concatenated with a `[SEP]` token and tokenized as a single sequence. Here, a data collator created concatenated sequences of text pairs, ensuring consistency for BERT's input format.

3.3.2 Training and Hyperparameter Optimization

- **Hyperparameters:** Both architectures were fine-tuned with consistent hyperparameters, tracked and optimized through WandB. Key parameters included batch size, learning rate, epochs, and weight decay. I used TrainingArguments from Hugging Face to establish settings for regular logging, evaluation frequency, and checkpointing.
- **Loss Functions:** I tested various loss functions across both architectures to determine their impact on model performance. For classification tasks, cross-entropy, focal loss, and KL divergence were used, while regression tasks employed MSE, MAE, and cosine embedding loss. For the bi-encoder, cosine embedding and contrastive loss were particularly effective due to the separate encoding of texts.
- **Checkpointing and Best-Model Saving:** To ensure model robustness, checkpoints were saved throughout training, with the best-performing model retained for final evaluation and pushed to the Hugging Face Hub.

3.3.3 Model Evaluation and Metric Computation

- **Classification Metrics:** Accuracy, F1-score, precision, and recall were calculated for classification tasks. These metrics offered a comprehensive view of the model's performance, particularly when fine-tuning for empathy classification.
- **Regression Metrics:** MSE, MAE, Pearson correlation, Spearman correlation, and cosine similarity were used for regression tasks, providing a thorough analysis of the model's continuous empathy scores.

3.3.4 Saving and Loading Models

Custom save and load functions were implemented to handle both architectures effectively. This customization ensured compatibility with the Hugging Face Hub, allowing models to be shared and retrieved easily for further fine-tuning or testing. Please refer to file `Save_and_load.py` in Appendix C.

4 Evaluation and Results

For empathy detection, both classification and regression metrics were essential to capture different aspects of model performance.

4.1 Metrics

4.1.1 Classification Metrics

Since empathy detection requires distinguishing nuanced levels of empathy, F1 score and accuracy were crucial for assessing how well the model could identify these differences across categories. Precision and recall offered insights into the model's handling of positive empathy predictions, helping refine the model's tendency to predict empathy levels without over or under-emphasizing specific classes.

- **Accuracy:** Measures the percentage of correct predictions, providing an overall view of how well the model can correctly classify empathy levels.
- **F1 Score:** Balances precision and recall, especially valuable for tasks with class imbalance. The F1 score helped gauge the model's effectiveness in distinguishing between empathy levels by giving equal importance to precision and recall.
- **Precision and Recall:** Precision represents the proportion of correctly predicted positive instances out of all predicted positives, while recall captures the proportion of true positives identified out of all actual positives. These metrics provided insights into how well the model differentiated between empathy levels without overpredicting or underpredicting any particular class.

4.1.2 Regression Metrics

Given empathy detection often involves subtle, continuous variations, MSE and MAE captured how closely predictions matched actual empathy scores, with MSE emphasizing larger errors. Pearson and Spearman correlations allowed us to examine how well the model preserved empathy score rankings, important for applications where the order of empathy levels is critical. Cosine similarity provided a further measure of alignment in embedding space, useful for tasks where vector similarity reflects empathy detection performance.

- **Mean Squared Error (MSE):** Calculates the average squared differences between predicted and actual empathy scores. This metric emphasizes larger errors, penalizing predictions that deviate significantly from the true empathy scores, making it ideal for regression tasks.
- **Mean Absolute Error (MAE):** Computes the average of absolute differences between predicted and actual scores, providing a straightforward measure of prediction accuracy. MAE was particularly useful for understanding the model's ability to produce close estimates without being skewed by large errors.
- **Pearson Correlation:** Measures the linear relationship between predicted and actual empathy scores, helping assess how well the model's predictions align with the true scores in a linear fashion.
- **Spearman Correlation:** Captures the rank correlation, useful if the relationship between predicted and actual scores is non-linear. It provides insight into whether the model's predictions preserve the ranking order of empathy levels.
- **Cosine Similarity:** Measures the alignment between prediction vectors in embedding space, providing an alternative similarity measure between predicted and actual scores.

4.2 Results

4.2.1 Cross-Encoder Classification Performance: Comparative Analysis of Loss Functions

The cross-encoder model was evaluated for classification using three distinct loss functions: KL Divergence, Focal Loss, and Cross-Entropy Loss. For more information about the fine-tune process and logs, please refer to file `train_clf_CrossEncoder_Ds2.ipynb` and `predict_clf_CrossEncoder_Ds2.ipynb` in Appendix C. Below is a summary and comparative analysis of the model's performance under each loss function: The results suggest that Focal Loss outperforms the other two loss functions (KL

Loss Function	Accuracy	Precision	Recall	F1 Score
KL Divergence	0.584	0.603	0.584	0.587
Focal Loss	0.575	0.631	0.575	0.589
Cross-Entropy Loss	0.565	0.576	0.565	0.568

Table 1: Performance Metrics for Cross-Encoder Classification

Divergence and Cross-Entropy) in terms of precision and F1 score, which are crucial for evaluating classification performance on imbalanced classes. This indicates that Focal Loss is more effective for this empathy detection task, as it emphasizes difficult-to-predict classes, potentially mitigating class imbalance.

- KL Divergence achieved the highest accuracy (0.584) but had a slightly lower precision compared to Focal Loss.
- Cross-Entropy Loss, while commonly used, provided the lowest performance across all metrics, suggesting it may be less suitable for empathy recognition tasks where class distributions might be imbalanced or subtle contextual differences affect classification outcomes.

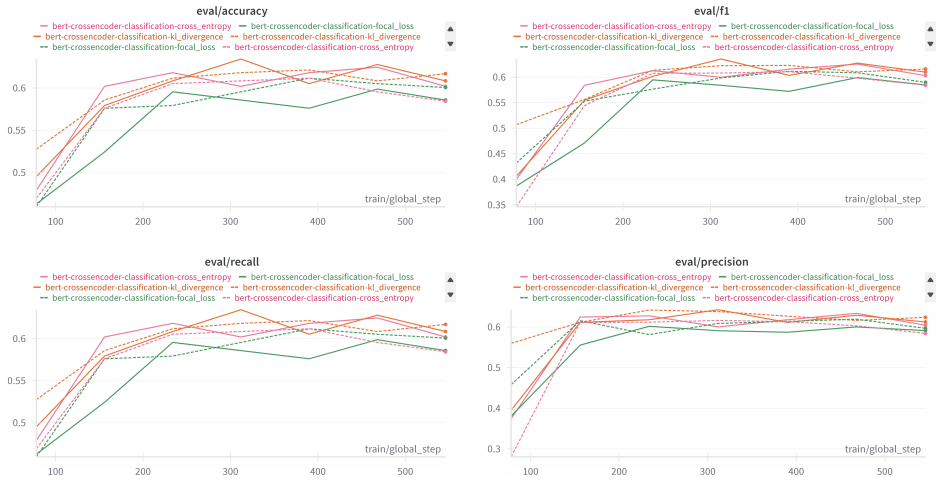

Figure 10: Classification Cross-Encoder model evaluation metrics (accuracy, F1 score, recall, precision) over training epochs per subject loss functions.

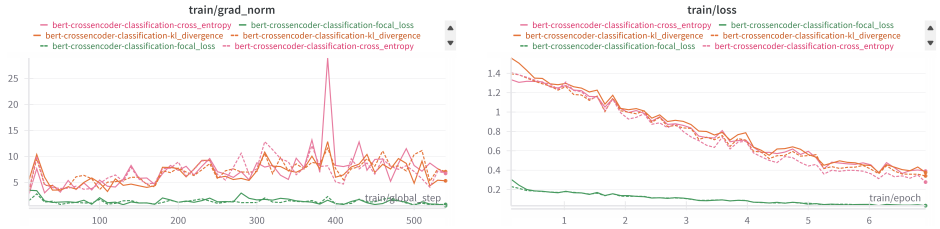

Figure 11: Training dynamics for the Cross-Encoder classification model over training epochs.

These findings highlight several implications for empathy detection:

- **Focal Loss's suitability:** Its higher precision and F1 score indicate it may be better suited for nuanced empathy classification tasks where classes are imbalanced.
- **Limitations of Cross-Entropy Loss:** The standard cross-entropy approach did not perform as well, implying that empathy detection may benefit from loss functions that can handle imbalanced and complex data distributions.

- **Potential of KL Divergence for Soft Classification:** Although KL Divergence did not achieve the highest precision, its relatively high accuracy could be valuable for scenarios that require probabilistic interpretation of class distributions.

4.2.2 Bi-Encoder Classification Performance: Comparative Analysis of Loss Functions

The bi-encoder model was assessed on classification tasks using Cross-Entropy Loss, Focal Loss, and KL Divergence. For more information about the fine-tune process and logs, please refer to file `train&predict_clf_BiEncoder_Ds2.ipynb` in Appendix C. Here's a summary of model performance:

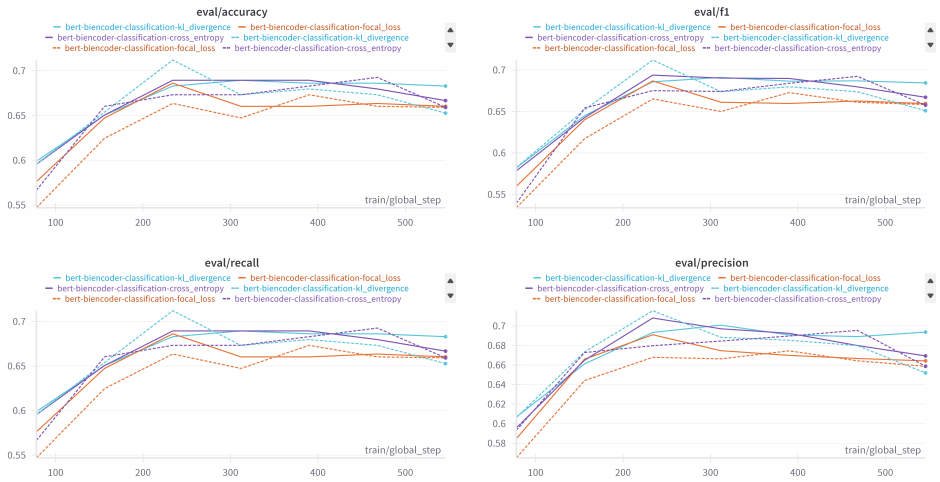

Figure 12: Classification Bi-Encoder model evaluation metrics (accuracy, F1 score, recall, precision) over training epochs per subject loss functions.

Loss Function	Accuracy	F1 Score	Precision	Recall
Cross-Entropy Loss	0.643	0.643	0.650	0.643
Focal Loss	0.653	0.651	0.651	0.653
KL Divergence	0.643	0.643	0.645	0.643

Table 2: Performance Metrics for Bi-Encoder Classification

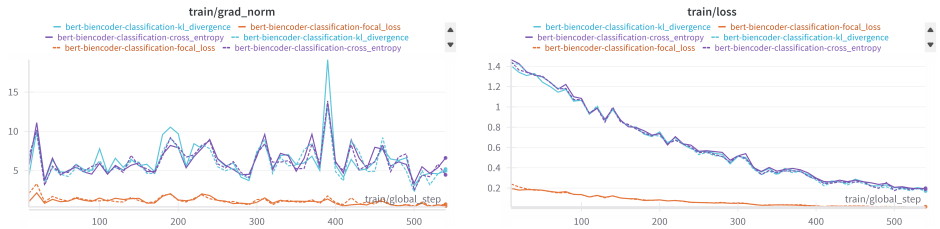

Figure 13: Training dynamics for the Bi-Encoder classification model over training epochs.

- **Focal Loss** provided the highest performance across all metrics, especially with a test accuracy of 0.653 and F1 score of 0.651, reinforcing its effectiveness in managing class imbalance.
- **Cross-Entropy Loss** and **KL Divergence** produced similar results, but both slightly underperformed compared to Focal Loss.

Bi-Encoder with Focal Loss achieved the best classification results overall. Cross-encoders generally performed similarly but didn't surpass bi-encoders in accuracy and F1 scores, indicating that handling

empathy might benefit from independent encoding with bi-encoders. This analysis suggests that Focal Loss is advantageous for empathy classification, particularly with the bi-encoder model.

4.2.3 Cross-Encoder Regression Performance: Comparative Analysis of Loss Functions

For regression tasks using the cross-encoder architecture, I tested multiple loss functions to evaluate their effectiveness for empathy rating prediction. For more information about the fine-tune process and logs, please refer to file `train_reg_CrossEncoder_Ds1.ipynb` and `predict_reg_CrossEncoder_Ds1.ipynb` in Appendix C. Here are the performance metrics of each model across Mean Squared Error (MSE), Mean Absolute Error (MAE), Pearson and Spearman correlations, and Cosine Similarity.

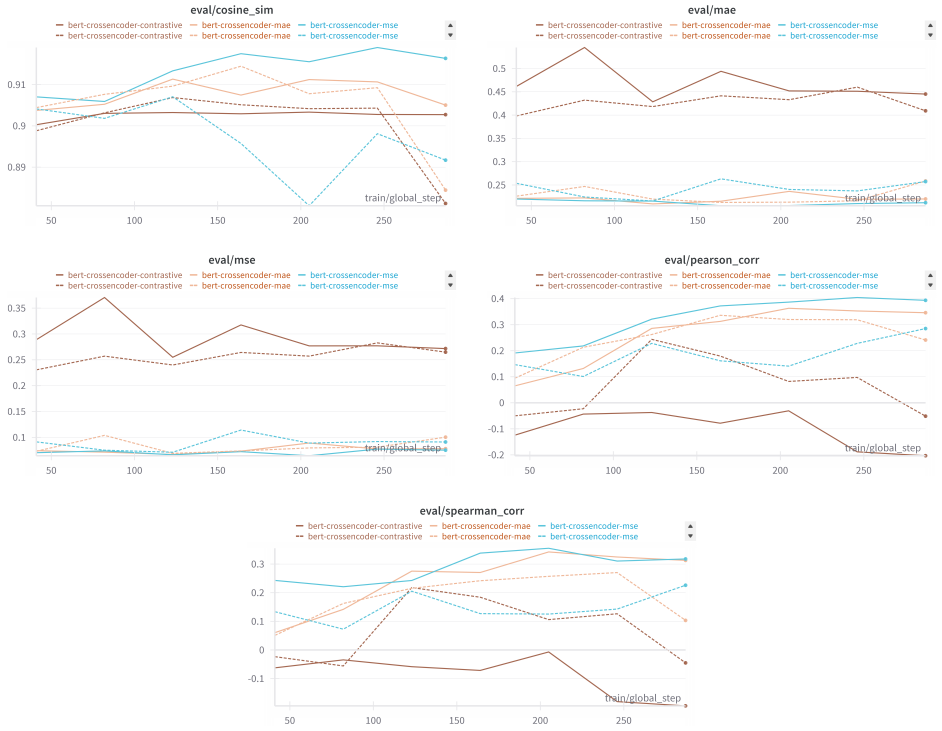

Figure 14: Regression Cross-Encoder model evaluation metrics (cosine similarity, mean absolute error (MAE), mean squared error (MSE), Pearson correlation, and Spearman correlation) over training epochs per subject loss functions.

Loss Function	MSE	MAE	Pearson r.	Spearman r.	Cosine Similarity
MSE	0.1038	0.2748	0.2103	0.1444	0.8772
MAE Loss	0.0934	0.2642	0.2800	0.2651	0.8915
Contrastive Loss	0.2631	0.4095	0.0298	0.0073	0.8820

Table 3: Performance Metrics for Cross-Encoder Regression

- **MAE Loss** provided the best overall performance across MSE, MAE, and correlation metrics, with notably higher Pearson and Spearman correlations and the highest Cosine Similarity (0.8915). This suggests that MAE Loss is effective in handling continuous empathy rating prediction for the cross-encoder setup.
- **MSE Loss** also performed well but fell short of the MAE loss in correlation metrics and Cosine Similarity.
- **Contrastive Loss** showed significantly lower performance, with higher MSE and MAE values and low correlation scores, indicating that it may not be suitable for continuous empathy scoring with a cross-encoder architecture.

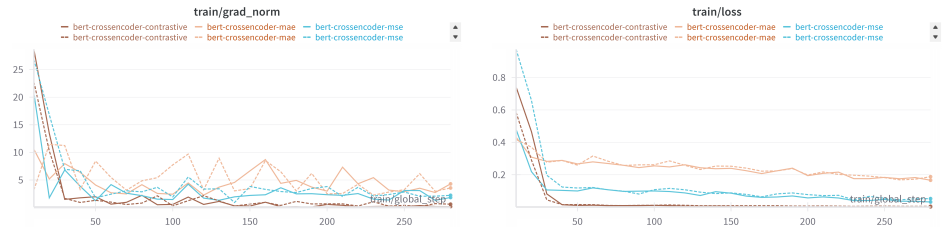

Figure 15: Training dynamics for the Cross-Encoder regression model over training epochs.

4.2.4 Bi-Encoder Regression Performance: Comparative Analysis of Loss Functionss

This section Provides summary of the fine-tune process and evaluation metrics. For more information about the fine-tune process and logs, please refer to file `train&predict_reg_BiEncoder_Ds1.ipynb` in Appendix C. The bi-encoder architecture was evaluated on various regression loss functions to

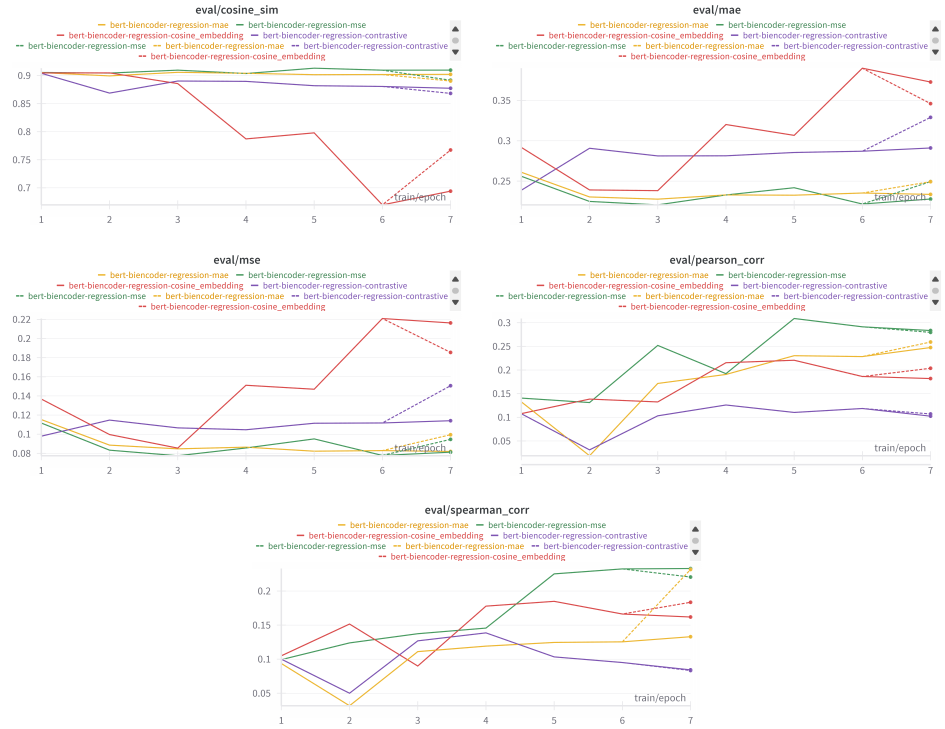

Figure 16: Regression Bi-Encoder model evaluation metrics (cosine similarity, mean absolute error (MAE), mean squared error (MSE), Pearson correlation, and Spearman correlation) over training epochs per subject loss functions.

determine its effectiveness in empathy rating prediction. Here are the key performance metrics for each model, comparing Mean Squared Error (MSE), Mean Absolute Error (MAE), Pearson and Spearman correlations, and Cosine Similarity.

- **MSE Loss** achieved the best results overall, with the lowest MSE and highest Cosine Similarity (0.8917), making it the most suitable choice for continuous empathy prediction with a bi-encoder architecture.
- **MAE Loss** also performed well but had slightly lower Pearson and Spearman correlations than MSE Loss.

Loss Function	MSE	MAE	Pearson r.	Spearman r.	Cosine Similarity
MSE Loss	0.0946	0.2494	0.2796	0.2205	0.8917
MAE Loss	0.0994	0.2493	0.2592	0.2317	0.8903
Contrastive Loss	0.1507	0.3291	0.1071	0.0833	0.8683
Cosine Embedding Loss	0.1855	0.3461	0.2037	0.1835	0.7674

Table 4: Performance Metrics for Bi-Encoder Regression

Figure 17: Training dynamics for the Bi-Encoder regression model over training epochs.

- **Contrastive Loss** and **Cosine Embedding Loss** underperformed in comparison, showing higher MSE and MAE values and lower correlation scores. These results indicate that these loss functions may not be optimal for this task in the bi-encoder setup.

MSE Loss with the Bi-Encoder architecture appears to be the most effective for regression tasks in empathy rating prediction. The final section will provide a comparative analysis of the bi-encoder and cross-encoder architectures to determine the overall best configuration.

5 Discussion and Findings

5.1 Architecture Comparison

The bi-encoder and cross-encoder architectures were both tested on empathy recognition for classification and regression tasks.

5.1.1 Performance Metrics

For classification, bi-encoder with focal loss outperformed cross-encoder configurations, achieving the highest accuracy (65.26%) and F1-score, indicating that bi-encoder was more capable in discrete empathy level prediction. In regression, both architectures showed varied results across different loss functions. However, bi-encoder with MSE loss provided the lowest MSE (0.0946) and the highest cosine similarity, suggesting it effectively captures the continuous empathy scale in regression.

5.1.2 Efficiency

The runtime and resources used for training and testing models were not recorded or analyzed in this study; however, working with both architectures provided insights into these aspects. Bi-encoder models generally demand more processing power and memory due to their separate encoding of text pairs, which can result in slower processing times compared to cross-encoder models. On the other hand, Cross-encoder models are more memory-efficient and faster, processing both texts jointly, making them preferable in scenarios where computational resources are limited. Bi-encoder outperforms cross-encoder in predictive accuracy, particularly for classification, while cross-encoder may be more efficient in terms of speed and resource usage, suiting environments with limited resources.

5.2 Loss Function Analysis

Testing various loss functions across both architectures allowed us to understand how they impact empathy detection differently.

5.2.1 Classification Losses

Focal Loss consistently produced the best classification results in both architectures. It effectively managed class imbalance, improving precision and recall, which are crucial for empathy recognition with imbalanced data. Cross-Entropy Loss, though standard for classification, did not achieve the same level of accuracy or F1 scores, indicating that focal loss's additional weighting on harder-to-predict samples benefited this specific task. KL Divergence was less effective in both architectures, likely due to its focus on soft-label distributions rather than discrete class distinctions, which may not align well with empathy classification.

5.2.2 Regression Losses

MSE Loss provided the best results in both architectures, yielding the lowest MSE values and highest correlations. This loss function's penalty on larger errors aligns well with the empathy detection task's need for precise predictions on a continuous scale. MAE Loss also performed well, but with slightly lower correlation scores than MSE, suggesting it's a viable but secondary option. Contrastive Loss and Cosine Embedding Loss did not perform as effectively, likely because their focus on similarity doesn't directly capture empathy as a continuous, nuanced score.

In summary, focal loss was the best choice for classification, and MSE loss proved most suitable for regression.

5.3 Best Model Identification

Performance analysis indicates that the top-performing models for empathy detection are the bi-encoder with focal loss for classification and the bi-encoder with MSE loss for regression. The classification model achieved the highest accuracy (65.26%) and maintained a strong balance between precision and recall, making it particularly effective for predicting discrete empathy levels. For regression, the bi-encoder with MSE loss delivered the lowest mean squared error and the highest cosine similarity, positioning it as ideal for capturing empathy on a nuanced, continuous scale. These models excel due to the synergy between the bi-encoder architecture, which encodes dual-text representations, and the alignment of focal and MSE losses with the specific demands of their tasks.

5.4 Regression vs. Classification

The comparison of regression and classification approaches reveals distinct strengths and limitations in empathy detection. Classification provides the advantage of discrete levels, which are easier to interpret and validate, making it simpler to identify high and low empathy cases. However, empathy as a continuous experience may not always fit neatly into categorical boundaries, potentially missing subtle variations. In contrast, regression allows for a more nuanced approach, capturing finer details in empathy expression that might be overlooked by classification. Despite this, interpreting regression results can be challenging, as minor differences in scores may lack meaningful significance. Both methods offer value depending on the application: classification may be more effective for user-facing scenarios where interpretability is essential, while regression provides a richer representation for detailed analysis or scoring.

6 Conclusions and Recommendations for Future Work

The development of a unified empathy rater remains out of reach due to the lack of comprehensive and standardized empathy datasets, as noted both in this study and in existing literature. This project aimed to explore and compare architectures and loss functions to inform future efforts in empathy detection, identifying effective methods that could eventually guide the creation of a robust empathy rating model.

My evaluation focused on Bi-Encoder and Cross-Encoder architectures, each showing distinct strengths. Bi-Encoders, though more resource-intensive, excelled in regression tasks, capturing empathy on a continuous scale. Cross-Encoders, by contrast, demonstrated efficiency in classification tasks, making them suitable for less complex empathy assessments. These findings suggest that while Bi-Encoders may be costly, they are valuable for tasks requiring detailed empathy understanding. For loss

functions, Mean Absolute Error (MAE) proved effective for regression-based empathy scoring, while Cross-Entropy worked best for classification, underscoring that empathy may be better represented through continuous rather than categorical measures. This project suggests that while both classification and regression approaches offer potential, targeted dataset improvements and methodological refinements will be key to achieving a robust empathy rating model.

Progress in empathy-focused datasets is essential. I recommend a balanced annotation approach, incorporating both continuous and discrete labels to support a wider range of modeling strategies. This study underscores that the right dataset structure and loss functions are foundational to progress in empathy modeling.

The findings here provide several recommendations for future work in empathy data collection and model design:

- **Dataset Expansion and Annotation:** The field of empathy detection requires larger, well-annotated datasets. The nuanced nature of empathy suggests that continuous scoring may capture the range of empathic expression more effectively. Future datasets could benefit from a mixed annotation approach, combining level-based labels for interpretability and regression scores for depth.

- **Refining Models for Empathy:** As empathy differs from traditional NLP tasks like sentiment analysis, models trained on mixed empathy-related tasks, potentially using bi-encoder architectures, may enhance performance in future applications. NLP tasks that involve nuanced relationship understanding, such as entailment, could provide valuable techniques for empathy modeling.

- **Guidelines for New Datasets:** To advance toward a unified empathy rater, future datasets should prioritize a clear empathy definition, consistent scoring criteria, and a balance of level-based and continuous annotations to best capture empathy's complexity.

A Unsuccessful Attempts

A.1 Use Sentence Transformers for Training

I initially tried the Sentence Transformers library for its support of both bi-encoder and cross-encoder architectures. However, limitations in custom loss functions, logging, and task-specific adaptations led me to develop custom model classes and training functions, providing full control over architecture and loss function selection. For details on the failed attempts refer to Figure 18 - 20.

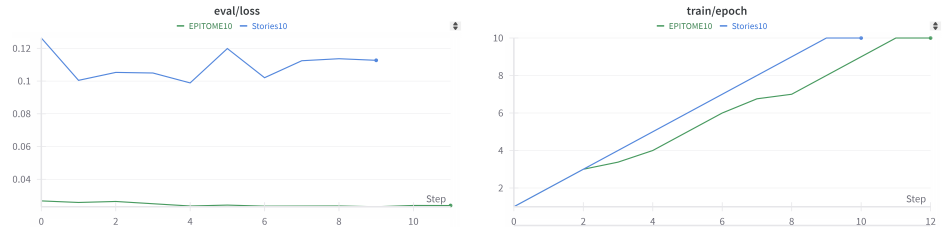

Figure 18: Evaluation and training metrics for initial unsuccessful model attempt, showing instability and lack of convergence.

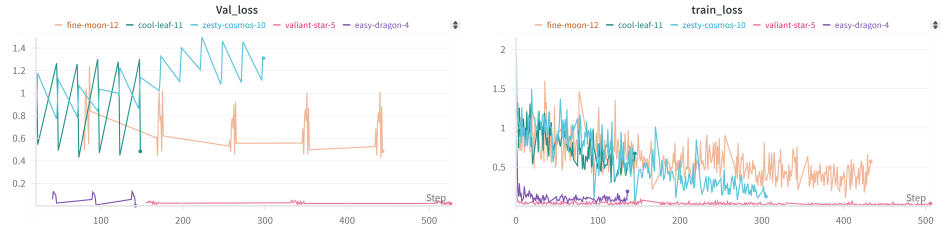

Figure 19: Second unsuccessful training attempt with Sentence Transformers, highlighting issues with logging and custom loss implementation.

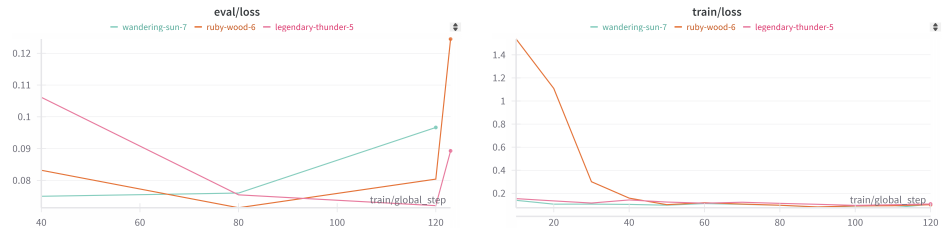

Figure 20: Third failed attempt due to limitations in task-specific adaptations, leading to unstable evaluation metrics.

A.2 EPITOME Dataset with Regression Labels for Empathy Scoring

Initially, the focus of this project was solely on regression for empathy prediction, using the two previously mentioned datasets. However, upon observing substantial differences in the behavior of the loss function values between these datasets, it became clear that treating the EPITOME dataset as a classification problem would better align with the nature and origin of its labels. Consequently, I defined a classification task for the EPITOME dataset and trained both bi-encoder and cross-encoder models accordingly. For additional details on this decision and the comparison of learning curves regression models among the two datasets, please refer to Figure 22 and 23.

	seeker_post	response_post	level
0	Help. Help me. I dunno what I'm doing anymore	That's pretty vague, do you not know what you'...	0.333333
1	I'm done saying I love you to her because I do...	idk what a Red pill means exactly but my advic...	0.000000
2	Always feel like I'm being criticized and mock...	I think it's social anxiety , that creates par...	0.333333

Figure 21: EPITOME data with continuous 0-1 labels.

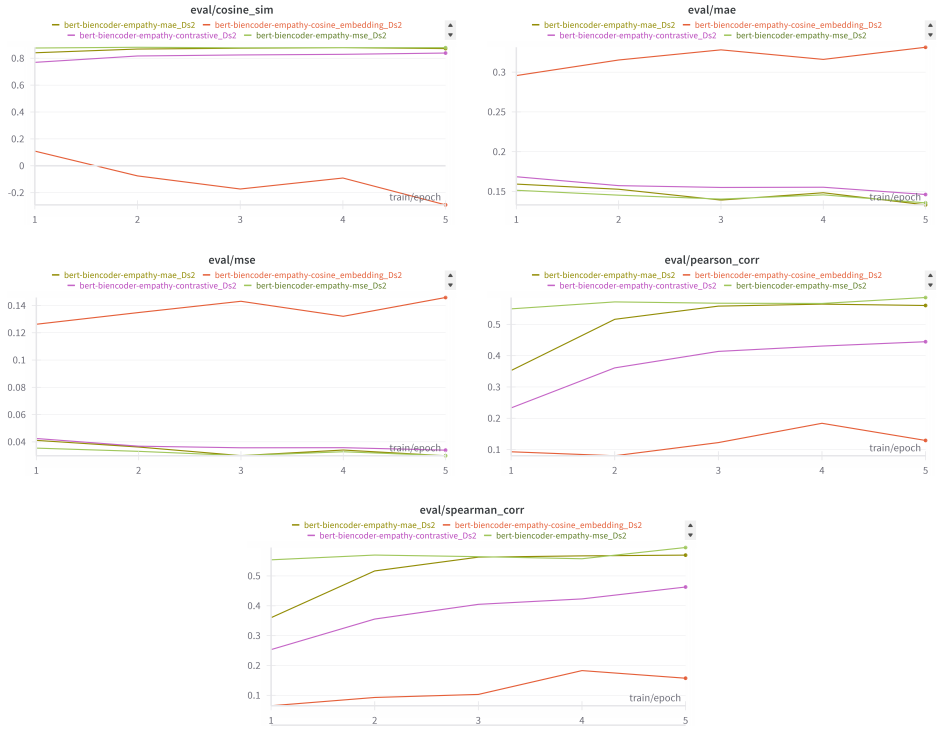

Figure 22: EP dataset in regression settings: Regression Bi-Encoder model evaluation metrics (cosine similarity, mean absolute error (MAE), mean squared error (MSE), Pearson correlation, and Spearman correlation) over training epochs per subject loss functions.

A.3 Save/Load Issues

Initial training rounds faced issues with Hugging Face's save/load functions for custom models, leading to data loss and re-training. I implemented custom save/load functions for seamless model versioning with the Hugging Face Hub, ensuring consistent model retrieval. In all the Wandb graphs, there are 2 logs per models with the same name and color, one indicating the log for the unsaved model and the other for the final successfuly saved model.

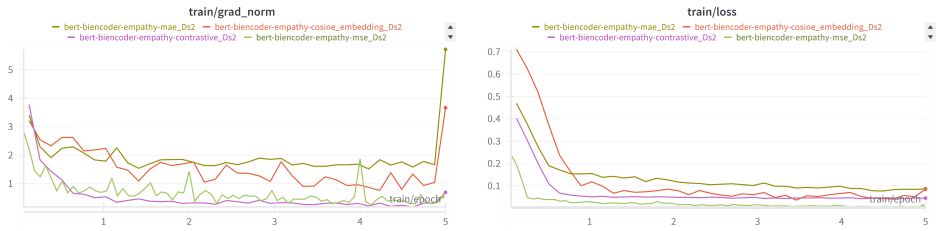

Figure 23: EP dataset in regression settings: Training dynamics for the Bi-Encoder regression model over training epochs.

B Loss Functions for Sentence Encoders

Tables 6 and 5 show more detailed information about the loss functions that have been used in this project, including their use-cases and mathematical definitions.

Loss Function	**Formula**
Cross-Entropy Loss: Standard loss for multi-class classification, comparing predicted probabilities with actual labels.	$$L_{\text{Cross-Entropy}} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$ where y_i is the true label, $\hat{y}_i$ is the predicted probability for class i, and C is the number of classes.
Focal Loss: Designed for class imbalance by down-weighting easy examples and focusing on hard ones.	$$L_{\text{Focal}} = -\alpha(1-\hat{y})^{\gamma}\log(\hat{y})$$ where α is a scaling factor, γ controls focus on hard examples, and $\hat{y}$ is the predicted probability for the true class.
KL Divergence: Measures the difference between two probability distributions.	$$L_{\text{KL}} = \sum_{i=1}^{C} y_i \log\left(\frac{y_i}{\hat{y}_i}\right)$$ where y_i is the actual probability and $\hat{y}_i$ the predicted probability for class i.

Table 5: Classification Loss Functions for Empathy Detection

Loss Function	Formula
Mean Squared Error: Calculates average squared differences between predicted and actual values.	$$L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$ where y_i is the actual value, $\hat{y}_i$ the predicted value, and N is the number of samples.
Mean Absolute Error: Calculates average absolute differences between predicted and actual values.	$$L_{\text{MAE}} = \frac{1}{N}\sum_{i=1}^{N}\lvert y_i - \hat{y}_i\rvert$$ where y_i is the actual value and $\hat{y}_i$ the predicted value.
Contrastive Loss: Minimizes distance between similar embeddings and maximizes distance between dissimilar ones.	$$L_{\text{Contrastive}} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i \cdot D^2 + (1-y_i)\cdot \max(0, m-D)^2\right)$$ where y_i indicates pair similarity, D is distance, and m is the margin.
Cosine Embedding Loss: Evaluates similarity between embeddings, maximizing cosine similarity for similar texts.	$$L_{\text{Cosine}} = \frac{1}{N}\sum_{i=1}^{N}(1 - y_i \cdot \cos(\theta))$$ where y_i is the target similarity label, and $\cos(\theta)$ is the cosine similarity.

Table 6: Regression Loss Functions for Empathy Detection

C Supplementary Files for This Project

All files referred to in the text are accessible inside `Empathy Project`. The following is the comprehensive list of files and directories associated with this project:

```
Empathy Project
├── Project_Plan_for_Directed_Studies___Fall_2024.pdf
├── Preprocess
│   ├── Annotated_story_pairs1.csv
│   ├── Annotated_story_pairs2.csv
│   ├── DataPreparation.ipynb
│   ├── DatasetHF.py
│   ├── EPITOME_pairs.csv
│   ├── EPITOME_pairs1.csv
│   └── EPITOME_pairs2.csv
├── train&predict_runs
│   ├── predict_clf_CrossEncoder_Ds2.ipynb
│   ├── predict_reg_CrossEncoder_Ds1.ipynb
│   ├── train&predict_clf_BiEncoder_Ds2.ipynb
│   ├── train&predict_reg_BiEncoder_Ds1.ipynb
│   ├── train_clf_CrossEncoder_Ds2.ipynb
│   └── train_reg_CrossEncoder_Ds1.ipynb
└── unsuccessful_attempts
    └── Fine_tune1.ipynb
```

```
│
│   ├── Fine_tune2.ipynb
│   ├── notebookd560e1c233.ipynb
│   ├── tuneBERTasCE.ipynb
│   ├── tuneBERTasCE2.ipynb
│   ├── Untitled5.ipynb
│   ├── Untitled7.ipynb
│   ├── ThisQ.ipynb
│   ├── ThisQ1.ipynb
│   ├── predict_clf_biEncoder_Ds2.ipynb
│   ├── notebook9b9ccb78ff.ipynb
│   ├── kgglenotebook7caa1896ef.ipynb
│   ├── colabpredict_clf_BiEncoder_Ds2.ipynb
│   ├── successful_training_saving_failed
│   │   ├── train_BiEncoder_Ds1.ipynb
│   │   ├── train_BiEncoder_Ds2(part1).ipynb
│   │   ├── train_BiEncoder_Ds2(part2).ipynb
│   │   ├── train_clf_BiEncoder_Ds2.ipynb
│   │   ├── train_clf_CrossEncoder_Ds2.ipynb
│   │   └── train_CrossEncoder_Ds1.ipynb
└── utils
    ├── ClassifierBiEncoder.py
    ├── ClassifierCrossEncoder.py
    ├── MyCEmetrics.py
    ├── RegressorBiEncoder.py
    ├── RegressorCrossEncoder.py
    └── Save_and_load.py
```