A Framework for Identifying Depression on Social Media: MentalRiskES@IberLEF 2023

Simón Sánchez Viloria[1,*], Daniel Peix del Río[1,†], Rubén Bermúdez Cabo[1,†], Guillermo Arturo Arrojo Fuentes[1,†] and Isabel Segura-Bedmar[1]

[1]*Universidad Carlos III de Madrid (UC3M), Av. Universidad, 30 (edificio Sabatini), 28911 Leganés (Madrid), Spain*

Abstract

This paper describes our participation in the MentalRiskES task at IberLEF 2023. The task involved predicting the likelihood of an individual experiencing depression based on their social media activity. The dataset consisted of conversations from 175 Telegram users, each labeled according to their evidence of suffering from the disorder. We used a combination of traditional machine learning and deep learning techniques to solve four predictive subtasks: binary classification, simple regression, multiclass classification, and multiclass regression. We approached this by training a model to solve the multiclass regression case and then transforming the predictions to work for the other three subtasks. We compare the performance of two different modeling approaches: fine-tuning a BERT-based model and using sentence embeddings as inputs to a linear regressor, with the latter yielding better results. The code to reproduce our results can be found at: https://github.com/simonsanvil/EarlyDepression-MentalRiskES

Keywords

Mental Health, Natural Language Processing, Depression, Social Media, Machine Learning, Deep Learning, Transformers, Sentence Embeddings

1. Introduction

Mental health is a growing concern in our society. According to the World Health Organization (WHO), 1 in 4 people will be affected by mental disorders at some point in their lives [1]. In addition, the COVID-19 pandemic 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 [2]. 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 [3].

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 [4].

IberLEF 2023, September 2023, Jaén, Spain

[*]Corresponding author.
[†]These authors contributed equally.
simsanch@inf.uc3m.es (S. S. Viloria); dpeix@pa.uc3m.es (D. P. d. Río); 100384003@alumnos.uc3m.es (R. B. Cabo); 100501115@alumnos.uc3m.es (G. A. A. Fuentes); isegura@inf.uc3m.es (I. Segura-Bedmar)
https://researchportal.uc3m.es/display/inv25506 (I. Segura-Bedmar)
0000-0002-7810-2360 (I. Segura-Bedmar)

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 2023 [5] aims to promote the development of NLP solutions specifically for Spanish-speaking social media. They propose three main areas of focus for early-risk detection: eating disorders (Task 1), depression (Task 2), and non-defined disorders (Task 3).

In this work, we present our proposed solution to Task 2 of the 2023 edition of MentalRiskES. This task involves evaluating the likelihood of a Telegram user experiencing depression based on their comments within mental-health-focused groups. The task is split into four predictive subtasks (2a, 2b, 2c, 2d) according to the type of output required. Our main contributions and findings can be then summarized threefold:

1. We conducted experiments using various language models based on BERT [6] to solve the task. We found that a RoBERTa model [7] that had been previously fine-tuned on a Spanish corpus to identify suicide behavior [8] tended to yield the most accurate results. This suggests that fine-tuning for an intermediate task can improve results for related tasks, which is supported by existing literature [9, 10].
2. Our approach to solving the task consisted of training only with the labels of the regression subtasks (2b, 2d), as we deemed them the most informative. Additionally, we show that you can use the labels of 2d to recover the labels of the other three subtasks. The models trained to target task 2d achieved the best results across all subtasks, even outperforming those that targeted 2b in the simple regression metrics.
3. We attempted two different predictive modeling approaches to solve the task using the language model (LM) mentioned above. The first one involved extracting the *sentence embeddings* of the messages of each user and using them as features to train and evaluate classic linear and non-linear machine-learning regressors. In the second one, we fine-tuned the LM directly for the subtask. The first approach proved advantageous in terms of allowing for quicker, more comprehensive experimentation and resulted in models that achieved the best overall performance when evaluated on the test set.

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 *6,248* individual messages from 175 Telegram users, each with a variable number of messages (see figure 1). The annotation process consisted of labeling each user based on the evidence from their conversation history of suffering from depression. Thus, a total of 10 annotators were used for the tasks. Each was asked to assign one of the following four labels to each user:

- **suffer+in favour**: Indicates evidence (from text messages) of the user suffering from depression but is also receptive/willing to help and overcome it.
- **suffer+against**: Indicates evidence of the user suffering from depression but is against receiving or providing help to overcome it.
- **suffer+other**: Indicates evidence of the user suffering from depression, but there's not enough information to assign them to any further category (against or in favour)
- **control**: Indicates no evidence of the user suffering from depression.

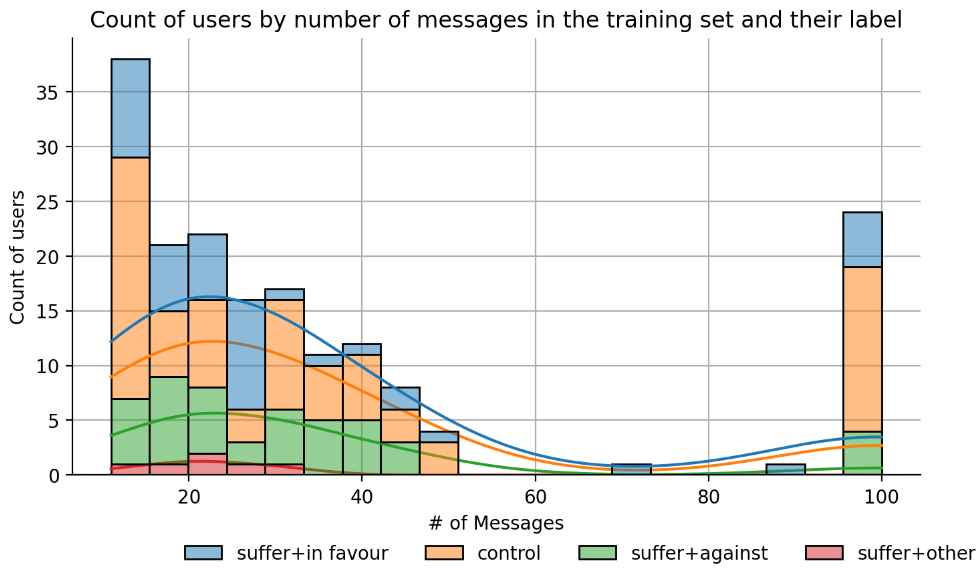

Figure 1: Plot of the count of users by the number of messages in the training set and their assigned label. x-axis: number of messages per user. y-axis: count of users with that amount of messages.

Furthermore, these labels were represented differently to support each of the four subtasks of MentalRiskES: simple classification (*task 2a*), binary regression (*task 2b*), multi-label classification (*task 2c*), and multi-class regression (*task 2d*).

In the classification tasks (2a, 2c), the label assigned to each user was the class that obtained the majority vote from the annotators, with the labels being "1" for the "suffer" classes and "0" for the control in the case of task 2a. For the regression tasks (2b, 2d), the values of the labels were presented as numeric probabilities in $[0, 1]$ representing the confidence of the respective class. They were calculated by adding the number of annotators who gave the classification and dividing by 10 (the total number of annotators). For task 2b, this was presented as one number representing the probability of suffering from depression, while for task 2d, each subject label was presented with four numbers representing the probability of each class. Appendix A shows examples of how this data was given.

The following figure displays the label distribution for each task in the training set. We can see how over 94 ($\sim$54%) of users were classified as having depression. Furthermore, there is an imbalance in the labels for the classification tasks due to the "suffer" label being divided into different categories (leading to an over-representation of the "control" label). Additionally, the "suffer+other" category is underrepresented when compared to the other three.

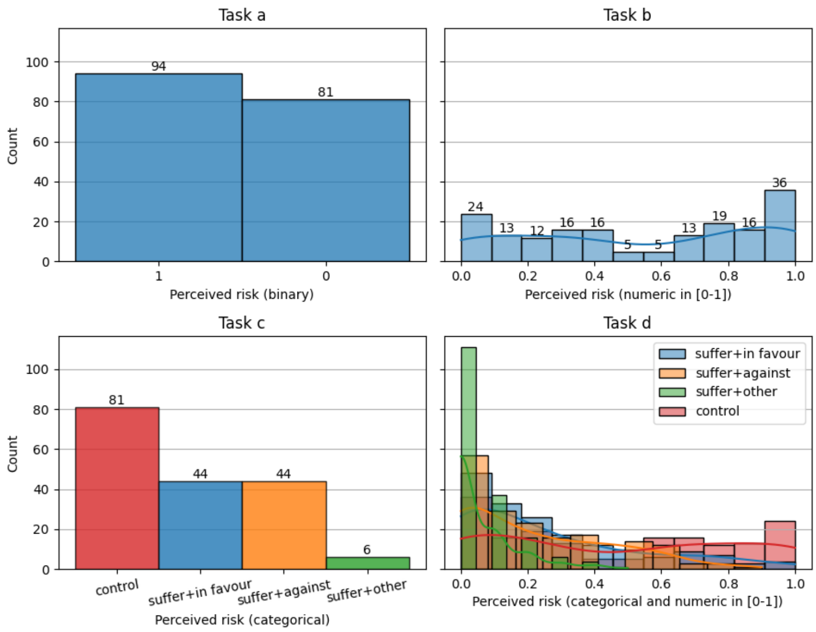

Figure 2. Distribution of labels assigned to the users for each task. In tasks a and d: 1 = evidence of suffering from depression. Task d: The sum of the four labels adds up to 1.

3. Methodology

We proceeded to evaluate different techniques to solve each of the four subtasks. Two main predictive-modeling approaches were explored: The first one involved fine-tuning a pre-trained language model on each 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 each subtask.

3.1. Data Processing and Augmentation

Independent of the approach taken to train the models, the data was pre-processed and augmented in the same way. 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 175 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.

To prepare for training, the data was split into training and validation sets, leaving a random 26 (15%) users in the latter for stratified cross-validation, where each set receives the same proportion of samples of each class [11]. The stratification was done using the labels of task c to ensure equal representation of the classes in both sets.

To increase the amount of data available for training and, 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.

This was done by first sorting the messages of each user in the training set by its date and then only taking the first half, the resulting dataset was then appended to the original training set to obtain a new one with twice the number of observations to be used for training.

3.2. Solving all substaks by solving for regression

By the discussion in section 2, it should be clear to see that not 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. Indeed, it's clear that the probability values of task 2b give more information about confidence in predicting depression than the simple binary labels of task 2a. For the same reasons, the labels of task 2d are more informative than those of task 2c as they give the full probability distribution across the four classes.

Furthermore, we can show that it's possible to use the multiclass-regression labels (2d) to recover the labels of the other three subtasks. To illustrate, the multiclass classification labels of task 2c can be recovered by selecting the class in the distribution that has the highest probability. Moreover, we can obtain the labels of task 2a by simply converting these classes into binary (1 for the "suffer" classes and 0 for all others). Lastly, the labels of task 2b can be obtained by summing the probabilities of the "suffer" classes in the distribution. We have confirmed this by applying these modifications to the labels of the training set for task 2d and comparing them to the original labels of the other three tasks.

This observation led us to consider using models that solve for more than one subtask by only training it with the labels of task 2b or 2d. 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.

Multi-output regression using standard machine learning regression, on the other hand, wasn't as trivial as in the simple regression case. The models we worked on didn't support multi-output regression out of the box. The approach we did involve training four regressors for each model, one for each class, and then combining the predictions. We explored two methods for this: training independent regressors or training them in a chain as explained by figure 3. The full details of the process are described in appendix D.

Finally, similar to the simple regression case, the predictions of the multi-output models were post-processed by dividing each of the four values by their sum to obtain a vector whose values add up to one. That is, $\hat{y}_i = \frac{\hat{y}_i}{\sum_i \hat{y}_i}$ for each i-th class. This was done to ensure that the predictions were valid probability distributions over the classes.

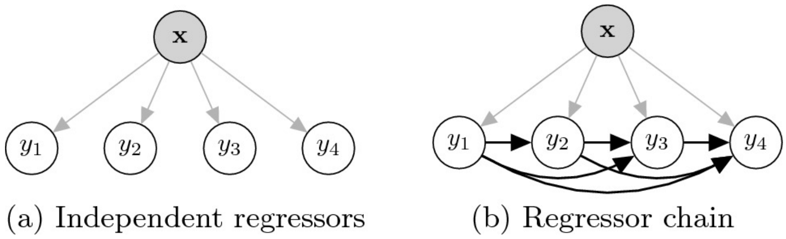

Figure 3: Graphical representation of the two methods used to implement multi-output regression taken from [12]. In (a), the regressors are trained independently with the same input, while in (b) they are trained in a chain with the predictions of the previous ones being passed as features to the next.

3.3. Modeling Approaches

3.3.1. Training a regressor with sentence embeddings

A sentence embedding 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 [13].

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 three different versions of BERT [6] trained with different corpora in Spanish. These versions are described in table 1. Additionally, we experimented with over 10 different regressors, including Least Squares Linear regression [14], Random Forest [15], and Gradient Boosting [16], among others. These models were chosen due to their ease of implementation and the fact that they are commonly used in the literature [17].

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 task 2d (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 B contains the results of this experiment. Based on that, `roberta-suicide-es` was deemed to be the best model for encoding the texts. Additionally, appendix C shows a detailed report of the evaluation of the best regression model with these embeddings.

Model	Description
RoBERTa-base-bne [7]	RoBERTa model [18] trained with data from Spain's National Library.
RoBERTa-suicide-es [8]	RoBERTa-base-bne fine-tuned for suicide detection.
BETO [19]	Variant of BERT [6] trained with Spanish corpora.

Table 1
Pre-trained BERT-based models used in our experiments.

3.3.2. Fine-tuning a Language Model for Regression

Apart from the approach mentioned above, we also experimented with the pure Deep Learning (DL) 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 [8]. 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 [9, 10].

The HuggingFace Transformers [20] and Pytorch [21] 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 or dimension 4 for multi-output regression. The models were trained using an NVIDIA T4 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 each epoch after that as in [22].

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

Table 2
Hyperparameters for fine-tuning a RoBERTa model for regression tasks.

We used an Adam Optimizer with Mean-Squared Error (MSE) for the simple regression models and a Cross-Entropy loss function for multi-regression (since the labels consisted of numeric probabilities). Furthermore, since the output for task 2d consisted of a probability distribution over the four classes, we experimented with a custom loss function that adds a term to the standard cross-entropy loss to penalize outputs whose sum is different from one. However, this did not improve the results empirically as compared with simply normalizing the outputs of the predictions after inference. The formula of this loss is shown in equation 1. Other hyperparameters are shown in table 2.

$$\mathcal{L}_{\text{custom}} = \mathcal{L}_{cross-entropy} + \epsilon(1 - \sum_{i \in [1,4]} \hat{y}_i)^2 \tag{1}$$

In the equation above, $\hat{y}$ is the output of the model, y is the target label, ϵ is a hyperparameter that controls the weight of the penalty term, and y_i is the i-th element of the target label.

4. Results

Using the approaches mentioned in the prior section, we came up with different models to solve the four subtasks of Task 2 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 149 subjects never observed during the training process and evaluated against the task's true labels.

In the tables below, we report the relevant metrics obtained for each 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 11 displays the inference-time CO_2 emissions and energy consumption of each model, based on computing their *absolute* predictions on the test set. These values were estimated using the *codecarbon* python library [23].

For the absolute metrics, we show the accuracy, precision, recall, and F1 scores for the classification tasks (2a and 2c) and the root mean squared error (RMSE) and coefficient of determination (R^2) for the regression tasks (2b and 2d). 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 [5]).

The metrics are shown along with the name of the model used to obtain them. The models are named as follows: *[task name]_[model name]_[approach]*. For example, *task2b_roberta-suicide-es_fine-tuning* refers to the model trained with the task 2b (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 Ridge 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 for task 2d also include the multi-regression method used (*ind* indicating that independent regressors were used and *chain* for chained regressors).

4.1. Results for task 2a: binary classification

Table 3
Task A absolute Metric Results
Ranked by Macro F1.

	accuracy	macro_precision	macro_recall	macro_f1
2d_roberta_embeddings_ind	**0.705**	**0.717**	**0.727**	**0.703**
BaseLine - Roberta Large	0.698	0.759	0.718	0.690
2d_roberta_embeddings_chain	**0.691**	**0.711**	**0.755**	**0.682**
2b_roberta_embeddings	**0.691**	**0.713**	**0.764**	**0.681**
2d_roberta-suicide-es_fine-tuning	**0.671**	**0.695**	**0.764**	**0.655**
BaseLine - Deberta	0.664	0.788	0.691	0.642
2b_roberta-suicide-es_fine-tuning	**0.638**	**0.663**	**0.735**	**0.616**
BaseLine - Roberta Base	0.631	0.744	0.658	0.605

Table 4
Task A early-detection Metric Results
Ranked by ERDE30.

	erde30	erde5	latency_tp	latency_weighted_f1	speed
2d_roberta-suicide-es_fine-tuning	**0.013**	**0.284**	**3.000**	**0.716**	**0.982**
2b_roberta_embeddings	**0.020**	**0.286**	**3.000**	**0.725**	**0.982**
2b_roberta-suicide-es_fine-tuning	**0.020**	**0.208**	**2.000**	**0.700**	**0.991**
2d_roberta_embeddings_chain	**0.027**	**0.283**	**3.000**	**0.722**	**0.982**
2d_roberta_embeddings_ind	**0.067**	**0.296**	**3.000**	**0.712**	**0.982**
BaseLine - Deberta	0.153	0.303	2.000	0.719	0.984
BaseLine - Roberta Large	0.159	0.290	4.000	0.704	0.951
BaseLine - Roberta Base	0.176	0.342	4.000	0.671	0.951

4.2. Results for task 2b: Simple Regression

Table 5
Task B absolute Metric Results
Ranked by RMSE.

	RMSE	r2
2d_roberta_embeddings_chain	**0.241**	**0.591**
2b_roberta_embeddings	**0.244**	**0.581**
2d_roberta_embeddings_ind	**0.259**	**0.526**
BaseLine - Roberta Base	0.277	0.770
2d_roberta-suicide-es_fine-tuning	**0.304**	**0.349**
2b_roberta-suicide-es_fine-tuning	**0.311**	**0.317**
BaseLine - Deberta	0.339	0.683
BaseLine - Roberta Large	0.390	0.503

Table 6
Task B Ranking-based Results: Ranked by the p@30 metric

	p@10	p@20	p@30	p@5
BaseLine - Roberta Base	0.800	0.700	0.567	0.600
BaseLine - Deberta	0.600	0.550	0.567	0.800
BaseLine - Roberta Large	0.500	0.550	0.567	0.400
2b_roberta-suicide-es_fine-tuning	**0.700**	**0.700**	**0.533**	**1.000**
2b_roberta_embeddings	**0.800**	**0.450**	**0.367**	**0.800**
2d_roberta_embeddings_ind	**0.700**	**0.350**	**0.233**	**0.600**
2d_roberta_embeddings_chain	**0.200**	**0.150**	**0.133**	**0.000**
2d_roberta-suicide-es_fine-tuning	**0.200**	**0.100**	**0.100**	**0.200**

4.3. Results for task 2c: Multiclass Classification

Table 7
Task C absolute Metric Results
Ranked by Macro F1.

	accuracy	macro_precision	macro_recall	macro_f1
2d_roberta-suicide-es_fine-tuning	**0.517**	**0.446**	**0.435**	**0.395**
2d_roberta_embeddings_ind	**0.557**	**0.429**	**0.395**	**0.394**
2d_roberta_embeddings_chain	**0.530**	**0.437**	**0.418**	**0.392**
BaseLine - Roberta Large	0.483	0.389	0.378	0.360
BaseLine - Deberta	0.456	0.395	0.344	0.293
BaseLine - Roberta Base	0.356	0.380	0.335	0.274

Table 8
Task C early-detection Metric Results:
Ranked by ERDE30.

	erde30*	erde5	latency_tp	latency_weighted_f1	speed
2d_roberta_embeddings_chain	**0.157**	**0.284**	**3.000**	**0.718**	**0.982**
2d_roberta-suicide-es_fine-tuning	**0.159**	**0.285**	**3.000**	**0.712**	**0.982**
2d_roberta_embeddings_ind	**0.172**	**0.297**	**3.000**	**0.708**	**0.982**
BaseLine - Deberta	0.190	0.330	2.000	0.695	0.984
BaseLine - Roberta Base	0.206	0.307	2.000	0.659	0.984
BaseLine - Roberta Large	0.232	0.283	2.000	0.652	0.984

4.4. Results for task 2d: Multi-output Regression.

Table 9
Task D absolute Metric Results.
Ranked by mean RMSE. Labels are shortened as: sf = suffer+in favour, sa = suffer+against, so = suffer+other, c =control

	rmse mean*	rmse sf	rmse sa	rmse so	rmse c	r2 mean	r2 sf	r2 sa	r2 so	r2 c
2d_roberta_embeddings_chain	**0.180**	**0.179**	**0.191**	**0.111**	**0.241**	**0.355**	**0.544**	**0.217**	**0.069**	**0.590**
2d_roberta_embeddings_ind	**0.187**	**0.181**	**0.192**	**0.114**	**0.259**	**0.320**	**0.532**	**0.208**	**0.012**	**0.526**
2d_roberta-suicide-es_fine-tuning	**0.222**	**0.212**	**0.230**	**0.143**	**0.304**	**0.006**	**0.358**	**-0.144**	**-0.538**	**0.349**
BaseLine - Deberta	0.232	0.246	0.250	0.125	0.306	0.484	0.661	0.295	0.260	0.721
BaseLine - Roberta Base	0.410	0.547	0.272	0.235	0.585	-0.145	-0.496	0.355	0.185	-0.624
BaseLine - Roberta Large	0.437	0.682	0.312	0.158	0.598	-0.209	-0.678	0.890	0.059	-0.306

Table 10
Task D ranking Metric Results: Ranked by p@30

	p@10	p@20	p@30	p@5	p@50
BaseLine - Deberta	0.300	0.338	0.350	0.250	0.250
2d_roberta_embeddings_ind	**0.300**	**0.300**	**0.292**	**0.600**	**0.280**
BaseLine - Roberta Large	0.275	0.263	0.275	0.350	0.350
2d_roberta_embeddings_chain	**0.275**	**0.275**	**0.250**	**0.550**	**0.240**
BaseLine - Roberta Base	0.300	0.225	0.192	0.250	0.250
2d_roberta-suicide-es_fine-tuning	**0.075**	**0.113**	**0.167**	**0.150**	**0.145**

4.5. Carbon Emissions

Table 11
Estimated CO_2 emissions of each model from predicting all messages on the test set (absolute).
Estimations were obtained with the `codecarbon` python library [23] using a Macbook Pro (2021) w/ M1 Pro and 16GB of RAM for inference. The models with the lowest emissions are highlighted in bold.

model	duration (secs)	emissions (kgCO2eq)	cpu_energy	ram_energy
2b_roberta-suicide-es_fine-tuning	**5.74**	**1.56e-06**	7.97e-06	2.53e-07
2d_roberta-suicide-es_fine-tuning	**7.393**	**2.00e-06**	1.03e-05	2.58e-07
2b_roberta_embeddings	23.287	6.70e-06	3.23e-05	2.94e-06
2d_roberta_embeddings_ind	23.976	6.94e-06	3.33e-05	3.25e-06
2d_roberta_embeddings_chain	23.721	7.14e-06	3.29e-05	4.63e-06

5. Conclusions

The results show that the approaches considered in this work were successful at modeling each 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 all tasks seems to be the one that uses the embeddings of the messages as input to a multi-output regression model (task 2d). At least one model trained with this approach reached the top ranking for tasks 2a, 2b, and 2d absolute ranking metrics and outperformed the baseline absolute metrics across all tasks.
- Most notably, the regression method that uses multi-output chained regressors obtained the best metrics for task 2d across all models, outperforming the fine-tuning approach by over 20% in the absolute metrics and reaching the second highest spot in the early-risk metrics for this task.
- Models trained for multi-output regression perform very well for binary classification and simple regression tasks, even outperforming the models trained for simple regression targets in their own subtask. This suggests that using one model to solve for multiple targets was indeed a good approach to this problem.

- The models obtained with a pure DL approach from fine-tuning a RoBERTa 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 multiple 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 depression 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 (as seen in [23]). 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.

References

[1] World Health Organization, The World Health Report 2001: Mental Disorders affect one in four people, 2001. URL: https://www.who.int/news/item/28-09-2001-the-world-health-report-2001-mental-disorders-affect-one-in-four-people.

[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, Journal of Affective Disorders 277 (2020) 55–64. URL: https://www.sciencedirect.com/science/article/pii/S0165032720325891. doi:10.1016/j.jad.2020.08.001.

[3] D. E. Losada, F. Crestani, J. Parapar, erisk 2017: Clef lab on early risk prediction on the internet: experimental foundations, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction: 8th International Conference of the CLEF Association, CLEF 2017, Dublin, Ireland, September 11–14, 2017, Proceedings 8, Springer, 2017, pp. 346–360.

[4] A. Dargahi Nobari, N. Reshadatmand, M. Neshati, Analysis of Telegram, An Instant Messaging Service, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 2035–2038. URL: https://dl.acm.org/doi/10.1145/3132847.3133132. doi:10.1145/3132847.3133132.

[5] A. M. Mármol-Romero, A. Moreno-Muñoz, F. M. Plaza-del Arco, M. T. Martín-Valdivia, L. A. Ureña-López, A. Montejo-Ráez, Overview of MentalriskES at IberLEF 2023: Early

Detection of Mental Disorders Risk in Spanish, Procesamiento del Lenguaje Natural 71 (2023).
[6] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019. URL: http://arxiv.org/abs/1810.04805. doi:10.48550/arXiv.1810.04805, arXiv:1810.04805 [cs].
[7] 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.
[8] 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.
[9] J. Phang, T. Févry, S. R. Bowman, Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks, 2019. URL: http://arxiv.org/abs/1811.01088. doi:10.48550/arXiv.1811.01088, arXiv:1811.01088 [cs].
[10] T.-Y. Chang, C.-J. Lu, Rethinking Why Intermediate-Task Fine-Tuning Works, 2021. URL: http://arxiv.org/abs/2108.11696, arXiv:2108.11696 [cs].
[11] R. Kohavi, A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, in: IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence, 1995, pp. 1137–1143. URL: https://www.semanticscholar.org/paper/A-Study-of-Cross-Validation-and-Bootstrap-for-and-Kohavi/8c70a0a39a686bf80b76cb1b77f9eef156f6432d.
[12] E. Antonenko, J. Read, Multi-modal Ensembles of Regressor Chains for Multi-output Prediction, in: T. Bouadi, E. Fromont, E. Hüllermeier (Eds.), Advances in Intelligent Data Analysis XX, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 1–13. doi:10.1007/978-3-031-01333-1_1.
[13] 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, arXiv:1806.06259 [cs] version: 1.
[14] A. E. Hoerl, R. W. Kennard, Ridge Regression: Biased Estimation for Nonorthogonal Problems, Technometrics 12 (1970) 55–67. URL: https://www.jstor.org/stable/1267351. doi:10.2307/1267351, publisher: [Taylor & Francis, Ltd., American Statistical Association, American Society for Quality].
[15] L. Breiman, Random Forests, Machine Learning 45 (2001) 5–32. URL: https://doi.org/10.1023/A:1010933404324. doi:10.1023/A:1010933404324.
[16] J. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, The Annals of Statistics 29 (2000). doi:10.1214/aos/1013203451.
[17] 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, Journal of Machine Learning Research 12 (2011) 2825–2830.
[18] 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, arXiv:1907.11692

[cs].
[19] J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang, J. Pérez, Spanish pre-trained bert model and evaluation data, in: PML4DC at ICLR 2020, 2020, pp. 1–10.
[20] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush, Transformers: State-of-the-Art Natural Language Processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. URL: https://aclanthology.org/2020.emnlp-demos.6. doi:10.18653/v1/2020.emnlp-demos.6.
[21] 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, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[22] C. C. Liu, J. Pfeiffer, I. Vulić, I. Gurevych, Improving Generalization of Adapter-Based Cross-lingual Transfer with Scheduled Unfreezing, 2023. URL: http://arxiv.org/abs/2301.05487, arXiv:2301.05487 [cs].
[23] 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, Cited on (2021) 20.

A. Dataset Examples

The data was given in JSON format after requesting the server. The following examples are meant to show the structure of how the data was given and later parsed.


```json
[
  {
    "id_message": "1",
    "message": "Me parece que es una buena idea, pero no estoy seguro",
    "date":  "2020-07-27 01:27:31"
  },
  {
      "id_message": 2,
      "message": "Buen dia a todos",
      "date": "2020-07-27 02:03:28"
  },
]
```


Example of the raw data describing the messages of one user. The original training set (later split into training+validation) constituted 175 JSON files like this.

To complement the files described in the example above, the labels of each subject were given in CSV format, where each row corresponded to one subject. Four of these files were given, one for each task. Table 12 below shows some examples of this.

subject_id	a_label	b_label	c_label	d_suffer_in_favour	d_suffer_against	d_suffer_other	d_control
subject101	1	0.9	suffer+in favour	0.7	0.1	0.1	0.1
subject104	1	0.7	suffer+in favour	0.4	0.0	0.3	0.3
subject106	1	1.0	suffer+in favour	0.5	0.5	0.0	0.0
subject108	1	0.5	suffer+in favour	0.4	0.1	0.0	0.5
subject109	0	0.1	control	0.0	0.0	0.1	0.9

Table 12

Example of user labels for each task. The letter in the prefix of the column name indicates the label of the task (See section 2). For example, column *a_label* indicates the label of the users for task 2a.

B. Evaluation of Embedding Models for Regression

Table 13 shows the scores after evaluating with different encodings for task 2d. The sentence embeddings were obtained after concatenating the messages of each user into a single string. RMSE scores were calculated as the mean of the results of 10 regressors trained with the respective encoding as features. The best-performing embeddings were the ones obtained with the RoBERTa model fine-tuned for suicide detection.

encoder	suffer+in favour	suffer+against	suffer+other	control	mean
roberta-base-bne-suicide-es	0.228	0.198	0.113	0.249	0.204
roberta-base-bne	0.220	0.208	0.119	0.277	0.214
bert-base-spanish-wwm-cased (BETO)	0.233	0.214	0.132	0.273	0.221

Table 13

Mean RMSEs scores after training 10 regressors with embeddings from different transformers.

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.

The implementation of the sentence embeddings was done using the HuggingFace library. Before obtaining the encodings, the models were loaded from the HuggingFace Hub. The reference and links to the models in Hugging Face is included in 1.

C. Evaluation of Regression models trained with the Sentence Embeddings Approach

Tables 14 and 15 report the regression metrics obtained on the validation set for various estimators trained with the `roberta-base-bne-suicide-es` embeddings. The results were obtained after evaluating over 10 estimators for the task, selecting the 4-6 best-performing regressors, and doing grid-search hyperparameter-tuning on them.

By these results, the Ridge Regression model (ridge), which implements Least Squares Linear Regression with L2 regularization, seems to be the best estimator for both tasks. The best-performing models of each task were then trained with the entire data (train+validation sets) and the prediction of the test set was obtained with them. These evaluations with these predictions are the ones reported in section 4.

Table 14
Binary regression results in the evaluation set of various estimators trained with embeddings for task 2b.

estimator	rmse	r2
ridge	0.254	0.489
ada	0.257	0.480
lgbm	0.259	0.471
svr	0.260	0.467
rf	0.264	0.448
mlp	0.319	0.198

Table 15
Multiclass regression results in the evaluation set of various estimators trained with embeddings for task 2d. Labels are shortened as: sf = suffer+in favour, sa = suffer+against, so = suffer+other, c = control

estimator	RMSE mean	RMSE sf	RMSE sa	RMSE so	RMSE c	R2 mean	R2 sf	R2 sa	R2 so	R2 c
ridge	0.183	0.172	0.193	0.108	0.260	0.271	0.410	0.287	-0.076	0.465
lr	0.193	0.202	0.197	0.108	0.267	0.201	0.183	0.256	-0.073	0.439
rf	0.206	0.195	0.220	0.106	0.305	0.133	0.240	0.071	-0.045	0.267
lgbm	0.288	0.285	0.262	0.181	0.424	-0.854	-0.631	-0.321	-2.043	-0.421

The estimators mentioned in the table above are implementations of common regressors from Python's Scikit-Learn library [17]. These include: Ordinary ("lr") and Ridge ("ridge") Least Squares Regression, Ada-Boost regression ("ada"), Light Gradient Boosting Machine ("lgbm"), Support Vector Regression ("svr"), Random Forests ("rf"), and a Multi-Layer Perceptron ("mlp"). References of the implementations of these models can be found in the Scikit-Learn documentation.

D. Multi-Output Regression with Independent Regressors and Regressor Chains

For task 2d, we were required to obtain four values corresponding to a probability distribution over the four classes (suffer+in favour, suffer+against, suffer+other, control). In section 3.2, we explained how this multi-output regression problem can be solved for the sentence embeddings approach by training four regressors and then combining their predictions using either the Independent Regressors or Regressor Chain methods.

Here we explain how these methods work and were implemented. First of all, the two methods can be summarized as follows, depending on how the regressors are trained to obtain the probability distributions. Figure 3 shows a graphical representation of the two methods.

1. **Independent Regressors**: Each regressor is trained independently with the labels of its corresponding class (e.g., the first regressor was trained with the labels of the suffer+in favour class, the second with the labels of the suffer+against class, and so on). The downside of this method is that it doesn't take into account the information of the other classes when training each regressor, which is important as we know the labels are not independent of each other (they must all sum to 1).
2. **Regressor Chain Method**: The regressors are trained in a chain, where the first regressor is trained to predict the first class, and its predictions are included in the features for the second regressor, and so on. This method is useful when the labels of each class are not independent of each other (like in our case), as it allows the regressors to learn from the predictions of the previous ones. Since the order of the classes matters in this method, we decided to put them in the order of most to least amount of users annotated with that class: control, suffer+in favour, suffer+against, suffer+other (see section 2).

To the second method, we can additionally add the option of applying Principal Component Analysis (PCA) to reduce the dimensionality of the input embeddings before training the models in the chain. Because the embeddings might have a large dimensionality, this is done to make these models more likely to use the information of the previous predictions. Both methods were implemented with the Scikit-Learn library [17] using the `MultiOutputRegressor` and `RegressorChain` classes. The number of components to keep for PCA was chosen using based on the percent of variance explained. The number of components was fine-tuned and the best results was obtained with 40 components (85% of variance).