UNSL at eRisk 2022: Decision policies with history for early classification

Juan Martín Loyola[1,3], Horacio Thompson[1,2], Sergio Burdisso[1,4] and
Marcelo Errecalde[1]

[1]*Universidad Nacional de San Luis (UNSL), Ejército de Los Andes 950, San Luis, C.P. 5700, Argentina*

[2]*Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)*

[3]*Instituto de Matemática Aplicada San Luis (IMASL), CONICET-UNSL, Av. Italia 1556, San Luis, C.P. 5700, Argentina*

[4]*Idiap Research Institute, Rue Marconi 19, 1920 Martigny, Switzerland*

Abstract

For the 2022 edition of the CLEF eRisk Laboratory, our research group at Universidad Nacional de San Luis (UNSL) added new approaches and improvements concerning our last participation. We proposed two decision policies for EarlyModel that take into account historic information available to the models, and incorporated two score normalization steps into the SS3 model. We also significantly reduced the runtime to process the inputs. Despite not having achieved the best performances, our team obtained the best results for the $ERDE_{50}$ in tasks T1 and T2. Besides, considering the $F_{latency}$, we were the third-best team for both tasks. Finally, a couple of our models got some of the best performance for the ranking-based metrics for task T1.

Keywords

Early Risk Detection, Early Classification, Learned Early Alert Policy

1. Introduction

The 2022 edition of the early risk prediction on the Internet laboratory (eRisk) [1] presents two tasks for early risk detection: early detection of signs of pathological gambling and early detection of depression. Both had been introduced in previous editions [2, 3, 4], thus the participants have a corpus to use for training or validation. The performance was assessed with the same metrics as last edition. That is, the standard classification measures (precision, recall, and F_1 score), measures that penalize delay in the response (ERDE [5] and $F_{latency}$ [6]), and ranking-based evaluation metrics were used. The F_1 and $F_{latency}$ scores were computed with respect to the positive class only.

The remaining sections describe the models and datasets used, and discuss the results obtained. Section 2 gives a brief description of the methods and points out the main differences from our last participation. Sections 3 and 4 describe the datasets, the models' parameters, and their

results for Task 1 and Task 2, respectively. Finally, Section 5 closes with the conclusions of the work.

2. Method

Our participation in this edition followed similar steps as the last edition [7]. The corpus generation and data pre-processing steps did not change. The same kinds of models were used, but we improved upon them and proposed new decision policies. Also, to compare our performance with the models from last year, we trained and validated our models using the generated corpus and tested them using the provided corpus. Finally, we reduced the time each run took to process the writings.

In what follows, the enhancements with respect to our previous participation are presented.

2.1. Early risk detection models

The models proposed for early risk detection were based on the early classification framework presented by Loyola et al. [8]. This framework divides the task into two separated but related problems: classifying with partial information and deciding the moment of classification. The task of classification with partial information (CPI) consists in obtaining an effective model that predicts the class of a document only using the available information. On the other hand, the task of deciding the moment of classification (DMC) involves determining the point at which we can stop reading the input with some certainties that the prediction made will be correct.

We used the same kind of models as last year: EarlyModel [7], EARLIEST [9], and SS3 [10]. The main difference was in the DMC component. We proposed two new decision policies for EarlyModel and two different normalization steps for the SS3 scores. The new decision policies consider the model's past scores and other information from the documents context. The motivation behind the normalization of the SS3 scores was to restrict the scores to the interval $[0, 1]$ and avoid an infinite increasing score [1].

Next, we briefly describe each early risk detection model and the proposed improvements.

EarlyModel. This model tackles both tasks of early classification separately. Thus, it is composed of two parts: a classifier with partial information and a decision policy in charge of raising an alarm. While the input is being processed, the partial classifier categorizes it, returning the class probability. The class probability and other information from the context are fed to the decision policy that decides if we should stop processing the input and raise an alarm or continue reading. The EarlyModels were built following our last edition workflow. The best *<representation, classifier with partial information>* pairs were selected and integrated with a decision policy. This year three policies were evaluated:

---

[1]Note that the score of each user is additive. That is, as new posts arrive, the user's score could potentially increase more and more, never reaching a limit. Since SS3 has a global decision policy, the score of all users is considered to make a decision. Even users that have already finished processing the input impact the decision of active users.

- **SimpleStopCriterion**. Decision policy used last year. To decide if an alarm for a user had to be emitted, three attributes were consider: the predicted class, the current delay, and the predicted positive class probability. If the user is predicted as positive, the probability of belonging to the positive class exceeds a certain threshold (δ), and more than n posts are processed, then an alarm is raised. Different combinations of δ and n were tested, selecting the best ones according to the F_{latency} measure obtained in the validation stage.
- **HistoricStopCriterion**. To avoid hasty alarms, SimpleStopCriterion was extended by considering some of the model's previous positive class probabilities. This policy defines that if the current probability exceeds the threshold and the last m predictions also do so, the system must issue a user at-risk alarm and end the analysis; otherwise, it is necessary to continue evaluating the user.
- **LearnedDecisionTreeStopCriterion**. Decision policy based on a learned decision tree. This was only used in the depression task since the laboratory organizers provided two datasets, one of which we could use to train the model without leakage of information. In order to train this model, we labelled the point p in which the system could emit an alarm. Fifty positive users of the depression training corpus for eRisk 2018 [2] were labelled for this task. Half of them were used for training and the other half for validation. For each user, multiple samples were generated, one for each of the posts. A sample in time t contained all the publications in time i such that $i \leq t$. To label each sample, we compared the time t of the sample with the decision point p. If $t < p$, a negative label was assigned to the sample. For the following ten samples, $p \leq t < p + 10$, the positive label was assigned. The features calculated for each sample were: current positive class probability given by the CPI, an average of the last ten positive class probabilities given by the CPI, an average of the last five positive class probabilities given by the CPI, median of the last ten positive class probabilities given by the CPI, current delay, the current label assigned by the CPI, an average of the last ten labels assigned by the CPI, number of words in the top 0.01 percentile of information gain for the depression training corpus for eRisk 2018, and the number of words in the top 0.015 percentile of chi-squared for the depression training corpus for eRisk 2018. Then, a decision tree model was trained using group k-fold to ensure all the samples for a user were in the same group. Grid search was used to find the best parameters. Finally, the best model was evaluated in the testing corpus built.

EARLIEST. It is an end-to-end deep learning model that tackles both early classification tasks at the same time. The model is composed of three parts: a base RNN that summarizes the partial input, a controller that decides if we should continue or not processing the input, and a discriminator that classifies the partial input once the controller halts the processing. Reinforcement learning is used in order to train the model. This year, a couple of bugs in the implementation were fixed and the model was forced to make decisions after seen five posts at least.

---

[2]Note that this corpus was not used for training, validation or testing of the system, thus any leakage was avoided.

SS3. It is a two-part early classification model where SS3 [10] is used as a classifier with partial information, and a user-global early alert policy is proposed to halt the processing. The policy used to raise an alarm for a particular user takes into account its score value, globally, with respect to the current score of all the other users. This year the users' scores were normalized using two approaches:

- **N1**. First, to limit the score to the interval $[0, 1]$, the *softmax* function was applied to the confidence values (*cv*) that the model gave to the positive and negative classes. Given the additive nature of the confidence values and the behaviour of the softmax function when the values increase [3], the confidence values were divided by the current delay of the document. For the users that do not have more posts, the delay is equal to the total number of posts sent. Finally, the score for a user was calculated as:

$$softmax\left(\frac{cv_{\text{positive}}}{\text{delay}}, \frac{cv_{\text{negative}}}{\text{delay}}\right)$$

- **N2**. Second, the proportion of the positive confidence value given the sum of both positive and negative confidence values was used. Thus, the score for a user was computed as:

$$\frac{cv_{\text{positive}}}{cv_{\text{positive}} + cv_{\text{negative}}}$$

2.2. Training workflow

Following other teams decision from previous editions of the laboratory, we augmented the provided corpus with posts from Reddit. Though, this time, we didn't used the provided corpus for the initial training and validation of the models. The generated Reddit corpus was used to select the models and its hyper-parameters. Once we selected the best models, we evaluated them in the provided corpus. Note that these datasets were used for testing in previous editions of the laboratory, thus we were able to compare the performance. Finally, before we started processing the writings, we retrained the models with the obtained hyper-parameters using all the datasets available.

In order to speed the training process and to simulate the laboratory pipeline, a mock server was developed [4]. This server replicate the GET/POST behaviour of the eRisk laboratory. That is, a team can ask for new writings using the same GET request structure as the one used during the laboratory. Similarly, the same POST request structure can be used to send the team response for each run. Besides these, the mock server can:

- Manage teams: create, list, and get information of a team.
- Show a separation plot for a given team and time.
- Plot the user score evolution for a given team and run.
- Plot the elapsed times of teams.
- Show the table with the results of all finished experiments.
- Plot the server elapsed times to answer the requests.

---

[3]When the values given to the function increase beyond one, the function assigns most of the probability to the largest input. On the other hand, when the parameters tend to zero, the softmax function returns equal probability to all the inputs.

[4]The source code and instructions to run the mock server are available at: https://github.com/jmloyola/erisk_mock_server.

Table 1

Details of the corpora used for Task T1: the different training and validation sets, as well as the test set used by the eRisk organizers to evaluate the participating models. The number of users (total, positives, and negatives) and the number of posts of each corpus are reported. The median, minimum, and maximum number of posts per user and words per post in each corpus are detailed.

Corpus	#users			#posts	#posts per user			#words per post		
	Total	Pos	Neg		Med	Min	Max	Med	Min	Max
T1_test	2,079	81	1,998	1,177,590	297	3	2,001	11	0	6,728
T1_valid	2,348	164	2,184	1,130,799	244	10	2,001	11	1	8,241
T1_redd_train	1,746	286	1,460	158,924	51	31	1,188	20	1	7,479
T1_redd_valid	1,746	286	1,460	161,204	53	31	1,337	20	1	3,234

2.3. Speed up runs

One of the concerns about our participation last year was the time it took our models to process the writings. Input/output operations to communicate with the laboratory server caused part of this delay. Each model had to wait for the previous one to finish sending its responses to start processing the input. Therefore, this year we processed each writing concurrently which allowed us to reduce the total processing time. That is, some models could be already processing the input while others could still be sending data and/or be waiting for server responses. We used the *asyncio* [5] Python package to implement this behaviour.

3. Task T1: Early Detection of Signs of Pathological Gambling

In this section, the details of our participation addressing the eRisk's early detection of pathological gambling task are given. Namely, the details of the datasets and the five models submitted to this challenge are introduced. Finally, the results obtained after the evaluation stage are shown.

3.1. Datasets

For Task T1, the eRisk's organizers provided a corpus to train, validate or test the participating models. The corpus was made available as a set of XML files, one for each user. In our case, we used the provided corpus to compare our results with last edition models, thus we initially did not train or validate our models using this. In order to train and validate our models a complementary corpus was built using data from Reddit following the same steps as last year [7]. This corpus was split into a training and a validation set, each containing half of the users. Finally, once the best hyper-parameters were found using the generated datasets, all the datasets (including the one provided by the organizers) were used to retrain the models before deploying them.

   Table 1 shows the details of each complementary corpus along with the validation and test datasets provided for this task. In this table, "T1_test" refers to the test set used to evalu-

---

ate all participating models, "T1_valid" to the validation set provided by the organizers, and "T1_redd_train" and "T1_redd_valid" to the training and validation sets built using Reddit.

Note that the corpus used to evaluate the participating models this year was more unbalanced than last year, thus, probably making this year task more difficult to address. Note that in the provided corpus 6.9% of the users are positives, whereas in the corpus used to evaluate participating models, only 3.8%. On the other hand, T1_redd_train and T1_redd_valid had a much lower number of total posts and posts per user than the datasets used by the organizers. Finally, in T1_test, there were empty posts (without words), which could be due to users who edited their posts after posting, deleting their content.

3.2. Models

This section describes the details of the models used by our team to tackle this task. Namely, from the results obtained after the model selection and the hyper-parameter optimization stage, the following five models were selected for participating:

> **UNSL#0**. An EarlyModel with a bag of words (BoW) representation and a logistic regression classifier. Words unigrams were used for the BoW representation with term frequency times inverse document-frequency (commonly known as *tf-idf*) as the weighting scheme. For the logistic regression, a balanced weighting for the classes was used, that is, each input was weighted inversely proportional to its class frequencies in the input data. Finally, for the decision-making policy, a *SimpleStopCriterion* with threshold $\delta = 0.7$ and minimum number of posts $n = 10$ was used.

> **UNSL#1**. An EarlyModel with a BoW representation and a support vector machine (SVM) classifier. For the BoW representation, character 4-grams were used with tf-idf as the weighting scheme. The support vector machine was parameterized with a radial basis function kernel with *gamma="scale"* and regularization parameter $C = 2$ weighted inversely proportional to its class frequencies in the input data. Finally, for the decision-making policy, a *SimpleStopCriterion* with threshold $\delta = 0.7$ and minimum number of posts $n = 10$ was used.

> **UNSL#2**. An EarlyModel based on BERT with an extended vocabulary. For the fine-tuning process, the following parameters were used: *architecture=BERT-based-uncased* [6], *optimizer=Adam*, *learning_rate=3e-5*, *batch_size=8*, and *n_epochs=3*. Also, 25 new words were added to the BERT's vocabulary [11] by applying the following process: an SS3 model was trained on all available data (Reddit and eRisk2021's datasets), then words were ordered according to their confidence values on the positive class, and finally, the top-25 most important words were extracted. Finally, the decision policy *HistoricStopCriterion* was applied with a threshold $\delta = 0.7$, a minimum number of posts $n = 10$, and considering the last $m = 10$ previous predictions.

---

[6] https://huggingface.co/bert-base-uncased

UNSL#3. An SS3 model[7] with a policy value of $\gamma = 2.5$ and the normalization *N1*.

UNSL#4. An EARLIEST model with a doc2vec representation. Each post was represented as a 300-dimensional vector. To learn this representation the Reddit training corpus, T1_redd_train, was used. The base recurrent neural network chosen was a one-layer LSTM with an input feature dimension of 300 and 256 hidden units. The discriminator of the EARLIEST model reduced the hidden state of the LSTM to two dimensions representing the probabilities of both, the positive and negative classes. Finally, the value of lambda used to train was λ = *1e-4*.

3.3. Results

Table 2 shows the results obtained for the decision-based performance metrics. As can be seen, our team achieved the best performance in terms of the $ERDE_{50}$ measure. Besides, considering all teams, we obtained the third-best team performance for the $ERDE_5$, F_1, and $F_{latency}$ measures. Despite not having obtained the best results, our team achieved competitive performance in most metrics, exceeding the average level among all teams for this edition.

Analyzing the performance of our team, the EarlyModels (UNSL#0, #1, and #2) reached the best score in the $ERDE_{50}$ measure, but only UNSL#1 could stand out, achieving the best F_1 and $F_{latency}$ scores. However, in $ERDE_5$, these models showed a lower performance. The SS3 model (UNSL#3) proved to be competitive, achieving acceptable performance in all metrics and, in particular, obtaining the best results for $ERDE_5$. Finally, the EARLIEST model (UNSL#4), did not perform well on this task.

On the other hand, Table 3 shows the results obtained for the performance metrics based on rankings. Our team achieved the best performance in most metrics with respect to the different rankings used for the evaluation. In particular, the EarlyModels (UNSL#0 and UNSL#1) were able to stand out from other models presented by our team. The only exceptions was NDCG@100 with 1 and 500 posts, where the results were very close to the best of the competition: for 1 post, the best was 0.76 by UNED-NLP#2 and our best model obtained 0.70; and for 500 posts, the best result was 0.95 by UNED-NLP#4 and our best model got 0.93.

It is also interesting to note that most of the NDCG@100 values improved as more writings were processed. In early classification problems, this is usually common since the models' accuracy improves as the size of the input increases. However, an excessive delay in classification can become a problem, as it could put people's lives at risk. An early decision could be made considering less data, but still, accuracy remains key to the final performance of the models. This behavior is not present for the ranking measures that only consider the top 10 scores (P@10 and NDCG@10) because their reduced sample size makes them more unstable. One sample could bias the final measure, unlike when 100 samples are used.

---

[7]SS3 models were coded in Python using the "PySS3" package [12] (https://github.com/sergioburdisso/pyss3).

Table 2

Decision-based evaluation results for Task T1. The best teams taking into account the $ERDE_5$, $ERDE_{50}$, and $F_{latency}$ are shown (values in bold), as well as the median and mean values of the results report for CLEF eRisk 2022.

Model	P	R	F_1	$ERDE_5$	$ERDE_{50}$	$latency_{TP}$	speed	$F_{latency}$
UNSL#0	0.401	0.951	0.564	0.041	**0.008**	11.0	0.961	0.542
UNSL#1	0.461	0.938	0.618	0.041	**0.008**	11.0	0.961	0.594
UNSL#2	0.398	0.914	0.554	0.041	**0.008**	12.0	0.957	0.531
UNSL#3	0.365	0.864	0.513	0.017	0.009	3.0	0.992	0.509
UNSL#4	0.052	0.988	0.100	0.051	0.030	5.0	0.984	0.098
UNED-NLP#4	0.809	0.938	**0.869**	0.020	**0.008**	3.0	0.992	**0.862**
SINAI#1	0.575	0.802	0.670	**0.015**	0.009	**1.0**	**1.000**	0.670
BLUE#0	0.260	0.975	0.410	**0.015**	0.009	**1.0**	**1.000**	0.410
Mean	0.223	0.846	0.279	0.034	0.021	4.8	0.985	0.297
Median	0.116	0.963	0.205	0.037	0.015	2.8	0.993	0.211

Table 3

Ranking-based evaluation results for Task T1. Results are reported according to the three classification metrics obtained after processing 1, 100, 500, and 1000 posts, respectively.

Ranking	Metric	UNSL#0	UNSL#1	UNSL#2	UNSL#3	UNSL#4
1 post	P@10	**1.00**	**1.00**	0.90	**1.00**	0.10
	NDCG@10	**1.00**	**1.00**	0.90	**1.00**	0.07
	NDCG@100	0.68	0.70	0.66	0.69	0.32
100 posts	P@10	**1.00**	**1.00**	**1.00**	0.60	0.10
	NDCG@10	**1.00**	**1.00**	**1.00**	0.58	0.07
	NDCG@100	**0.90**	**0.90**	0.77	0.72	0.32
500 posts	P@10	**1.00**	**1.00**	0.90	0.80	0.20
	NDCG@10	**1.00**	**1.00**	0.92	0.81	0.13
	NDCG@100	0.93	0.92	0.78	0.77	0.33
1000 posts	P@10	**1.00**	**1.00**	0.90	0.80	0.30
	NDCG@10	**1.00**	**1.00**	0.90	0.81	0.22
	NDCG@100	**0.95**	0.93	0.77	0.78	0.37

4. Task T2: Early Detection of Depression

In this section, the details of our participation addressing the eRisk's early detection of depression task are given. Namely, the details of the datasets and the five models submitted to this challenge are introduced. Finally, the results obtained after the evaluation stage are shown.

4.1. Datasets

For Task T2, the eRisk's organizers provided datasets to train and validate the participating models. Each corpus was made available as a set of XML files, one for each user. Part of the supplied training corpus was used to train the decision policy *LearnedDecisionTreeStopCriterion*. The provided validation corpus was used to compare our results with the models from eRisk

Table 4
Details of the corpora used for Task T2: the different training and validation sets, as well as the test set used by the eRisk organizers to evaluate the participating models. The number of users (total, positives, and negatives) and the number of posts of each corpus are reported. The median, minimum, and maximum number of posts per user and words per post in each corpus are detailed.

Corpus	#users			#posts	#posts per user			#words per post		
	Total	Pos	Neg		Med	Min	Max	Med	Min	Max
T2_test	1,400	98	1,302	898,326	457.0	6	2,000	12	0	8,009
T2_train	887	135	752	531,394	321.0	10	2,000	13	1	7,450
T2_valid	820	79	741	545,188	411.5	10	2,000	13	1	7,280
T2_redd_train	1,056	499	557	142,059	66.0	31	2,282	21	1	6,792
T2_redd_valid	1,057	500	557	130,534	61.0	31	2,220	20	1	6,629

2018 [8]. In order to train and validate our models a complementary corpus was built using data from Reddit following the same steps as last year [7]. This corpus was split into a training and a validation set, each containing half of the users. Next, the best hyper-parameters were found using the generated datasets and, finally, all the datasets (including the ones provided by the organizers) were used to retrain the models before deploying them.

Table 4 shows the details of each complementary corpus along with the validation and test datasets provided for this task. In this table, "T2_test" refers to the test set used to evaluate all participating models, "T2_train" and "T2_valid" to the training and validation sets provided by the organizers, and "T2_redd_train" and "T2_redd_valid" to the training and validation sets built using Reddit.

As with Task T1, the corpus used to evaluate the participating models was more unbalanced than in previous years. Note that in the provided training and validation sets 15.21% and 9.63% of the users were positive, respectively, whereas in the test set only 7%. On the other hand, the number of total posts and posts per user in T2_redd_train and T2_redd_valid was notably smaller than in T2_test (used for evaluation). Finally, in the same way as with T1_test, in T2_test there were empty posts with no words in them, i.e. empty posts.

4.2. Models

This section describes the details of the models used by our team to address this task. Namely, from the results obtained after model selection and hyper-parameter optimization, the following five models were selected for participating:

> **UNSL#0**. An EarlyModel with latent semantic analysis representation and a logistic regression classifier. Fifty factors were used to depict each user, transforming the input and projecting it into the space generated by the single value decomposition algorithm trained in T2_redd_train. For the logistic regression, a balanced weighting for the classes was used, that is, each input was weighted inversely proportional to its class frequencies in the input data. Finally, for the decision-making policy, a *LearnedDecisionTreeStopCriterion* was used. The first levels of the obtained decision tree are shown in Figure 1 and the full

---
[8]The last year the task of early detection of depression was evaluated was 2018.
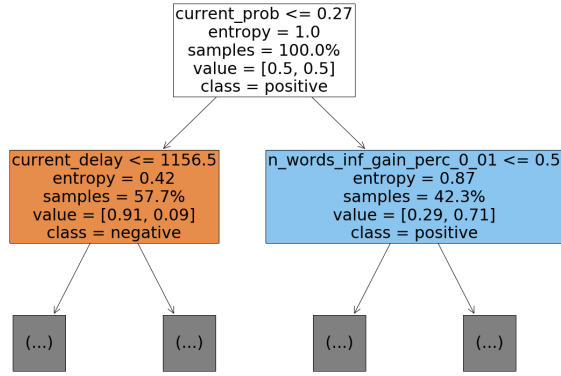
Figure 1: First levels of the decision tree classifier learned for task T2 in the context of the *LearnedDecisionTreeStopCriterion* decision policy.

decision tree is shown in Figure 2. The best hyper-parameters found after grid search were: *class_weight="balanced"*, *criterion="entropy"*, *max_depth=4*, *min_samples_leaf=1*, and *splitter="best"*.

UNSL#1. An EarlyModel with a BoW representation and an SVM classifier. For the BoW representation, character 3-grams were used with tf-idf as the weighting scheme. The support vector machine was parameterized with a radial basis function kernel with *gamma="scale"* and regularization parameter $C = 8$. Finally, for the decision-making policy, a *SimpleStopCriterion* with threshold $\delta = 0.7$ and minimum number of posts $n = 10$ was used.

UNSL#2. An SS3 model with a policy value of $\gamma = 2.5$ and the normalization *N1*.

UNSL#3. An SS3 model with a policy value of $\gamma = 2$ and the normalization *N2*.

UNSL#4. An EARLIEST model with the same hyper-parameters and structure as the one used in UNSL#4 for Task T1. In this case, to learn the doc2vec representation, T2_redd_train was used.

4.3. Results

The Table 5 shows the results obtained for the decision-based performance metrics. As can be seen, our team achieved the best performance in terms of the ERDE_{50} measure together with SCIR2. Besides, considering all teams, we obtained the second-best team performance for the ERDE_5 measure and we were the third-best team for the F_{latency}.

If we compare our models performance, we can see that UNSL#2 (SS3) outperform the rest in almost every metric. Looking at the performance of both SS3 models (UNSL#2 and UNSL#3),

Table 5
Decision-based evaluation results for Task T2. The best teams taking into account the $ERDE_5$, $ERDE_{50}$, and $F_{latency}$ are shown (values in bold), as well as the median and mean values of the results report for CLEF eRisk 2022.

Model	P	R	F_1	$ERDE_5$	$ERDE_{50}$	$latency_{TP}$	speed	$F_{latency}$
UNSL#0	0.161	0.918	0.274	0.079	0.042	14.5	0.947	0.260
UNSL#1	0.310	0.786	0.445	0.078	0.037	12.0	0.957	0.426
UNSL#2	0.400	0.755	0.523	0.045	**0.026**	3.0	0.992	0.519
UNSL#3	0.144	0.929	0.249	0.055	0.035	3.0	0.992	0.247
UNSL#4	0.080	0.918	0.146	0.099	0.074	5.0	0.984	0.144
NLPGroup-IISERB#0	0.682	0.745	**0.712**	0.055	0.032	9.0	0.969	**0.690**
LauSAn#4	0.201	0.724	0.315	**0.039**	0.033	**1.0**	**1.000**	0.315
SCIR2#3	0.316	0.847	0.460	0.079	**0.026**	44.0	0.834	0.383
Mean	0.200	0.730	0.278	0.068	0.048	22.8	0.922	0.288
Median	0.149	0.847	0.249	0.070	0.041	6.0	0.981	0.256

Table 6
Ranking-based evaluation results for Task T2. Results are reported according to the three classification metrics obtained after processing 1, 100, 500, and 1000 posts, respectively.

Ranking	Metric	UNSL#0	UNSL#1	UNSL#2	UNSL#3	UNSL#4
	P@10	0.60	**0.80**	0.70	0.10	0.10
1 post	NDCG@10	0.40	**0.88**	0.68	0.06	0.12
	NDCG@100	0.36	0.46	0.50	0.15	0.05
	P@10	0.20	0.60	0.50	0.40	0.00
100 posts	NDCG@10	0.13	0.73	0.39	0.27	0.00
	NDCG@100	0.46	0.64	0.55	0.43	0.03
	P@10	0.30	0.60	0.70	0.30	0.20
500 posts	NDCG@10	0.28	0.73	0.73	0.21	0.19
	NDCG@100	0.43	0.66	0.61	0.42	0.07
	P@10	0.60	0.60	0.70	0.30	0.00
1000 posts	NDCG@10	0.72	0.71	0.73	0.21	0.00
	NDCG@100	0.45	0.66	0.61	0.42	0.04

we can see that the choice of normalization step and the value of γ play a critical role. However, considering the EarlyModels, UNSL#0 and UNSL#1 achieved measures barely over the median of all teams. Finally, the EARLIEST model (UNSL#4), did not perform well on this task either.

On the other hand, Table 6 shows the results obtained for the performance metrics based on rankings. Here, our team did not accomplish a performance as good as with task T1. The EarlyModel UNSL#1 was the only one to reach the best performance among all teams when reading one post for P@10 and NDCG@10. UNSL#2 came close to this model, surpassing it in some metrics when reading more than 500 posts. The rest were not able to achieve good enough results.

In this task, we observe the same behavior with NDCG@100 as with task T1. When the number of posts processed increased, the ranking measure improved. EARLIEST was the only model that didn't show this behavior, which could be related to the model underfitting the data.

5. Conclusions

In this paper we briefly presented the models proposed for early risk detection by our team from Universidad Nacional de San Luis for tasks T1 and T2 of the eRisk 2022 laboratory. The differences with our previous participation are described in detail and the final results are presented. Furthermore, we show a summary of the datasets used, both the provided and the generated ones.

Even though we improved last year's models, the performance obtained was not as good as the last edition. Nonetheless, our team got the best score for $ERDE_{50}$ for T1 and T2. Also, we were the third-best team with respect to the $F_{latency}$ metric for both tasks. Finally, considering the ranking-based metrics, our results for T1 were one of the best among all the teams.

References

[1] J. Parapar, P. Martín-Rodilla, D. E. Losada, F. Crestani, Overview of eRisk 2022: Early risk prediction on the internet, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. 13th International Conference of the CLEF Association, CLEF 2022, Springer, 2022.

[2] D. E. Losada, F. Crestani, J. Parapar, eRisk 2017: CLEF lab on early risk prediction on the internet: experimental foundations, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2017, pp. 346–360.

[3] D. E. Losada, F. Crestani, J. Parapar, Overview of eRisk: early risk prediction on the internet, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2018, pp. 343–361.

[4] J. Parapar, P. Martín-Rodilla, D. E. Losada, F. Crestani, Overview of eRisk 2021: Early risk prediction on the internet, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2021, pp. 324–344.

[5] D. E. Losada, F. Crestani, A test collection for research on depression and language use, in: Proc. of Conference and Labs of the Evaluation Forum (CLEF 2016), Evora, Portugal, 2016, pp. 28–39.

[6] F. Sadeque, D. Xu, S. Bethard, Measuring the latency of depression detection in social media, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 495–503.

[7] J. M. Loyola, S. Burdisso, H. Thompson, L. Cagnina, M. Errecalde, UNSL at eRisk 2021: A comparison of three early alert policies for early risk detection, in: Working Notes of CLEF 2021-Conference and Labs of the Evaluation Forum, Bucarest, Romania, 2021.

[8] J. M. Loyola, M. L. Errecalde, H. J. Escalante, M. M. y Gomez, Learning when to classify for early text classification, in: Argentine Congress of Computer Science, Springer, 2017, pp. 24–34.

[9] T. Hartvigsen, C. Sen, X. Kong, E. Rundensteiner, Adaptive-halting policy network for early classification, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 101–110.

[10] S. G. Burdisso, M. Errecalde, M. Montes-y Gómez, τ-SS3: A text classifier with dynamic

n-grams for early risk detection over text streams, Pattern Recognition Letters 138 (2020) 130 – 137. doi:https://doi.org/10.1016/j.patrec.2020.07.001.

[11] W. Tai, H. Kung, X. L. Dong, M. Comiter, C.-F. Kuo, exBERT: Extending pre-trained models with domain-specific vocabulary under constrained training resources, in: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 1433–1439.

[12] S. G. Burdisso, M. Errecalde, M. Montes-y Gómez, PySS3: A python package implementing a novel text classifier with visualization tools for explainable ai, arXiv preprint arXiv:1912.09322 (2019).
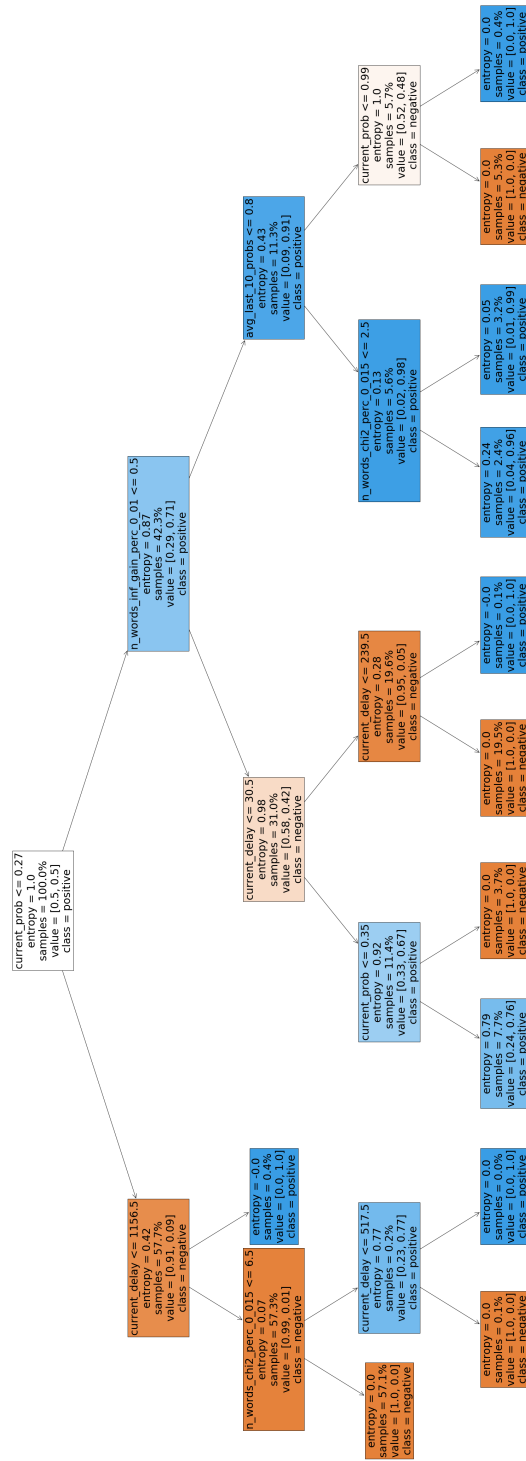
Figure 2: Decision tree classifier learned for task T2 in the context of the *LearnedDecisionTreeStopCrite-rion* decision policy.