wangkongqiang at MentalRiskES@IberLEF 2025: Early Detection of Mental Disorders Risk in Spanish

Kongqiang Wang*

[1]*Yunnan University, School of Information Science and Engineering, Kunming, Yunnan, 650500, China*

Abstract

According to a recent report by the World Health Organization, there is 1 in every 8 people in the world suffering from a mental disorder. The organisation MentalRiskES at IberLEF 2025 considers that early identification is a key effective intervention to prevent these problems. The task I participated in was Task 1: Risk Detection of Gambling Disorders. This is a binary classification task aimed at determining whether a user is at high risk (label = 1) or low risk (label = 0) of developing a gambling-related disorder based on their messages. The objective is to enable early detection and facilitate timely interventions. We compare the performance of two different modeling approaches: fine-tuning a roberta-base model and using sentence embeddings as inputs to a linear regressor, with the latter yielding better results. My final experimental result is Accuracy 0.519, Macro_R 0.500, ERDE_5 0.332, ERDE_30 0.250.

Keywords

Mental Health, Natural Language Processing, Transformers, Sentence Embedding

1. Introduction

Mental health is a growing concern in our society[1]. According to the World Health Organization (WHO), 1 in 4 people will be affected by mental disorders at some point in their lives. In addition, the COVID-19 pandemic[2] has had a negative impact on the mental health of the general population, with an increase in the number of people suffering from mental disorders. Thus, it is becoming increasingly important to evaluate the use of new technologies to assess the risk of mental illness and the healthcare needs of the population.

At the same time, social media platforms such as Telegram have become a popular way for people to express their feeling and emotions. Telegram is a free, end-to-end encrypted messaging service that allows users to send and receive messages and media files in private chats or groups that can be focused on particular topics and allow any user to observe or actively participate. These characteristics make Telegram a suitable source for text-mining.

With this context, an interesting approach is to use Natural Language Processing (NLP) techniques to analyze the language used by people who suffer from mental illness and discover patterns that can be used to identify them and provide the necessary support. The MentalRiskES task at IberLEF 2025 [3] aims to promote the development of NLP solutions specifically for Spanish-speaking social media. They propose two main areas of focus for early-risk detection: Risk Detection of Gambling Disorders (Task 1), Type of Addiction Detection (Task 2).

In this work, we present our proposed solution to Task 1 of the 2025 edition of MentalRiskES[4]. This task involves evaluating the likelihood of a Telegram user experiencing Gambling Disorders based on their comments within mental-health-focused groups. The task is split into two predictive labels (at high risk (label = 1) or low risk (label = 0)) according to the type of output required. Our main contributions and findings can be then summarized fivefold:

- `concat_messages` for message processing.

IberLEF 2025 September 2025, Zaragoza, Spain

*Corresponding author.

wangkongqiang60@gmail.com (K. Wang)

https://github.com/WangKongQiang/ (K. Wang)

The first thing we did was group all the messages by the user they belonged to and concatenated them into a single string, obtaining a total of 357 messages (one per user). This was done to obtain a single representation of each user's conversation history (from which the labels were assigned) to be able to use it as input for the models.

- `augment the train data` for train dataset settings.
  To increase the amount of data available for training, at the same time, attempt to model early detection (obtaining predictions early on in the lifetime of the message history), we augmented the training set by adding observations that only contained half of their messages (The first half and the second half), One third of their messages (The first third of the data, the middle third of the data, and the last third of the data). This was done by first sorting the messages of each user in the training set by its date and then only taking the augment data, the resulting dataset was then appended to the original training set to obtain a new one with sixfold the number of observations to be used for training.
- `pretrain model` for model training.
  We used two models, and they are respectively models–PlanTL-GOB-ES–roberta-base-bne[5] and models–somosnlp-hackathon-2023–roberta-base-bne-finetuned-suicide-es[6]. The model we fine-tuned was a version of RoBERTa pre-trained for detecting suicidal behavior from texts in Spanish. We chose this model due to the fact of having been trained previously for a task that shares similar characteristics to ours.
- `embedding model` optional package if regression estimators is required in the document.
  The sentence embeddings were obtained after concatenating the messages of each user into a single string. The difference in performance between roberta-base-bne-suicide-es encodings and the other embeddings can be justified by the fact that we are taking advantage of the information gained from the prior fine-tuning for suicide detection of this model, which likely shares semantic similarities with our data.
- `regression estimators` optional sklearn package for using sentence embeddings as inputs to a linear regressor.
  The estimators mentioned in the paper are implementations of common regressors from Python's Scikit-Learn library.These include: Ordinary (" lr ") and Ridge (" ridge ") [7], Least Squares Regression, Ada-Boost regression (" ada "), LightGradientBoosting Machine (" lgbm "), SupportVectorRegression (" svr "), RandomForests (" rf ") [8], and a Multi-Layer Perceptron (" mlp "). References of the implementations of these models can be found in the Scikit-Learn documentation.

The rest of the paper is organized as follows: In the next section, we analyze the dataset used for the task (Section 2). Then, we describe in detail our methodology for training and evaluating the models (Section 3). Finally, we discuss the results obtained (Section 4) and present our conclusions and future lines of work (Section 5).

2. Dataset Analysis

The dataset given for the task consisted of a total of thousands of individual messages from 357 Telegram / Twitch / Reddit users (see Table 1), each with a variable number of messages. The annotation process consisted of labeling each user based on the evidence from their conversation history of suffering from gambling disorders. Thus, a total of 2 labels were used for the tasks. Each was asked to assign one of the following two labels: at high risk (label = 1) or low risk (label = 0) to each user.

To increase the amount of data available for training, at the same time, attempt to model early detection (obtaining predictions early on in the lifetime of the message history), we augmented the training set by adding observations that only contained part of their messages. This was done by first sorting the messages of each user in the training set by its date and then only taking the part of data, the resulting dataset was then appended to the original training set to obtain a new one with sixfold the number of observations to be used for training. Now, taking the sample with the subject-id of user1002

Table 1
The official data set is provided

Dataset Type	Telegram users	Comments
trial dataset	7	Telegram/Twitch/Reddit platform
train dataset	350	Telegram/Twitch/Reddit platform
trial and train dataset	357	Used in training model
test dataset	136	Used in testing model

Table 2
Take the sample with user1002 as the subject-id as an instance

index	subject-id	message	round	id-message	date	Risk	num-messages	Comments
0	user1002	["|por dos| |JAJAJAJAJ| |yo ni sabia de los c...	-1	15470730081	2022-08-19 06:56:09+01:00	1.0	125.0	original data instance
1	user1002	["|por dos| |JAJAJAJAJ| |yo ni sabia de los c...	-1	15470730081	2022-08-19 06:56:09+01:00	1.0	62.0	The first half original data instance
2	user1002	Ya cierra el puto hocico| |PENDEJO DE MIERDA|:.	-1	15470730081	2022-08-19 06:56:09+01:00	1.0	63.0	The second half original data instance
3	user1002	["|por dos| |JAJAJAJAJ| |yo ni sabia de los c...	-1	15470730081	2022-08-19 06:56:09+01:00	1.0	41.0	The first third of the original data instance
4	user1002	CIERRA EL HOCICO| |ya puto arjensimio| |jajaja...	-1	15470730081	2022-08-19 06:56:09+01:00	1.0	41.0	The middle third of the original data instance
5	user1002	|cuando siempre sus partidos quedan 0-0| |J...	-1	15470730081	2022-08-19 06:56:09+01:00	1.0	42.0	The last third of the original data instance

as an instance, how to create an instance with five more partial data than the original data (see Table 2).

3. Methodology

We proceeded to evaluate different techniques to solve this subtasks. Two main predictive-modeling approaches were explored: The first one involved fine-tuning a pre-trained language model on this subtask and the second was about training a standard ML regressor using sentence embeddings encoded from the user's messages as features. The following section describes the steps taken for each approach, first describing how the data was pre-processed and later explaining the training and evaluation process done for this subtask.

3.1. Data Processing and Augmentation

Based on the detailed description and practical examples provided earlier in the article, a brief account will be given here, include:

- `concat_messages` : According to the messages of the telegram users, they are spliced in chronological order, that is, the rounds sequence. The standard for splicing is based on their user ids.
- `augment_data` : Expand based on half and one-third of the original data set, and eventually expand to six times the original size (including the original data). [1]

To prepare for training, the original data was split into training and validation sets, leaving a random 54 (15%) users in the latter for stratified cross-validation, where each set receives the same proportion of samples of each class. The stratification was done using the labels of Subtask 1 to ensure equal representation of the classes in both sets.

3.2. Solving Substask1 by Solving for Regression

By the discussion in Section 2, it should be clear to see that all labels of the subtasks give the same amount of information about the condition of the subject and the likelihood of predicting it based on the available data. This observation led us to consider using models that solve for this one subtask by

[1]We can enable the display and use of data only the official data set is provided and no additional extended datasets were used.

Table 3
Pre-trained BERT-based models used in our experiments.

Model	Description
models–PlanTL-GOB-ES–roberta-base-bne	RoBERTa model trained with data from Spain's National Library.
models–somosnlp-hackathon-2023–roberta-base-bne-finetuned-suicide-es	RoBERTa-base-bne fine-tuned for suicide detection.

only training it with the labels of Subtask1. This allowed us to reduce the number of models that had to be trained and focus on solving for a single data modality (regression on [0, 1]).

We approached simple regression in a standard way training models, training models to minimize the Mean Squared Error between the output values and the real ones. Additionally,we included the post-processing step of clipping the output predictions of models of this type to the [0, 1] range to ensure that they were valid probabilities. simple-output regression using standard machine learning regression, on the other hand,wasn't as trivial as in the simple binary classification case. We used many regressors in the sklean python package.

3.3. Modeling Approaches

3.3.1. Training a Regressor with Sentence Embeddings

A sentence embedding[9] is a semantically meaningful real-valued vector representation of a sentence, obtained from the outputs of the hidden layers of a language model. The properties of this representation are so that sentences that express similar meanings are mapped (encoded) closer to each other in the vector space.

In this way, the process of encoding text as numeric vectors can be used directly to extract features for a classifier or regressor, which will try to learn from the semantic information of these encodings to predict the label of their corresponding messages. Note, however, that this approach requires the need to have a pre-trained model to perform this encoding. Furthermore, it assumes that the model will be good enough at capturing the semantic information of the texts given as input, enough for the classifier / regressor to learn from it.

Assuming that this is the case, this approach has the advantage that it is much faster to train these kinds of regressors with regular CPUs, with the most time-consuming part being obtaining the embeddings of the training / evaluation messages, which only has to be done once. However, it is necessary to evaluate different encoding models and different classifiers / regressors (prediction models) to find the best combination for the task at hand.

As such, we conducted experiments using different language models to find the best encoding model. Particularly, we tested two different versions of roberta trained with different corpora in Spanish. These versions are described in Table 3. Additionally, we experimented with over 10 different regressors, including Least Squares Linear regression, Random Forest,and Gradient Boosting[10], among others. These models were chosen due to their ease of implementation and the fact that they are commonly used in the literature[11].

The process of training and evaluating these models proceeded then as follows: First, the training set was encoded using the language model and the resulting embeddings were used as features for a regressor. The regressor was then trained using the labels of Subtask1 (the most informative ones) and the resulting model was used to predict the labels of the validation set. The predictions were then evaluated with the root mean squared error (RMSE). This process was repeated for each combination of language model and regressor.

Appendix contains the results of this experiment. Based on that, roberta-suicide-es was deemed to be the best model for encoding the texts. Additionally, Table 8 shows a detailed report of the evaluation of the best regression model with these embeddings.

Table 4
Hyperparameters for fine-tuning a RoBERTa model for regression tasks.

Hyperparameters	Value
Optimizer	AdamW
Learning rate	1e-5
Max Tokens	1024
Num Epochs	30
Batch Size	1

Table 5
Official results from various models on a test set.

model	Accuracy	Macro_P	Macro_R	Macro_F1	Micro_P	Micro_R	Micro_F1	ERDE5	ERDE30
$baseline_1$	0.550	0.657	0.533	0.428	0.550	0.550	0.550	0.329	0.252
run_0	0.520	0.261	0.501	0.343	0.520	0.520	0.520	0.332	0.250
run_1	0.519	0.259	0.500	0.342	0.519	0.519	0.519	0.333	0.250
run_2	0.518	0.257	0.500	0.340	0.517	0.517	0.520	0.350	0.250
$baseline_2$	0.519	0.259	0.500	0.342	0.519	0.519	0.519	0.280	0.256

3.4. Fine-tuning a Language Model for Regression

Apart from the approach mentioned above, we also experimented with the pure Deep Learning (DL) [12] approach of taking a language model and fine-tuning it with the labels of the corresponding subtask. The model we fine-tuned was a version of RoBERTa pre-trained for detecting suicidal behavior from texts in Spanish. We chose this model due to the fact of having been trained previously for a task that shares similar characteristics to ours. Intermediate fine-tuning has been proven to improve the results of downstream tasks by prior literature. The HuggingFace Transformers and Pytorch libraries in Python were utilized for loading the model weights and implementing the training loop. We changed the head of the pre-trained model to a linear layer consisting of output dimension 1 for simple regression. The models were trained using an NVIDIA GeForce RTX 3090 24G GPU for a total of 30 epochs, where the weights of the pre-trained model remained fully frozen for the first half and then were progressively unfrozen[13] each epoch after that as in (see Table 4).

We used an Adam Optimizer with Mean-Squared Error (MSE) for the simple regression models. However, this did not improve the results empirically as compared with simply normalizing he outputs of the predictions after inference. The formula of this loss is shown in (LOSS_custom = LOSS_crossentropy). Other hyperparameters are shown in Table 4.

4. Results

Using the approaches mentioned in the prior section, we came up with different models to solve the subtask of Task 1 of MentalRiskES. The results in this section are obtained from selecting the best-performing models after evaluating the different approaches and hyperparameters on the validation set. The final predictions were obtained from a test set of messages from 136 subjects never observed during the training process and evaluated against the task's true labels.

In the tables (Table 5, Table 6) below, we report the relevant metrics obtained for this subtask and compare them against the ones obtained from baseline models provided by the organizers of the competition. In particular, we report both absolute metrics, obtained after observing all the messages of each subject, and early detection metrics, obtained after incrementally observing the messages across several rounds. Additionally, Table 7 displays the inference-time CO2 emissions and energy consumption of each model, based on computing their absolute predictions on the test set. These values were estimated using the codecarbon[14] python library.

Table 6
run compared with the actual model

run	actual_model
run_0	roberta-base-bne-finetuned-suicide-es embeddings and Ridge EmbeddingsRegressor
run_1	models–somosnlp-hackathon-2023–roberta-base-bne-finetuned-suicide-es
run_2	models–PlanTL-GOB-ES–roberta-base-bne
$baseline_1$	Robertuito
$baseline_2$	Roberta Base

Table 7
the inference-time CO2 emissions and energy consumption of each model

run	duration_mean	emissions_mean	cpu_energy_mean	gpu_energy_mean	ram_energy_mean	energy_consumed_mean
run_0	1.00E+00	1.20E-05	3.40E-05	0.00E+00	1.90E-08	3.40E-05
run_1	1.03E+00	1.20E-05	3.43E-05	0.00E+00	1.94E-08	3.43E-05
run_2	1.04E+00	1.22E-05	3.45E-05	0.00E+00	1.95E-08	3.46E-05
$baseline_1$	null	null	null	null	null	null
$baseline_2$	null	null	null	null	null	null

Table 8
The embeddings approaches and regressors combination yielded the results representation

estimator	r2_score_risk	mean_squared_error_risk	mean	std	embedding_models
LinearRegression	-1.5410	0.6352	-0.4529	1.0881	roberta-base-bne
RandomForestRegressor	0.0027	0.2493	0.1260	0.1232	roberta-base-bne
LGBMRegressor	0.0669	0.2332	0.1501	0.0831	roberta-base-bne
GradientBoostingRegressor	0.0187	0.2453	0.1320	0.1132	roberta-base-bne
AdaBoostRegressor	-0.0232	0.2558	0.1162	0.1395	roberta-base-bne
SVR	-0.2316	0.3079	0.0381	0.2697	roberta-base-bne
MLPRegressor	-1.5072	0.6268	-0.4402	1.0670	roberta-base-bne
Lasso	-0.0005	0.2501	0.1247	0.1253	roberta-base-bne
Ridge	-0.0094	0.2523	0.1214	0.1309	roberta-base-bne
LinearRegression	-0.6514	0.4128	-0.1193	0.5321	roberta-base-bne-finetuned-suicide-es
RandomForestRegressor	0.0529	0.2367	0.1448	0.0919	roberta-base-bne-finetuned-suicide-es
LGBMRegressor	-0.0023	0.2505	0.1241	0.1264	roberta-base-bne-finetuned-suicide-es
GradientBoostingRegressor	-0.0480	0.2620	0.1069	0.1550	roberta-base-bne-finetuned-suicide-es
AdaBoostRegressor	0.0055	0.2486	0.1270	0.1215	roberta-base-bne-finetuned-suicide-es
SVR	-0.0316	0.2579	0.1131	0.1447	roberta-base-bne-finetuned-suicide-es
MLPRegressor	0.2625	0.1843	0.2234	0.0391	roberta-base-bne-finetuned-suicide-es
Lasso	-0.0005	0.2501	0.1247	0.1253	roberta-base-bne-finetuned-suicide-es
Ridge	0.1545	0.2113	0.1829	0.0284	roberta-base-bne-finetuned-suicide-es

For the absolute metrics, we show the accuracy, precision, recall, and F1 scores for the classification task (Subtask 1) and the root mean squared error (RMSE) and coefficient of determination (R2) for the regression task (expand Subtask 1). The early detection metrics include the early-risk detection metric (erde) computed after observing different rounds of messages as well as other metrics (more details are provided in the competition guidelines).

The metrics are shown along with the name of the model used to obtain them. The models are named as follows: [model name]-[approach]. For example, roberta-suicide-es-fine-tuning refers to the model trained with the task 1 (binary classification) labels by fine-tuning the Roberta model pre-trained for suicide detection. The " approach " can be either embeddings or fine-tuning for the two approaches described in Section 3.

Furthermore, all ML regressors trained with embeddings as features were simple regressors, and all embeddings were obtained using roberta-suicide-es encodings as this combination yielded the best results in the evaluation set. The embeddings approaches and regressors for task 1 (see Table 8).

For R2-score, it can be understood in a simple way as using the mean as the error reference to see if the prediction error is greater than or less than the mean reference error. R2-score = 1. The predicted values in the sample are exactly equal to the true values without any error, indicating that the interpretation of the dependent variable by the independent variable in the regression analysis is

$$R2= 1-\frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (\bar{y}_i - y_i)^2}$$

Figure 1: R2 coefficient of determination equation of the combination of two different models and estimator.

$$MSE= \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

Figure 2: Mean Squared Error equation of the combination of two different models and estimator.

$$RMSE= \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$$

Figure 3: Root Mean Squared Error equation of the combination of two different models and estimator.

better. R2-score = 0. At this point, the numerator is equal to the denominator, and each predicted value of the sample is equal to the mean. R2-score is not the square of r. It may also be negative (numerator > denominator). The model is equivalent to blind guessing. It is better to directly calculate the average value of the target variable. Specific formula representation see Figure 1.

MSE is the abbreviation of Mean Squared Error and is a commonly used indicator to measure the prediction accuracy of regression models. It represents the average of the sum of squares of the differences between the predicted values and the true values, and is usually used to evaluate the performance of regression models (see Figure 2). RMSE is the abbreviation of Root Mean Squared Error and is a commonly used indicator to measure the prediction accuracy of regression models. It represents the average magnitude of the difference between the predicted value and the true value, and is usually used to evaluate the performance of the regression model (see Figure 3). Among them, y-i is the true value of the i-th sample, ŷ-i is the predicted value of the model for the i-th sample, and m is the number of samples.

The smaller the MSE and RMSE are, the higher the prediction accuracy of the model is. However, it should be noted that MSE and RMSE are greatly affected by outliers. Therefore, in practical applications, a comprehensive evaluation needs to be conducted in combination with other indicators (such as the maximum error, max-error) .

5. Conclusions

The results show that the approaches considered in this work were successful at modeling of the predictive subtasks, with at least one of our models outperforming the baselines in most cases. We can make the following observations:

• The best-performing approach across task1 seems to be the one that uses the embeddings of the messages as input to a simple-output regression model. At least one model trained with this approach

reached the top ranking for task absolute ranking metrics and outperformed the baseline absolute metrics across this task.

• Most notably, the regression method that uses regressors obtained the best metrics for task across all models, outperforming the fine-tuning approach by over 20% in the absolute metrics and reaching the better highest spot in the early-risk metrics for this task in our valid dataset (train dataset split into 15% of subject ids for validation).

• Models trained for single-output regression perform very well for binary classification and simple regression tasks, even outperforming the models trained for simple transformer targets in their own subtask. This suggests that using one model with MLPRegressor to solve for single targets was indeed a good approach to this problem.

• The models obtained with a pure DL approach from fine-tuning a RoBERTa[15] model are estimated to produce over 3-4x less emissions at inference time than the hybrid approach from training linear regressors on sentence embeddings. This gap is likely because the fine-tuning approach requires less computation at inference time than the hybrid approach, which requires the computation of the sentence embeddings before feeding them to regressors, while the fine-tuning approach is made in one forward pass.

Another finding we can conclude from these insights is that while our models achieve great results in the absolute ranking metrics, they do not perform as well for the metrics that assess early-risk performance. In our work, we did not model explicitly for an early detection scenario. We only added information about prior messages through data augmentation. This limitation means our models may not perform as well in real-world situations where we aim to detect signs of gambling disorder in a conversation early on.

Thus, it may be important to explore different training approaches to improve the performance of early-risk detection. This might include directly employing online learning to predict and update the model as new messages come in or incorporating an ensemble of models to make independent decisions about a message's risk level and combining them for a final decision. Additionally, we may also look into more efficient implementations of the hybrid approach to minimize the disparity in emissions compared to pure DL models. These improvements are crucial when considering the deployment of our models in real-world situations and will be the focus of future work.

6. Acknowledgments

Thank you for MentalRiskES@IberLEF 2025 organizing the competition and providing the dataset and other support, and thanks to students of Yunnan University individuals and groups that assisted in the research and the preparation of the work.

7. Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] Álvarez-Ojeda, Pablo, Cantero-Romero, M. Victoria, Semikozova, Anastasia, Montejo-Ráez, Arturo, The precom-sm corpus: Gambling in spanish social media, in: Proceedings of the 31st International Conference on Computational Linguistics, 2025, pp. 17–28.

[2] J. Xiong, O. Lipsitz, F. Nasri, L. M. W. Lui, H. Gill, L. Phan, D. Chen-Li, M. Iacobucci, R. Ho, A. Majeed, R. S. McIntyre, Impact of COVID-19 pandemic on mental health in the general population: A systematic review, in: Journal of Affective Disorders 277, 2020, p. 55–64. URL: https://www.sciencedirect.com/science/article/pii/S0165032720325891. doi:10.1016/j.jad.2020.08.001.

[3] González-Barba, J. Ángel, Chiruzzo, Luis, Jiménez-Zafra, S. María, Overview of IberLEF 2025: Natural Language Processing Challenges for Spanish and other Iberian Languages, in: Proceedings

of the Iberian Languages Evaluation Forum (IberLEF 2025), co-located with the 41st Conference of the Spanish Society for Natural Language Processing (SEPLN 2025), CEUR-WS. org, 2025.
[4] A. M. Mármol-Romero, P. Álvarez-Ojeda, A. Moreno-Muñoz, F. M. P. del Arco, M. D. Molina-González, M.-T. Martín-Valdivia, L. A. Ureña-López, A. Montejo-Ráez, Overview of mentalriskes at iberlef 2025: Early detection of mental disorders risk in spanish, Procesamiento del Lenguaje Natural 75 (2025).
[5] A.G.Fandiño, J.A.Estapé, M. Pàmies, J.L.Palao, J.S.Ocampo, C.P.Carrino, C.A.Oller, C.R.Penagos, A.G.Agirre, M. Villegas, Maria: Spanish language models, Procesamiento del Lenguaje Natural 68 (2022). URL: https://upcommons.upc.edu/handle/2117/367156#.YyMTB4X9A-0.mendeley. doi:10.26342/2022-68-3.
[6] D. L. Padial, D. Gómez, hackathon-somos-nlp-2023-roberta-base-bne-finetuned-suicide-es· hugging face (2023). URL: https://huggingface.co/hackathon-somos-nlp-2023/roberta-base-bne-finetuned-suicide-es.
[7] A. E. Hoerl, R. W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, in: Technometrics, 55–67, [Taylor Francis, Ltd., American Statistical Association,American Society for Quality], 1970, p. 12. URL: https://www.jstor.org/stable/1267351. doi:10.2307/1267351.
[8] L. Breiman, Random forests, in: Machine Learning, volume 45, 2001, p. 5–32. URL: https://doi.org/10.1023/A:1010933404324. doi:10.1023/A:1010933404324.
[9] C. S. Perone, R. Silveira, T. S. Paula, Evaluation of sentence embeddings in downstream and linguistic probing tasks, 2018. URL: http://arxiv.org/abs/1806.06259.
[10] J. Friedman, Greedy function approximation: A gradient boosting machine, in: The Annals of Statistics, 2000, p. 29. doi:10.1214/aos/1013203451.
[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R.Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, in: Journal of Machine Learning Research, 12, 2011, p. 2825–2830.
[12] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, J. B. L. Fang, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, p. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[13] C. C. Liu, J. Pfeiffer, I. Vulić, I. Gurevych, Improving generalization of adapter-based crosslingual transfer with scheduled unfreezing, 2023. URL: http://arxiv.org/abs/2301.05487.
[14] V. Schmidt, K. Goyal, A. Joshi, B. Feld, L. Conell, N. Laskaris, D. Blank, J. Wilson, S. Friedler, S. Luccioni, Codecarbon: estimate and track carbon emissions from machine learning computing, in: Cited on, 2021, p. 20.
[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019. URL: http://arxiv.org/abs/1907.11692. doi:10.48550/arXiv.1907.11692.

A. Appendices

The sources for the fine-tune pre-trained models are available via:

- fine-tune a pre-trained RoBERTa model (models–PlanTL-GOB-ES–roberta-base-bne), see Figure 4.
- fine-tune a pre-trained RoBERTa model (models–somosnlp-hackathon-2023–roberta-base-bne-finetuned-suicide-es), see Figure 5.

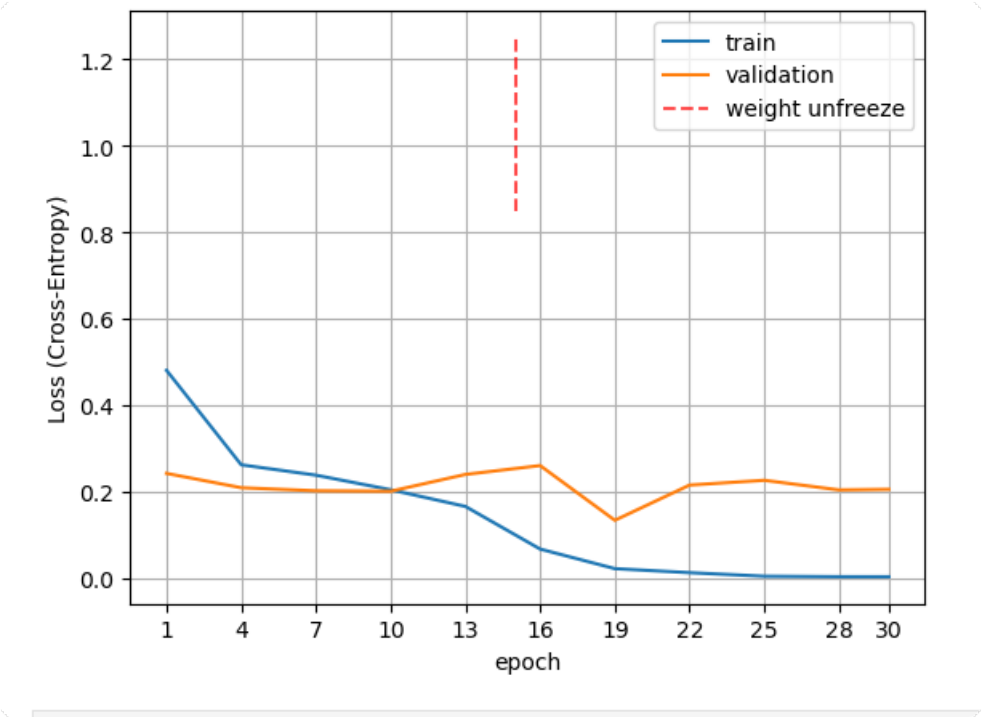

Figure 4: We'll fine-tune a pre-trained RoBERTa model (models–PlanTL-GOB-ES–roberta-base-bne) from HuggingFace to perform single-output regression on this task's data.

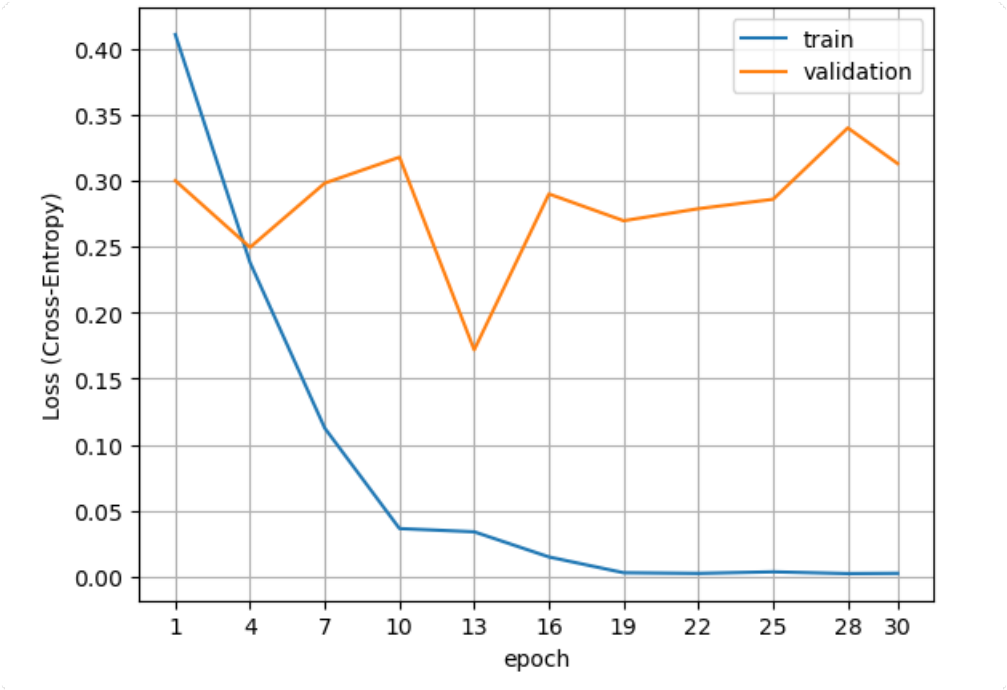

Figure 5: We'll fine-tune a pre-trained RoBERTa model (models–somosnlp-hackathon-2023–roberta-base-bne-finetuned-suicide-es) from HuggingFace to perform single-output regression on this task's data.