Early risk detection of mental illnesses using various types of textual features

Rodrigo Ferreira, Alina Trifan and José Luís Oliveira

DETI/IEETA, University of Aveiro, Portugal

Abstract
This paper documents the participation of our team, BioInfo@UAVR, in the first and second task of the 2022 edition of CLEF eRisk. With the goal of achieving an early detection of subjects at risk of specific mental illnesses, pathological gambling and depression for tasks 1 and 2 respectively, using data sourced from social networks, more specifically, Reddit. To this end, we trained several machine learning models, dividing our experiments into three feature engineering approaches of increasing complexity, corresponding to very commonly used vectorization methods in natural language processing, those of bag-of-words, distributional semantics word embeddings and contextualized language models. Additionally, we evaluated the impact of the inclusion of sentiment analysis features. Despite having subpar results on the official evaluation, we managed to improve them considerably post submission with a few tweaks, showing that these solutions can work if properly fine-tuned.

Keywords
Natural language processing, Machine learning, Mental health, Social mining

1. Introduction

Mental health is a state of well-being related to an individual's cognitive, behavioral, and emotional state, and it plays an important role in self-esteem along with our life. Mental health disorders, given their less physical nature when compared to something like muscle, skeletal or organ issues (which can be extracted/poked/seen through imaging) can be harder to properly diagnose. Sadly, the lack of proper support surrounding these disorders as well as the stigma attached to them, often prevent individuals from seeking out help, which may lead to the worsening of their conditions and to the potential risk of self-harm or even suicidal behaviours. According to statistics published by the World Health Organization (WHO)[1], over 700 000 people die to suicide each year, with many more having failed attempts. Many of these could be prevented with the appropriate care.

The traditional process of diagnosing a mental illness is far from optimal since it may fail due to various factors. First, the requirement of the patient's physical presence may be hard to fulfil due to lack of access, or the secluding nature of many mental illnesses and the social stigma surrounding them. For the same reasons, the subject might not feel comfortable being transparent when describing their situation to the healthcare professional. Patient aside, it may fail from the lack of experience of the healthcare professional, which may sometimes be only a

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
rodrigommf@ua.pt (R. Ferreira); alina.trifan@ua.pt (A. Trifan); jlo@ua.pt (J. L. Oliveira)

[1]https://www.who.int/teams/mental-health-and-substance-use/data-research/suicide-data

general health practitioner from a health center or emergency room with somewhat limited expertise. Naturally, it might also fail due to the screening tool used, although those tend to be sturdy with large amounts of research backing them up.

It should come as no surprise that with all these possible failing points, the traditional methods lead to suboptimal results. This is where the idea of using social data comes in, since it addresses many of these points. Social data corresponds to the data that social media users publicly share in an online scenario (Reddit[2], Twitter[3], etc), its public and easily accessible nature make it a big facilitator for a lot of big data applications by substituting or complementing traditional data, which is harder to acquire. Naturally, it also brings a fair share of concerns, despite the fact that it is public, users might not be comfortable having their data being used for purposes other than the typical use of social media.

The process of analyzing social data with the end goal of understanding some trends, opinions or behaviours within a population is called social monitoring, it has been used across many fields [1] with goals such as:

- measuring consumer sentiment.
- measuring political sentiment.
- forecasting sales.
- estimating traffic congestion.
- forecasting elections.
- contagious disease surveillance.
- monitoring gun violence cases.

Given that the traditional screening solutions in the mental health field can fail at various points, a social monitoring approach may be useful since it may be able to tackle some of the issues found in the traditional methods. Imagining, for example, a depression screening tool employed in social media, one can easily see how the requirement of being physically present with a health specialist is solved. And, since there is no direct interaction with a health specialist, the users might also feel more comfortable to accurately portray their thoughts, as they already tend to do on social media.

Naturally, this is not a perfect solution as there are still some concerns, namely regarding the effectiveness of such screening tool and the public's acceptance to being subjected to it, since in the past, tools of this type have been met with concerns from the public due to ethical concerns on the use of potentially sensitive personal data in a way that was not intended by the author at the time of publication [1]. An example of a target of the public's backlash was a Twitter plug-in called Samaritan's Radar [4] that monitored user's social media data without their consent with the hope of notifying its user when the accounts they followed showed signs of struggling with suicidal thoughts. It launched in October 2014, it was suspended the following month and by March of 2015 it was permanently closed, due to the public's concern for being monitored without consent, and the potential dangers originated by this tool if used with bad intentions (as it makes it easier to target suicidal people).

[2]https://www.reddit.com
[3]https://twitter.com
[4]https://www.samaritans.org/about-samaritans/research-policy/internet-suicide/samaritans-radar/

Having that said, the early risk[5] (eRisk) workshop, a part of the Conference and Labs of the Evaluation Forum[6] (CLEF), incentivizes the development of solutions capable of capturing traits of individuals at risk on the internet, in the context of shared tasks, by providing test collections[2], while employing, besides the classic options, novel evaluation metrics like ERDE[2] and latency-weighted F score[3], that evaluate the participating models not only on the correctness of their classifications, but also on their timeliness, since faster classifications allow for better interventions. The focus is put on common issues in the mental health area, having tackled the detection of issues such as depression, self-harm, eating disorders and pathological gambling. Here, we document the development of models potentially capable of classifying users with signs of mental illnesses, namely pathological gambling and depression, using writings posted by them on the Reddit social network, corresponding to tasks 1 and 2 of the 2022 edition of eRisk[4].

2. Data

This section seeks to explain what datasets were used to accomplish each task, the preprocessing procedures and how they were split to allow for our machine learning experimentations.

2.1. Task 1: Pathological gambling

In order to address this challenge, the official dataset provided by the CLEF organizers was used, it consisted of last year's test dataset for the same task [5]. The dataset was made available on a dedicated server and it consists of a golden truth text file, that maps subjects to their corresponding label, 0 or 1, for control and pathological gamblers respectively, and a folder containing multiple files in the xml format, one for each subject, containing information regarding their writings (posts/comments).

In its original state, there were records of writings for 2348 subjects, 164 of which were labeled 1, and 2184 were labeled 0. In order to achieve a more user friendly dataset format, after preprocessing, all writings were aggregated to a single csv file.

Plenty of writings had artifacts of the extraction process, mostly in the format of html tags. At this stage, a jupyter notebook was used, it would iterate through the golden truth text file, save the subject id's and corresponding labels and then iterate through each of the subject's xml files. For each file the writings were subject to the following sequence of transformations (done mostly with regular expressions[7] and contractions[8] libraries):

1. alphabetical characters were converted to lowercase.
2. html and decimal unicode artifacts were converted to their corresponding symbols.
3. various types of emojis such as ":)" and ":(" were converted to "smiling emoji" and "negative emoji" respectively.

[5]https://erisk.irlab.org
[6]https://clef2022.clef-initiative.eu
[7]https://docs.python.org/3/library/re.html
[8]https://github.com/kootenpv/contractions

4. mentions of website links, reddit users and subreddits were converted to the tokens "url", "user" and "subreddit" respectively.
5. numbers beginning or ending with "€" (Euro) or "$" (Dollar) were substituted by the token "money".
6. remaining isolated tokens comprised only of numbers were converted to the token "number".
7. contractions of common words were transformed to their original form (for example "you're", "I'm" turn into "you are" and "i am").
8. a dictionary of common internet slang terms was constructed to convert words to their original form (for example "omg", "ty" turn into "oh my god" and "thank you").
9. punctuation characters were discarded except for "!.?,", as these can be important for some of the features extraction methods used.

The resulting output was put through a fasttext language detection model [9] [6, 7], and posts classified as non-English were discarded, while English writings were printed to a csv file. Regarding the language detection, this model was chosen due to its speed and decent accuracy. With some manual checking early on, there was a significant amount of misclassifications, especially with shorter texts and text containing lots of abbreviations and acronyms. In order to improve this, the script was changed to return the top 2 most likely languages for each writing, and if English was present in the top 2, the writing would be kept, or otherwise discarded. With this change, upon some manual checking, the outputs seemed to be much better, and of 1129560 total writings, 57534 were filtered out. The remaining preprocessed posts were then aggregated in a csv file in original and lemmatized forms.

The resulting dataset was significantly imbalanced, the control group represented 93% (2184) of the total subjects vs 7% (164) of pathological gamblers, and 95% (1016947) of the total writings vs 5% (54915) for pathological gamblers. The method chosen to handle this imbalance was random undersampling in terms of users (not writings), which was repeatedly executed to bring the number of control group subjects down to match the number of positive subjects in a way that results in a somewhat balanced number of writings for each class (since subjects do not have a fixed amount of writings).

With the undersampling approach we end up with a dataset balanced in terms of subjects of each class (164 each), and nearly balanced in terms of posts from subjects of each class (65266 vs 54915). At this point, the data was split 80%/20% into a training and validation sets respectively, the split was performed on subjects, ensuring that any given subject's writings may only appear in either the training or validation set. This was made to make sure that there is no information leak and that we can evaluate the trained models on data belonging to completely new subjects.

The official testing set was only made available at the time of submission, so it wasn't used in the initial training stage but only afterwards when evaluating, it was comprised of 81 positive and 1998 control subjects, with 14627 and 1014122 writings respectively. More information on all mentioned sets of data can be found in Table 1.

[9] https://fasttext.cc/docs/en/language-identification.html

Table 1
Composition of the various sets of data for task 1.

Set	Subjects/Writings		
	Positive	Negative	Total
Original	164 (7%) / 55677 (5%)	2184 (93%) / 1073883 (95%)	2348 / 1129560
Preprocessed	164 (7%) / 54915 (5%)	2184 (93%) / 1016947 (95%)	2348 / 1071862
Undersampled	164 (50%) / 65266 (54%)	164 (50%) / 54915 (46%)	328 / 102181
Training set	131 (50%) / 44805 (46%)	131 (50%) / 53053 (54%)	262 / 97858
Validation set	33 (50%)/10110 (45%)	33 (50%) / 12213 (55%)	66 / 22323
Testing set	81 (4%) / 14627 (1%)	1998 (96%) / 1014122 (99%)	2079 / 1028749

Table 2
Composition of the various sets of data for task 2.

Set	Subjects/Writings		
	Positive	Negative	Total
Original	214 (13%) / 90222 (8%)	1493 (87%) / 986360 (92%)	1707 / 1076582
Preprocessed	214 (13%) / 89010 (8%)	1493 (87%) / 970622 (92%)	1707 / 1059632
Undersampled	214 (50%) / 89010 (42%)	214 (50%) / 121917 (58%)	428 / 210927
Training set	171 (50%) / 73670 (42%)	171 (50%) / 100503 (58%)	342 / 174173
Validation set	43 (50%) / 15340 (42%)	43 (50%) / 21414 (58%)	86 / 36754
Testing set	98 (7%) / 35332 (5%)	1302 (93%) / 687228 (95%)	1400 / 722560

2.2. Task 2: Depression

In accordance with the choice made for task 1, for task 2, only the official data provided by the organizers was used, which consisted of the training and testing datasets from 2017[8] and the testing dataset from 2018's edition of eRisk[9]. This dataset's structure was slightly different than that of the previous task, instead of using a golden truth text file, the xml files containing each user's writings were grouped into positive and negative folders, corresponding to the 1 and 0 labels respectfully, and divided by years, since as mentioned, the organizers provided data from 2017 and 2018. Regarding the individual files the format was the same as in the pathological gambling task.

Originally, the data from 2017 contained 752 negative and 135 positive subjects. Likewise, the 2018 folder contained 741 negative and 79 positive subjects. This comes up to a total of 1707 subjects, 214 of which are positive (depressed). Like before, the total writings were aggregated into a single csv file.

The preprocessing was the same as in the previous task, following the same transformations and language detection methods. Out of 1076582 total writings, 16950 were discarded. Though not as much as the pathological gambling case, this dataset was still heavily imbalanced, with 87% of the users being labeled as negative, and 13% as positive, and 92% of the writings corresponding to the negative class vs 8% of the positive users.

Naturally, a random undersampling algorithm was employed, to reduce the amount of control group subjects from 1493 to 214 in order to match the amount of depressed subjects. As expected, the number of writings also became more balanced, going from 970622 negative writings to 121917, much closer to the number of positive writings of 89010. Again, this resulting dataset was split into a 80%/20% train/validation split, divided by users, leading to a training dataset containing 171 positive and 171 negative subjects, and 73670 positive vs 100503 negative writings. The validation set contained 43 positive and 43 negative subjects and 15340 positive vs 21414 negative writings. As it was the case previously, we only obtained the testing dataset at the time of submission, it was comprised of 35332 writings belonging to 98 positive subjects and 687228 belonging to 1302 control subjects. A brief summary of the mentioned sets of data can be seen in Table 2

3. Feature engineering techniques

When it comes to feature engineering, many different approaches have been used in the past. In this work we rely mostly on textual features, though experiments with sentiment analysis features were also integrated.

First, we must decide on what constitutes a single sample. Is it a single writing from a subject, multiple writings grouped by some criteria (length, time of posting, etc), or even all of their writings? Initially, user writings were used individually as samples, but this quickly led to subpar results, likely due to the fact that some writings are extremely short and it is difficult to assess which class a writing's author belongs to given only 2 or 3 words. To overcome this, inspired by NLP-UNED's run [10] in 2021's edition of CLEF eRisk, a k-sliding-window method was implemented, where each sample consists of the last k writings at the time. The choice of k is also important here, in initial experiments we were essentially employing a sliding window of k=1, we also performed experiments k values of 3 and 5 which resulted in immediate improvements across the board. Larger values were likely to produce better results for the first 2 feature engineering approaches, which we will discuss next, but given the sequence length limitation of the language model used in approach 3 (since the windows are constructed prior to the feature extraction process), to keep things fair and simple, we decided to ensure that the models submitted for each task all ran on writing windows of the same length, resulting in a maximum k size of 5.

The textual features were essentially split into 3 categories: we implemented models using Bag of Words (BoW) features with Tf-Idf in approach 1, GloVe [11] word embeddings in approach 2, and contextualized language model representations in approach 3. The previously mentioned sentiment analysis features are those of sentiment analysis tools in Vader[10] and TextBlob[11]. TextBlob's tool provided us with 2 scores for each text analyzed, regarding its subjectivity (opinionated or fact based) and polarity (how positive or negative the text is). Vader's features consisted of a 4 values corresponding to scores regarding a text's negativity, neutrality, positivity, and a compound value which acts as a single overall measure of the previous values. The set of features generated from these two tools will be referred to from now on as SA features (for Sentiment Analysis). Experiments were ran with textual features alone and with textual and SA features, to evaluate their effectiveness in this scenario.

[10]https://github.com/nltk/nltk/blob/develop/nltk/sentiment/vader.py
[11]https://github.com/sloria/textblob

3.1. Approach 1: Bag-of-Words

A Bag-of-Words (BoW) feature extraction method treats documents exactly as the name suggests, documents are defined by the presence/amount of each word it contains. This method works but is usually too simplistic, it is often improved by employing a Term-frequency Inverse-document-frequency (Tf-Idf) weighting methodology. This methodology tries to weight how important a certain word is in a given document relative to the whole corpus assigning higher weights to words that occur more often within fewer documents. For a given term t, document d found in corpus D:

$$Tf(t,d) = \frac{occurrences\ of\ t\ in\ d}{total\ occurrences\ in\ d} \tag{1}$$

$$Idf(t,D) = \log \frac{|D|}{|documents\ in\ D\ with\ t|} \tag{2}$$

$$TfIdf(t,d,D) = Tf(t,d) \times Idf(t,D) \tag{3}$$

The end result of this feature extraction method is the learned vocabulary and Idf weights, when vectorizing a document we can calculate the Tf-Idf weights of its tokens by performing the previously mentioned calculations with the learned Idf weights for the tokens present in the learned vocabulary (with unknown tokens being usually ignored), providing a simple yet effective way to represent text as number vectors. This process can be altered by changing some parameters such as:

- max features: maximum amount of tokens in the learned vocabulary.
- n_gram range: how many tokens to group as a unit in the vocabulary (1-gram being groups of 1 token, 2-grams 2 tokens, etc).
- stopwords list: tokens that are not valid candidates for the learned vocabulary and are discarded.
- max/min document frequency: filter out tokens that occur in more/less than x documents, for max and min respectively, with x being an absolute value or a fraction.

Thanks to the library sklearn [12], this is all implemented in the form of the class TfIdfVectorizer which allows us to create a vectorizer that given the desired parameters and a training corpus, can later transform samples of text into number vectors.

To find the optimal Tf-Idf parameter values (of those previously mentioned) to use in both tasks, a randomized search was used to experiment with different TfIdfVectorizer parameter combinations, feeding the transformed data into a Naive Bayes classifier to evaluate their effectiveness. This classifier was chosen due to its speed and good results out-of-the-box to facilitate these experiments.

3.2. Approach 2: Distributional semantics word embeddings

In this approach, as previously stated we use pre-trained distributional semantics word embeddings (which we will refer to as WE) to represent the windows of writings. Word embeddings

[12] https://github.com/scikit-learn/scikit-learn

consist of models that map tokens to numerical vector representations based on their distributions in the corpus at the learning stage, these tend to perform better than BoW methods since the models can capture the meaning of the tokens (its distribution regarding other tokens, hence the distributional semantics part), whereas BoW only takes into account its distribution across the corpus and documents. Interesting properties are achieved as a result of this, for instance many types of relations can be encoded in the resulting vector space and analogy operations are made possible.

Despite this improvement, word embeddings like these still fail to take context into account, for instance the word "bank" in "by the river bank" and "got a loan from the bank" would have the same vector representation, even though they are referring to 2 different entities.

There were lots of possibilities for pre-trained models but 2 were selected, first, a 200 dimensional GloVe [13] [11] model trained on Twitter data, chosen due to the reliability of GloVe models and the similarity between the Twitter and Reddit social networks. The other one was a 300 dimensional FastText [14] model trained on data from Common Crawl[15], selected due to its versatility handling out-of-vocabulary tokens. Windows of writings were represented by the mean of its tokens embeddings.

3.3. Approach 3: Contextualized language models

Contextualized language models (LMs) leverage the power of deep learning by training large neural networks to achieve great performance in a variety of Natural Language Processing (NLP) tasks. Given the large computational costs that come with building language models from scratch, we decided to use a pre-trained model in this case.

There was a large variety of models we could use, but we selected the all-MiniLM-L6-v2[16] as the model to use when extracting the sequence embeddings using the SentenceTransformers[17] library [12]. This MiniLm model originated from the pre-trained 6 layer version of Microsoft's MiniLM-L12-H384-uncased (by keeping every second layer) [13], fine-tuned on a 1B sentence-pair dataset, where given a sentence from a pair, the model had to select its correct match from a set of random samples. MiniLM-L12-H384-uncased in turn was achieved by the process of knowledge distillation from BERT. Knowledge distillation is the process of compressing large models (also known as teachers in this context) by training smaller models (known as students) to imitate their behaviour. Instead of learning by computing loss from the golden truth labels, the student model uses the output of the teacher's layers as the true label (not necessarily the final layer, as was the case with MiniLm). It maps sentences to a 384 dimensional vector space.

This choice resulted from the fact that despite being a very lightweight model, according to the comparison table[18] in the SentenceTransformers documentation, the all-MiniLM-L6-v2 performed rather well across 14 sentence embedding tasks, another relevant factor was that a large subset of its training data derived from Reddit comments.

[13]https://nlp.stanford.edu/projects/glove/
[14]https://fasttext.cc/docs/en/english-vectors.html
[15]https://commoncrawl.org/2017/06/
[16]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[17]https://www.sbert.net
[18]https://www.sbert.net/docs/pretrained_models.html

Table 3
Top performing models on task 1 training data for each window size and textual feature type, classifying individual writing windows.

Features	Window Size	Model	F1	Sentiment Analysis
TF-IDF	1	NB	0.67	No
	3	LR	0.72	No
	5	LR	0.75	No
GloVe	1	LR	0.67	No
	3	LR	0.74	Yes
	5	SVM	0.77	Yes
MiniLm	1	SVM	0.68	No
	3	LR	0.74	No
	5	LR	0.76	Yes

4. Models

We divided the tasks into 2 parts, one regarding the classification of windows of writings, to assess whether a certain number of consecutive user writings (a window) is likely to come from a positive subject, and another which turns the results of the window classifications of a given subject, into a subject classification, which is our end goal. It is worth mentioning that we evaluated the window classification models with 5-fold cross-validation on the training set, and dedicated the validation set to finding confidence thresholds for the user classification.

4.1. Writing window classification

In order to make the performance of the different feature types somewhat comparable, the models used remained mostly the same across experiments, consisting of sklearn's implementations of Logistic Regression (LR), Naive Bayes (NB), linear Support Vector Machine (SVM), ExtraTrees and Perceptron models. It is worth noting that Naive Bayes was only used in approach 1 due to its inability to deal with negative features, present in the features of approaches 2 and 3. Also, due to the large feature space and amount of samples, both LR and SVM models were trained using stochastic gradient descent with sklearn's SGDClassifier.

Here, for each feature extraction approach we perform 5-fold cross validation while ensuring that no subject's writings are both present in the training folds and testing fold simultaneously to better imitate the end goal of the models being developed and to avoid information leakage.

Due to the sheer amount of experiments, we show only the best performing models for each feature type and window size in Tables 3 and 4. One can make some observations from these results, for instance, larger window sizes lead to better a F1, GloVe embeddings consistently outperformed those of FastText, the inclusion of SA features was not consistently good or bad, and that it seems that windows of depressed subjects seem to be harder to classify. The best performing models in terms of F1 score of each approach of window size 5 were selected.

Table 4
Top performing models on task 2 training data for each window size and textual feature type, classifying individual writing windows.

Features	Window Size	Model	F1	Sentiment Analysis
TF-IDF	1	NB	0.54	No
	3	NB	0.65	No
	5	NB	0.66	No
GloVe	1	SVM	0.53	No
	3	LR	0.66	Yes
	5	LR	0.67	Yes
MiniLm	1	LR	0.53	No
	3	SVM	0.63	Yes
	5	SVM	0.66	Yes

4.2. User Classification

At this stage, we essentially selected the best models for each approach and applied some criterion to classify the authors of the writings, since until now we were classifying windows of writings and not the authors themselves. Various choices can be made here, the NLP-UNED team [10], from which we drew inspiration for the rolling window method, used a parameter k of consecutive positive windows as a signal that the subject should be classified as positive, but other metrics such as a percentage or an absolute value of positive window classifications up to a point (since writings are retrieved chronologically) can be used.

We followed the "absolute value" approach with the threshold of 1, meaning that a single positive window is enough to classify as user as positive. Additionally, since our models were trained with a balanced dataset (as a result of the undersampling process), and it is unlikely that a real world scenario such as the one simulated by the official testing set will have a similar distribution, leading to skewed decisions, we imposed a threshold on the confidence of the classifiers. If a model is really confident that a window of writings belongs to a positive subject, we can classify the subject as positive. To put it simply, one of three situations may occur:

1. the window of writings was classified as positive and the confidence in its decision meets the required threshold.
2. the window of writings was classified as positive but the confidence in its decision does not meet the required threshold.
3. the window of writings was classified as negative.

The subject is only classified as positive in case 1, where case 2 and 3 lead to a negative classification.

The thresholds were selected for each model by measuring the F1-score achieved with different random threshold values at round 100, on our previously unseen validation set.

The event allowed up to 5 runs for each task. We decided to take advantage of this by running the best window classifier for each approach (BoW, WE and LM) as the models of the first 3 runs, and ensemble methods reliant on the same models for runs 4 and 5. The condition the model for run 4 needs to meet to classify a subject as positive, is to have any of the first 3 run's

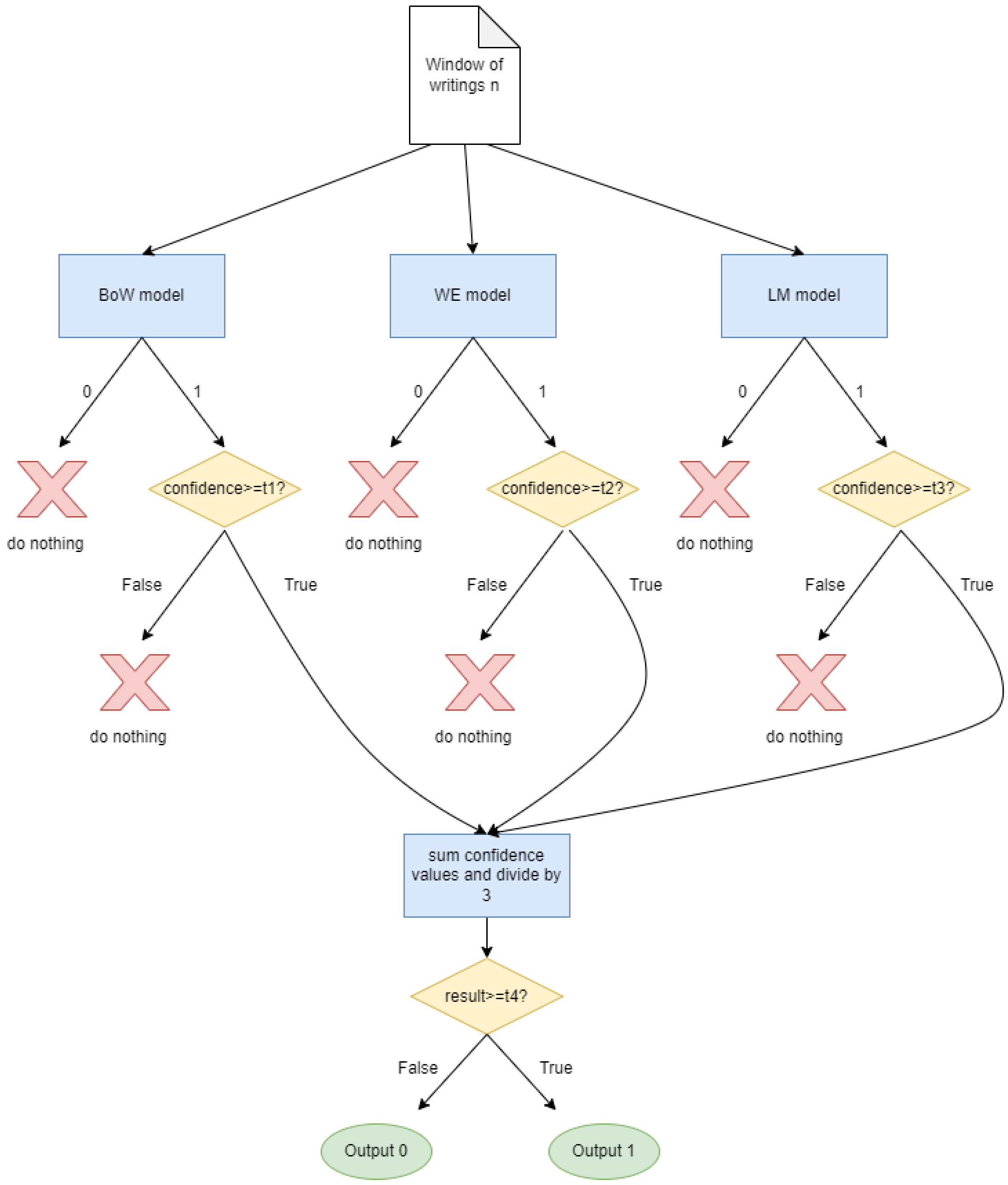

Figure 1: Illustration of the decision process followed in run 5 for a given window of writings.

classify the same user as positive. The model for run 5 sums the decision confidence of the original 3 models with a positive output (contributing 0 if their classification was negative) and divides the summed value by 3 to get an average confidence, if it passes a custom threshold, the final output is positive, it is easier to understand this logic with the illustration in Figure 1.

To put it simply, in order to output a value of 1, the condition for each run's decisions are:

Table 5
Performance of the chosen writing window models when classifying users in task 1, using the best thresholds found.

Run	Model	Features	Threshold	Precision	Recall	F1
1	LR	BoW	0.80	0.91	0.97	0.94
2	SVM	WE	0.85	0.86	0.97	0.91
3	LR	LM	0.90	0.73	1.00	0.85
4	Ensemble 1	All	None	0.61	1.00	0.76
5	Ensemble 2	All	0.80	0.97	1.00	0.99

Table 6
Performance of the chosen writing window models when classifying users in task 2, using the best thresholds found.

Run	Model	Features	Threshold	Precision	Recall	F1
1	NB	BoW	0.90	0.86	0.84	0.85
2	LR	WE	0.98	0.66	0.98	0.79
3	SVM	LM	0.98	0.81	0.91	0.86
4	Ensemble 1	All	None	0.52	1.00	0.68
5	Ensemble 2	All	0.92	0.78	0.93	0.85

1. BoW model classifies a window as 1 with a confidence higher than a threshold value.
2. WE model classifies a window as 1 with a confidence higher than a threshold value.
3. LM model classifies a window as 1 with a confidence higher than a threshold value.
4. Have any of the previous models provide a final decision of 1.
5. Sum the confidence of the first 3 run's that had a positive result and divide by 3, this averaged confidence must be higher than a threshold value.

The event's server also expects to receive a score showing how confident each model is in its decision. For this we simply send the model's estimated probability of the predicted output class at each inference. Tables 5 and 6 illustrate the thresholds and corresponding metrics on the held-out validation set following the evaluation logic that a single positive classification at any time is enough to lead to a final positive classification.

5. Results and discussion

Despite showing good results in our experiments, our approach proved ineffective in this evaluation context. When the amount of writings rises dramatically from the one used in training (roughly 1000 or 500 vs 100), it is only natural that some positive classifications might be made where they shouldn't, resulting in a high number of false positives and high recall at the expense of a low precision and F1 score since we are following a single positive user classification protocol. This is especially obvious in the results of task 2, where half the writings were processed and the results were significantly better than those of task 1, despite the opposite being observed consistently throughout the training process.

Table 7
Comparison of the performance on task 1 upon official evaluation (processing roughly 1000 posts) vs when mimicking the tuning scenario of only analyzing the first 100 writings.

Run	P	R	F1	F-Latency	P@100	R@100	F1@100	F-Latency@100
1	0.09	0.99	0.17	0.17	0.15	0.95	0.27	0.26
2	0.07	1.00	0.13	0.12	0.10	0.99	0.19	0.19
3	0.05	1.00	0.10	0.1	0.06	1.00	0.12	0.12
4	0.05	1.00	0.10	0.09	0.06	1.00	0.11	0.11
5	0.19	0.99	0.32	0.32	0.31	0.91	0.47	0.46

5.1. Task 1: Pathological gambling

The official evaluation of the server submissions showed results significantly different from those observed during the training and testing process. The left half of Table 7 displays the final precision, recall, F1 and latency-weighted F1 obtained during the event for task 1, drastically different from those previously seen in our testing, displayed in Table 5. We suspected that this discrepancy may have been caused by multiple factors:

1. The selected confidence thresholds were tuned for the first 100 windows of writings of the users in our test set, a big difference from the number of windows received from the server (roughly 1000).
2. The user classification protocol's confidence thresholds were overfitting on the limited validation data.
3. The user classification protocol chosen might just be ineffective.

Upon further inspection, using the golden truth file received post-submission, we see a clear improvement across the board on F1 scores by simply reducing the amount of writings evaluated to 100, mimicking the scenario used in the tuning process. These results, shown in the right half of Table 7, while definitely better, were still lacking, showing signs that the main issue lied elsewhere.

Since, we already established that item number 1 on our list of concerns, while easily fixable, had an impact on the results, we decided to address concern number 2. In order to do so, we changed the threshold tuning process, while before we were tuning the thresholds by using a validation subset of the training data (since the rest was used to train the writing windows classifiers), we changed the procedure to a 5-fold cross-validation (with training + validation subsets) that also includes the data left out during the undersampling process (where we retrain the full models for every fold to avoid information leakage), to better simulate a real life distribution of positive to negative samples. When tested with the official testing set, we noticed significantly better results, shown in Table 8, while still only analyzing 100 posts per subject. This improved tuning process results in higher threshold values, significantly improving the precision of our models at the cost of a worse recall, but resulting in improvements on the F1 and F-latency scores across the board, with the top performer, run 1, achieving a remarkable 0.87 F1.

Table 8
Performance on the first 100 writings of the official testing set of task 1 using the same user classification protocol but with properly tuned confidence thresholds.

Run	P@100	R@100	F1@100	F-Latency@100
1	0.94	0.81	0.87	0.86
2	0.25	0.67	0.36	0.35
3	0.32	0.89	0.47	0.46
4	0.21	0.91	0.34	0.34
5	0.96	0.56	0.70	0.67

Table 9
Comparison of the performance on task 2 upon official evaluation (processing roughly 500 posts) vs when mimicking the tuning scenario of only analyzing the first 100 writings.

Run	P	R	F1	F-Latency	P@100	R@100	F1@100	F-Latency@100
1	0.22	0.95	0.36	0.35	0.31	0.90	0.46	0.45
2	0.09	0.97	0.17	0.16	0.10	0.95	0.19	0.18
3	0.17	0.97	0.29	0.28	0.23	0.89	0.37	0.35
4	0.09	0.99	0.17	0.16	0.10	0.97	0.18	0.18
5	0.38	0.86	0.53	0.49	0.47	0.76	0.58	0.55

Table 10
Performance on the first 100 writings of the official testing set of task 2 using the same user classification protocol but with properly tuned confidence thresholds.

Run	P@100	R@100	F1@100	F-Latency@100
1	0.61	0.63	0.62	0.60
2	0.13	0.81	0.22	0.21
3	0.54	0.67	0.60	0.56
4	0.14	0.91	0.24	0.23
5	0.74	0.29	0.41	0.38

5.2. Task 2: Depression

Since we followed a similar approach in task 2, the disparity between results in testing and official evaluation was also present in this case. Although despite the worse performance in training, the combination of already stricter thresholds and reduced number of writings analyzed brought better results when compared to those of task 1.

Naturally, after seeing the improvements made in task 1's results by addressing concerns number 1 and 2, we followed the same improvement procedures in this case and also noticed significant boosts in performance. Despite achieving lower recall values, leading to undesirable false negatives, we achieve better precision values. With the run 1 being the best performer yet again, with run 3 as a close second, and run 2 having the worst performance.

Regardless of the approach, it seems that classifying depressed subjects through their social media writings is inherently harder than it is to classify pathological gamblers, the contrast in performance between BoW and LM methods (a 0.40 F1 difference in task 1 and 0.02 difference for task) 2 suggest that pathological gamblers provide cues to their condition that are more explicit, while depressed subjects tend to be more subtle, explaining why the performance of methods relying on a specific vocabulary (BoW) and more complex models (LM) had closer results in task 2.

6. Conclusions and future work

Several points regarding the window classifiers can be drawn from our experiments. First, despite the huge variation in number of textual features and the simplicity or complexity of the process of their extraction, results did not vary as much as expected across approaches in training. However, they did vary unexpectedly when using the official testing data where the top performers where at either end of the complexity spectrum, with BoW models performing the best, followed by LM models. The models relying on distributional semantics word embeddings seemed to falter even with the properly tuned confidence thresholds, indicating that perhaps our vectorization approach of averaging the tokens embeddings in each window was not the best. Perhaps the choice of a larger language model could have also resulted in better F1 values in the case of the LM approach. The inclusion of sentiment analysis in general did not have a great effect on classification, showing little to no improvement or even degrading performance in some cases.

The results seem to indicate that this singular window confidence protocol can work, but the choice of the threshold value to use is crucial and it needs to be tuned specifically to the number of writings that will be analyzed per subject.

In the future, we wish to experiment with other protocols (concern number 3), as we feel that there is a lot of room for exploration at this stage of the classification, ranging from simple conditions such as fixed absolute amount or ratio thresholds of positive classifications, to more advanced deep learning based sequence classification models that find their own criteria in the training process. We would also like to revisit approach 2 (WE), to determine the cause of its ineffectiveness in our last experiments.

References

[1] M. J. Paul, M. Dredze, Social monitoring for public health, Synthesis Lectures on Information Concepts, Retrieval, and Services 9 (2017) 1–183. URL: https://doi.org/10.2200/S00791ED1V01Y201707ICR060. doi:10.2200/S00791ED1V01Y201707ICR060.

[2] D. E. Losada, F. Crestani, A test collection for research on depression and language use, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9822 LNCS (2016) 28–39. URL: https://link.springer.com/chapter/10.1007/978-3-319-44564-9_3. doi:10.1007/978-3-319-44564-9_3.

[3] F. Sadeque, D. Xu, S. Bethard, Measuring the latency of depression detection in social media, WSDM 2018 - Proceedings of the 11th ACM International Conference on Web Search and Data Mining 2018-Febuary (2018) 495–503. doi:10.1145/3159652.3159725.

[4] J. Parapar, P. Martín-Rodilla, D. E. Losada, F. Crestani, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. 13th International Conference of the CLEF Association, CLEF 2022., Springer International Publishing, Bologna, Italy, ????

[5] J. Parapar, P. Martín-Rodilla, D. E. Losada, F. Crestani, Overview of erisk 2021: Early risk prediction on the internet, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12880 LNCS (2021) 324–344. doi:10.1007/978-3-030-85251-1_22.

[6] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference 2 (2016) 427–431. URL: https://arxiv.org/abs/1607.01759v3. doi:10.48550/arxiv.1607.01759.

[7] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, Fasttext.zip: Compressing text classification models (2016). URL: https://arxiv.org/abs/1612.03651v1. doi:10.48550/arxiv.1612.03651.

[8] D. E. Losada, F. Crestani, J. Parapar, erisk 2017: Clef lab on early risk prediction on the internet: Experimental foundations, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10456 LNCS (2017) 346–360. doi:10.1007/978-3-319-65813-1_30.

[9] D. E. Losada, F. Crestani, J. Parapar, Overview of erisk: Early risk prediction on the internet, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11018 LNCS (2018) 343–361. doi:10.1007/978-3-319-98932-7_30.

[10] E. Campillo-Ageitos, H. Fabregat, L. Araujo, J. Martinez-Romo, Nlp-uned at erisk 2021: self-harm early risk detection with tf-idf and linguistic features, in: Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, 2021.

[11] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[12] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019. URL: https://arxiv.org/abs/1908.10084.

[13] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. arXiv:2002.10957.