Sorbonne University

PLDAC PAPER

Early Risk Detection of Anorexia

Master 1 Computer Science - Data Science

UE Project 4I205

Ouarda FENEK
Nora BOUKHATA

2018 - 2019

Table of Contents

1 Introduction

The Internet has turned our existence upside down. It has revolutionized communications, to the extent that it is now our preferred medium of everyday communication. We are spending an average of 2 hours and 22 minutes per day on social networking and messaging platforms.

In almost everything we do, we use the Internet. Ordering a pizza, buying a television, sharing a moment with a friend, sending a picture over instant messaging. Before the Internet, if you wanted to keep up with the news, you had to walk down to the newsstand when it opened in the morning and buy a local edition reporting what had happened the previous day. But today a click or two is enough to read your local paper and any news source from anywhere in the world, updated up to the minute.

Unfortunately There is no such a thing as a free lunch. This huge access to the internet could be harmful.

The Cross-Language Evaluation Forum (CLEF) promotes Research and Development in multilingual information access. It proposes an eRisk task which aims at detecting early trace of risk on the Internet, especially those related to safety and health. Its main goal is to pioneer a new interdisciplinary research area that would be potentially applicable to a wide variety of situations and to many different personal profiles, such as potential pedophiles, stalkers, individuals with a latent tendency to fall into the hands of criminal organizations, people with suicidal inclinations, or people susceptible to depression.

To achieve this goal the task organizers of CLEF 2019 proposed three tasks : early risk detection of Anorexia, early risk detection of depression and Measuring the severity of the signs of depression. We participated in the first task : early risk detection of Anorexia.

2 Task

2.1 Task Description

Early risk detection of Anorexia is the first task proposed by ERisk CLEF 2019. Among this task, we explore the evaluation method, efficiency measures and practical applications especially those related to health and safety of early detection of risks on the Internet(Anorexia mostly).

This task consists of treating the writings posted by anorexic or non-anorexic users and learning to detect as early as possible the first signs of Anorexia.

To do this, the favored solution focused on text Mining and, consequently, on the different representations of these publications and comments. The texts are processed in their order of creation. In this way, systems that effectively perform this task could be applied to sequentially track user interactions in blogs, social networks, or other types of online media.

The collection of data contains textual interactions and submissions or writings (posts or comments) done by Social Media(mostly Reedit) users. There are two classes of users, anorexic and non-anorexic.
For each user,the collection contains his sequence of submissions (in chronological order) and this sequence was split into 10 chunks. The first chunk has the oldest 10% of the submissions,the second chunk has the second oldest 10%, and so forth. The **chunk** represents a sequence of writings for a given user in a period of time. In each chunk, the writings of each user are stored in an XML file named by that user's ID which has the following structure :

```
<INDIVIDUAL>
<ID> ... </ID>
<WRITING>
<TITLE> ...    </TITLE>
<DATE> ... </DATE>
<INFO> ... </INFO>
<TEXT> ...  </TEXT>
</WRITING>
<WRITING>
<TITLE> ... </TITLE>
<DATE> ... </DATE>
<INFO> ... </INFO>
<TEXT> ... </TEXT>
</WRITING>
....
</INDIVIDUAL>
```

ID : contains the anonymous id of the subject
TITLE : title of the post if available (if it is a comment then TITLE is empty)
INFO : additional info about the writing (source of the post/comment)
TEXT : body of the post or comment

2.2 Task Environment

The eRisk 2019 early detection of signs of anorexia was divided into different stages.

2.2.1 Offline Training

Initially, we had access to training data. we had the entire history of the submissions made by the different users. All pieces of all training users were considered participants. In addition, the actual class (anorexic or non-anorexic) of each user who took the training was also provided (*i.e.* whether the user explicitly stated that he or she had been diagnosed with anorexia). Here is an overview of the corpus for this task.

	Training		Test	
Number of	Anorexic	Non anorexic	Anorexic	Non anorexic
Users	20	132	41	279
Posts	2,009	21,624	2,096	35,781
Comments	7,154	61,916	16,702	124,578

2.2.2 Online Testing

For the CLEF, in this stage , we had to connect to REST API server using our token and the server has iteratively provided user writings (each request will be followed by a server response containing one writing per subject). But in this paper, we used the test data provided with the train.

2.3 Evaluation Measures

The evaluation of this task took into account standard classification measures such as $F1$, $precision$ and $recall$ and early risk detection measure $ERDE$[9].

To do this, it is important to describe the confusion matrix :

		Predicted	
		Positive	**Negative**
Actual	**Positive**	True Positive	False Negative
	Negative	False Positive	True Negative

F1-score

The $F1$ score (also $F - score$ or $F - measure$) is a measure of a test's accuracy. It considers both the $precisionp$ and the $recallr$ of the test to compute the score.

Precision is the fraction of retrieved documents that are relevant to the query, its formula is :

$$Precision = \frac{truePositive}{truePositive + falsePositive}$$

Recall is the fraction of the relevant documents that are successfully retrieved, its formula is :

$$Recall = \frac{truePositive}{truePositive + falseNegative}$$

Which gives at the end this $F1$ general formula :

$$F1 = \frac{2 * truePositive}{2 * truePositive + falseNegative + falsePositive}$$

$$= 2 * \frac{precision * recall}{precision + recall}$$

ERDE

The main goal of this task is to detect as soon as possible anorexic users. As standard classification measures (such as $F1$) do not penalize decisions delay, a new performance measure that rewards early warning was used. Specifically, $ERDE$, an error measure for early risk detection.

$ERDE$, takes into account the correctness of the binary decision and the delay taken by the system to make the decision. The delay is measured by counting the number k of distinct submissions seen before taking the decision. For instance, imagine a user u who posted a total number of 250 posts or comments ($i.e.$ exactly 25 submissions per chunk to simplify the example). If a team's system emitted a decision for the user u after the second chunk of data then the delay k would be 50 (because the system needed to see 50 evidence in order to make its decision).

This function is parameterized by $"o"$, which controls the location on the X axis where the cost increases faster and k which represents the number of distinct submissions seen before the decision. The $ERDE$ measure is defined as :

$$ERDE_o(d,k) = \begin{cases} c_{fp} & \text{if } d\text{=positive AND ground truth=negative (FP)} \\ c_{fn} & \text{if } d\text{=negative AND ground truth=positive (FN)} \\ lc_o(k) \cdot c_{tp} & \text{if } d\text{=positive AND ground truth=positive (TP)} \\ 0 & \text{if } d\text{=negative AND ground truth=negative (TN)} \end{cases}$$

Setting cfp and cfn depends on the application domain and the implications of FP and FN decisions. We will often deal with detection tasks where the number of negative cases is larger than the number of positive ones. Hence, if we want to avoid building trivial systems that always say no, we need to have $cfn \gg cfp$.

During the whole experiments we used the following values :

-cfn fixed to 1.

-cfp proportion of positive cases in the test collection, it was set to 0.1296.

-$lco(k)$ the cost associated to the delay in detecting true positives.

-$lco(k)$ ∈[0,1] , and ctp set to 1.

$$lc_o(k) = 1 - \frac{1}{1 + e^{k-o}}$$

3 Related Work

3.1 Preprocessing and Features

To clean up the data, uppercase letters were turned to lowercase, numbers were removed, and negations were joined to the following word ($example$, $don't\ want$ is transformed to a single token $don'twant$).

In addition to these features some teams have considered other lexicon features such as :

- Self-Reference : High frequency of self-reference words.
- Over generalization : Depressive users use words like : $everyone$, $everywhere$, $everything$ a lot.
- Sentiment : Use of Vader analyzer for assigning a-polarity score to users' posts : $Negative < -0.05$, $Positive > 0.05$, $Neutralotherwise.$
- Emotion : High frequency of emotionally negative words used
- Depression symptoms & related drugs.
- Past words : High frequency of past words .
- Specific verbs : High frequency of $were$ and was, $like$, $have$, $being$.
- Targeted "I" : Depressive people tend to target themselves more in subjective context especially using adjectives.
- Negative words : High frequency of negative words used. $SentiWordNet$, which is a lexical resource explicitly devised for supporting sentiment classification and opinion mining applications, was used to detect negative words in texts.
- Art-Of-Speech frequency : Higher usage of verbs and adverbs and lower usage of nouns.
- Relevant 3-grams : Higher frequency of 3-grams .
- Relevant 5-grams : Higher frequency of 5-grams .
- Relevant 1-grams : Higher frequency of 1-grams .

Some teams considered the user's behavior to analyze. For that they constructed the representation of the user taking into account certain statistics coming from the writings and the classifier used. The team who presented this approach was **UACH-INAOE (participation at eRisk 2017)**[4]. They first classified all the posts of a document Di. Then, they modeled Di as a sequence of predicted labels for each post $Pj \in Di$, that is, $Di = (y1, ..., yh)$.

After which, they represented each user by a 12-dimensional vector. The dimensions are as following :

- Percentage of posts that were classified as "depressive".
- Percentage of posts that were classified as "non-depressive".
- Percentage of times that a "non-depressive" post is followed by another"non-depressive" post.
- Percentage of times that a "non-depressive" post is followed by a "depressive" post.
- Percentage of times that a "depressive" post is followed by a "non-depressive"post.
- Percentage of times that a "depressive" post is followed by another "depressive" post.
- Percentage of "depressive" posts written in the morning.
- Percentage of "non-depressive" posts written in the morning.
- Percentage of "depressive" posts written in the evening.
- Percentage of "non-depressive" posts written in the evening.
- Percentage of "depressive" posts written in the night.
- Percentage of "non-depressive" posts written in the night.

3.2 Data Representation

Different feature engineering techniques exist in the literature of text mining. Each team have considered both raw text features and semantic features in theirs proposed methods.

3.2.1 Bag Of Words

In this model, a text such as a sentence or a document is represented as the multi-set of its words. The model creates a vocabulary of all the unique words, excluding punctuation, occurring in all the documents of the training set.

Example :

$It\ was\ the\ best\ of\ times.$
$It\ was\ the\ worst\ of\ times.$
$It\ was\ the\ age\ of\ wisdom.$
$It\ was\ the\ age\ of\ foolishness.$

First, the vocabulary will be created. For this example it would be : $[It, was, the, best, of, times, worst, age, wisdom, foolishness]$.
Then, vectors will be created counting the apparitions of each token of the vocabulary in the documents.

$It\ was\ the\ best\ of\ times$ = $[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]$
$It\ was\ the\ worst\ of\ times$ = $[1, 1, 1, 0, 1, 1, 1, 0, 0, 0]$
$It\ was\ the\ age\ of\ wisdom$ = $[1, 1, 1, 0, 1, 0, 0, 1, 1, 0]$
$It\ was\ the\ age\ of\ foolishness$ = $[1, 1, 1, 0, 1, 0, 0, 1, 0, 1]$

Each word (or token) is called a “gram”. Creating a vocabulary of two-word pairs is called a bi-gram model. The bi-grams of the first document : “It was the best of times” are : $[it\ was, was\ the, the\ best, best\ of, of\ times]$.

Following the same logic, we can create the n-$grams$ Models.
This representation is simple to understand and to implement, however, it has two drawbacks. First, the quantity of words is huge. Second, it is not feasible to calculate the relationship between words.

3.2.1.1 Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF is an information retrieval technique derived from the BoW model. It weights a term frequency (tf) and its inverse document frequency (idf). The product of the tf and idf scores of a term is called the tf-idf weight of that term.
It is used to weight a keyword in a content and assign the importance to that keyword based on the number of times it appears in the document. More importantly, it checks how relevant the keyword is throughout the corpus.

$$TFIDF(t, d, D) = tf(t, d) * idf(t, D)$$

The first part of the formula $tf(t, d)$ is simply to calculate the number of times each word appeared in each document. Of course, as with common text mining methods : stop words like a, the and punctuation marks will be removed beforehand and words will all be converted to lower cases. The second part, is described as :

$$idf(t, D) = log\frac{|D|}{1 + |d \in D : t \in d|}$$

The numerator : D is inferring to the document space. It can also be seen as $D = d1, d2, \ldots, dn$ where n is the number of documents in the collection.

The denominator : $|d \in D : t \in d|$ implies the total number of times in which term t appeared in all of document d (the $d \in D$ restricts the document to be in the current document space). Note that it does not matter if a term appeared 1 time or 100 times in a document, it will still be counted as 1, since it simply did appear in the document. As for the plus 1, it is there to avoid zero division.

The higher the numerical value of $TFIDF$ is, the rarer the term. The smaller it is, the more common the term. This method is described in **An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation**[3].
JKKTeam **TUA1 at eRisk 2018**[8] was one of the teams who applied $TFIDF$ on the writings.

3.2.1.2 Sequential Incremental Classification (SIC)

This method was applied by **UNSL's participation at eRisk 2018 Lab**[5]

During the training phase a dictionary of words is built for each category, in which frequency of each word is stored.

The team used these frequencies and a valuation function called gv to calculate the degree of confidence with which a word w is believed to belong to a category c, thus, gv takes a word and a category as inputs.

Suppose the given categories are :

$$C = \{food, music, health, sports\}$$

We could have :

$$\begin{array}{ll} gv(`sushi', food) = 0.85; & gv(`the', food) = 0; \\ gv(`sushi', music) = 0.09; & gv(`the', music) = 0; \\ gv(`sushi', health) = 0.50; & gv(`the', health) = 0; \\ gv(`sushi', sports) = 0.02; & gv(`the', sports) = 0; \end{array}$$

And then :

$$gv(`sushi`) = (0.85, 0.09, 0.5, 0.02);\ gv(`the`) = (0, 0, 0, 0);$$

Classification is finally carried out, for each subject, by means of the cumulative sum of all words :

$$\overrightarrow{d} = \sum_{w \in S} \overrightarrow{gv}(w)$$

3.2.2 Statistical Semantic Analyses

3.2.2.1 Latent semantic analysis (LSA)

This approach helps overcoming the BoW representation drawbacks. In order to construct a semantic space for a language, LSA first casts a large representative text corpus into a rectangular matrix of words by documents, each cell containing a transform of the number of times that a given word appears in a given document. The matrix is then decomposed in such a way that every document is represented as a vector whose value is the sum of vectors standing for its component words.

Similarities between words and words, documents and words, and of documents to documents are then computed as dot products, cosines or other vector-algebraic metrics.

Dimension Reduction
A reduced-rank singular value decomposition (SVD) is performed on the matrix, in which the k largest singular values are retained, and the remainder set to 0. The resulting representation is the best k-dimensional approximation to the original matrix in the least-squares sense. Each document and term is now represented as a k-dimensional vector in the space derived by the SVD. In most applications the k dimension is much smaller than the number of terms in the term-passage matrix. For most language simulations $50 < k < 1000$ dimensions are optimal, with $300 + / - 50$ most often best, although there is neither theory nor method to predict the optimum.

$$\begin{array}{cc}
 & \mathbf{d}_j \\
 & \downarrow \\
\mathbf{t}_i^T \rightarrow & \begin{bmatrix}
x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n}
\end{bmatrix}
\end{array}$$

Mathematics Formula For Reduction
Let X be the term-document matrix where x_{ij} describes the occurrence of term i in document j.
A row in this matrix will be a vector corresponding to a term, giving its relation to each document.
A column in this matrix will be a vector corresponding to a document, giving its relation to each term.
The dot product between two term vectors $t_i Transp.t_p$ gives the correlation between the terms over the set of documents.
The matrix product $X.XTransp$ contains all these dot products.

$$\begin{array}{cccccccc} & X & & U & & \Sigma & & V^T \\ & (\mathbf{d}_j) & & & & & & (\hat{\mathbf{d}}_j) \\ & \downarrow & & & & & & \downarrow \\ (\mathbf{t}_i^T) \to & \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n} \end{bmatrix} & = \quad (\hat{\mathbf{t}}_i^T) \to & \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} \dots \begin{bmatrix} \\ \mathbf{u}_l \\ \\ \end{bmatrix} \end{bmatrix} & \cdot & \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} & \cdot & \begin{bmatrix} [& \mathbf{v}_1 &] \\ & \vdots & \\ [& \mathbf{v}_l &] \end{bmatrix} \end{array}$$

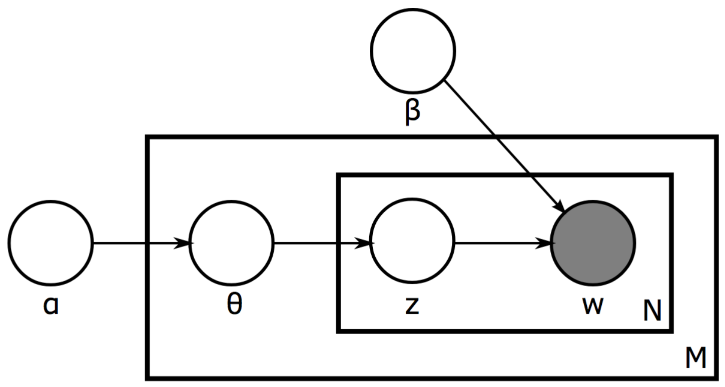
According to linear algebra, there exists a decomposition of X, $X = U.Z.VTransp$ such that U and V are orthogonal matrices and Z is a diagonal matrix.

The values z_i are called singular values, u_i and v_i are the left and right singular vectors.

At this step, we can see how related documents and terms are in the low-dimensional space typically by cosine or euclidean similarity. **TUA1 at eRisk 2018**[8] used this method. More details are given in [7].

3.2.2.2 Latent Dirichlet Allocation (LDA)

LDA is a 3-layer hierarchical model : each document is modeled by a mixture of topics that then generates each word in the document.



For example, if the observations (β) are the words collected in a textual document (M), the LDA assumes that each document M is a mixture(θ) of a small number of subjects or themes (α topics), and that the creation of each word w is attributable (probabilities) to one of the themes t of the document.

Specifically :
- A corpus is a collection of M documents, $M = (d1, ..., dD)$.
- The variables z_{dn} represent the topic chosen for the word w_{dn}.
- The parameters θ_d represent the topics distribution of the document d.
- α and n define the prior distributions on θ and β respectively, where β_k describes the distribution of the topic k.

The parameters of the distributions can be estimated by the Expectation–Maximization (EM) algorithm.

The generative process followed by LDA for a document w_d is the following :
- Choose $\theta \sim$ Dirichlet (α).
- For each word w_{dn} in all the document d words w_d :

. Choose a topic $z_n \sim$ Multinomial (θ)
. Choose a word $w_n \sim$ Multinomial (βk), with $k = z_n$.

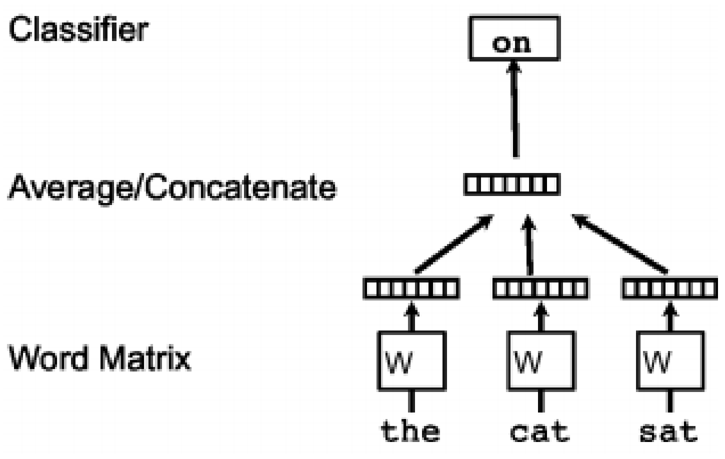
This approach is implemented with CNN classifier in the paper **Using Topic Extraction on Social Media Content for the Early Detection of Depression**[10].

3.2.3 Distribution Representation

The idea behind word embedding is to capture with them as much semantic, morphological, contextual and hierarchical content.

3.2.3.1 Word Embedding WORD2VEC

$WORD2VEC$ is a well known concept, used to generate representation vectors out of words.
It is a representation created using 2 algorithms : Continuous Bag-of-Words model $CBoW$ and the $Skip\text{-}Gram$ model.
$CBoW$ creates a sliding window around the current word to predict it from the surrounding words ($the\ context$). Each word is represented as a feature vector. After training, these vectors become the word vectors.



As said before, vectors which represent similar words are close by different distance metrics, and additionally encapsulate numeric relations, such as the $king - queen = man$ from above.

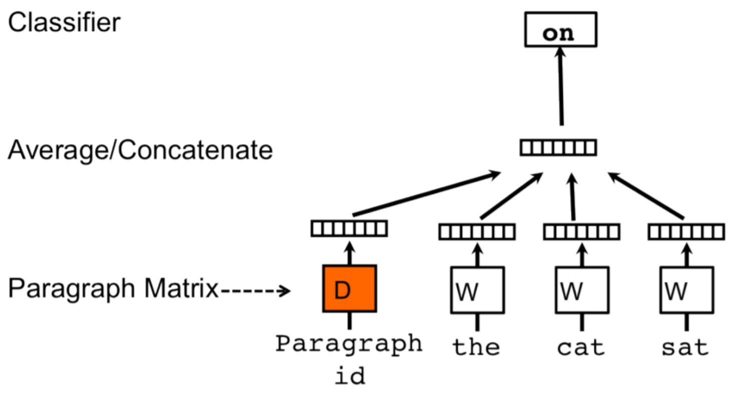
The second algorithm $Skip\ gram$ is actually the opposite of $CBoW$: instead of predicting one word each time, it uses 1 word to predict all surrounding words ($the\ context$). $Skip\ gram$ is much slower than $CBoW$, but considered more accurate with infrequent words.

3.2.3.2 Word Embedding DOC2VEC

$DOC2VEC$ is supposed to be an extension to $WORD2VEC$ such that $WORD2VEC$ learns to project words into a latent d-dimensional space whereas $DOC2VEC$ aims at learning how to project a document into a latent d-dimensional space.
The basic idea behind $DOC2VEC$ is inspired from $WORD2VEC$. In the $CBoW$ model of $WORD2VEC$, the model learns to predict a center word based on the context. For example, given a sentence "The cat sat on sofa", $CBoW$ model would learn to predict the word sat given the context words the, cat,

on and $sofa$. Its central idea is to find similarities. It randomly samples consecutive words from a paragraph and predicts a center word from the randomly sampled set by taking as input the context words and paragraph id.

Matrix D has the embedding for $seen$ paragraphs ($i.e.$ arbitrary length documents), the same way $WORD2VEC$ models learns embedding for words. For $unseen$ paragraphs, the model is again ran through gradient descent to infer a document vector.



In the given model, we see Paragraph Matrix, Average/Concatenate and Classifier sections. Paragraph matrix is the matrix where each column represents the vector of a paragraph. By Average/Concatenate, it means whether the word vectors and paragraph vector are averaged or concatenated. Lastly, the Classifier part takes the hidden layer vector (the one that was concatenated/averaged) as input and predicts the center word.
Embeddings were used in several papers which are **Word Embeddings and Linguistic Metadata at the CLEF 2018 Tasks for Early Detection of Depression and Anorexia**[15] and **Temporal Mood Variation : at the CLEF eRisk-2018**[13]

3.3 Predictive Models

The teams who participated to this task have adapted different approaches. Some of them implemented non sequential models which observe the user's entire history at once. Others, considered each of the user's post at a time. There were also some teams who combined the approaches.

Their classifiers were trained to answer $depressed$ or $wait$. If the classifier predicts a user as $depressed$ in the first $n\%$ of the history then this answer is definitive. On the final round, if the classifier returns $wait$ then the user is $not\ depressed$.

3.3.1 Classical Machine Learning Models

3.3.1.1 Naive Bayes

It is a conditional probability model, given a problem instance to be classified, represented by a vector $x = (x1, \ldots, xn)$ representing some n features (independent variables), it assigns to this instance probabilities. Parameter estimation for naive Bayes models uses the method of maximum of likelihood.

UACH-INAOE participation at eRisk 2017 [2] team trained a Naive Bayes classifier.
They defined a list $D = (D1, y1), ..., (Dn, yn)$

where :

- $y_i \in depressed, non - depressed.$
- $Di = \{(P1, y_i), ..., (Pn, y_i)\}$be the set of posts from user Di.

Then this vector P is given to the classifier .

3.3.1.2 Decision Tree

It creates a model that predicts the value of a target variable based on several input variables.
LIDIC - UNSL Research Groups participation at eRisk 2017 : Pilot task on Early Detection of Depression Notebook for eRisk at CLEF 2017 team used a decision tree (Weka's J48) obtained by first selecting, using $TFIDF$, the 100 words with the highest information gain and then removing from that list the rest . This algorithm was used also to assist to the TVT method in the initial chunks.
Detecting Early Risk of Depression from Social Media User-generated Content[2] also used it.

3.3.1.3 Random Forest

It creates a forest and makes it somehow random, builds multiple decision trees and merges them together to get a more accurate and stable prediction.

IRIT at e-Risk[14] team has performed with the Random Forest learning algorithm, for that this team calculated each feature as follows :
a) They extracted the considered lexicon words from each user's post.
b) For a given user, each lexicon word is weighted by the normalized word frequency (division of the frequency by the total number of words in the user's posts).
c) They then created one feature by averaging the obtained weights over the lexicon words.

Features were calculated for each user as follows : they first calculated the value of the feature for each of its posts or comments, and then averaged it on its posts in the chunk. when multiple chunks were used, they averaged the characteristic values obtained for each chunk for the considered user. The result (frequency of the feature) was stored in a vector which was given in entrance to the random forest algorithm.

3.3.1.4 Support Vector Machine

SVM is a non-sequential machine learning technique. For SVM models that have been used, the feature vectors needed to summarize the full user history. they converted post-level gross counting features to user-level ratio features (for example, converting the number of times anorexia was used in each publication to the proportion of all words in all of a user's posts that have been posted).

they used two out-of-the-box implementations of support vector machines :
−**Weka** implemented the sequential minimal optimization algorithm for training support vector classifiers. The model has been defined to produce probability estimates and it normalizes all default attributes. The other parameters have been set to their default values. they used a polynomial kernel of degree 1 and a cache size of 250007, which allowed better performance during preliminary experiments on the training data.

–**LibSVM** implements support vector machines using the C-support vector classification. In addition to setting the model for probability estimation outputs, they used the default settings. they used the core of the radial base function for this one because it worked best during preliminary experiments on the learning data. Here we had team **PEIMEX at eRisk 2018**[11] who used this classifier.

3.3.2 Neural Approaches

3.3.2.1 Convolutional Neural Networks (CNN)

A CNN consists of at least three layers of nodes : an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. It uses a supervised learning technique called back-propagation for training. Its multiple layers and non-linear activation distinguish it from a linear perceptron. It can distinguish data that is not linearly separable.

There has been several teams who adapted this learning algorithm such as the team from **National Taiwan University**[6]. For that ,they computed the $TFIDF$ for each word in each chunk then they selected the top 300 words with the highest $TFIDF$ score and rejected the rest ,with this 300 words they defined a vector which they used to convert each word to a unique range between 1 and 300. the contents in $title$ and in $text$ from a $writing$ were concatenated as a sequence of words. They discard the words other than the top 300 keywords. The rest of the sequence was trained to encode as a vector by using the CNN-based sentence encoder.

They performed the CNN classifier to predict every post/comment in a chunk of a use, they fixed three thresholds $\theta 1$, $\theta 2$, and $\theta 3$ which were real values. At first they performed the CNN so they could emit a risk on a user if more than $\theta 1$ of the writings are labeled as positive, and they emit negative when more then $\theta 2$ of writings are labeled as negative, else they do not emit on this user until the last chunk ,where they emit a risk if more then $\theta 3$ of writings are labeled as positive.
Also team **TUA1 at eRisk 2018**[8] used this method.

3.3.3 Temporal Approaches

3.3.3.1 Time Series

The main aim of time series modeling is to carefully collect and rigorously study the past observations of a time series to develop an appropriate model which describes the inherent structure of the series.

ARMA

An $ARMA(p,q)$ model is a combination of $AR(p)$ and $MA(q)$ models and is suitable for uni-variate time series modeling.
An $AR(p)$ model the future value of a variable is assumed to be a linear combination of p past observations and a random error together with a constant term.Generally, the random shocks are assumed to follow the typical normal distribution. Mathematically the $AR(p)$ model can be expressed as [12, 23] :

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \varepsilon_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \ldots\ldots\ldots\ldots\ldots + \varphi_p y_{t-p} + \varepsilon_t$$

Here t y and tϵ are respectively the actual value and random error (or random shock) at time period t and $\phi(1, 2, ..., p)$ are model parameters and c is a constant.

Just as an $AR(p)$ model regress against past values of the series, an $MA(q)$ model uses past errors as the explanatory variables. Conceptually a moving average model is a linear regression of the current observation of the time series against the random shocks of one or more prior observations. Fitting an MA model to a time series is more complicated than fitting an AR model because in the former one the random error terms are not fore-see-able.
The $MA(q)$ model is given by [12, 21, 23] :

$$y_t = \mu + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t = \mu + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots\dots\dots\dots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Here μ is the mean of the series, t and $\theta(1, 2, ..., p)$ are the model parameters and q is the order of the model.
In the paper **Temporal Mood Variation : at the CLEF eRisk-2018** [13], Moving average was used combined with Bayesian inference.

ARIMA
An $ARIMA$ model is a class of statistical models for analyzing and forecasting time series data. It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skillful time series forecasts[1].
AR : Autoregression.
I : Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
MA : Moving Average.
Each of these components are explicitly specified in the model as a parameter. A standard notation is used of $ARIMA(p, d, q)$ where the parameters are substituted with integer values to quickly indicate the specific $ARIMA$ model being used.
The parameters of the $ARIMA$ model are defined as follows :
p : The number of lag observations included in the model, also called the lag order.
d : The number of times that the raw observations are differenced, also called the degree of differencing.
q : The size of the moving average window, also called the order of moving average.

3.3.3.2 Threshold function

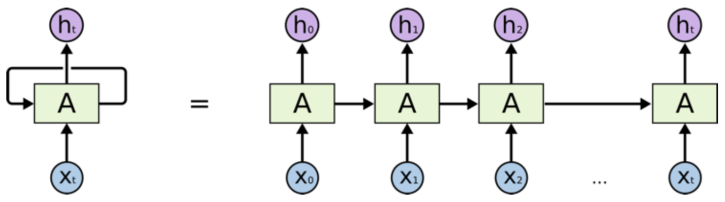
As the problem is not an ordinary classification problem due to the delay that should be taken in consideration, most teams considered a threshold function that decreases according to the number of chunks processed. Comparing its value to the results of the classifiers, the system decides to give a decision or wait. In **IRIT at e-Risk 2018**[14], The evolution of the decision threshold runs according to the considered chunk was set to :

Chunk									
1	2	3	4	5	6	7	8	9	10
0.95	0.95	0.95	0.9	0.9	0.8	0.5	0.5	0.5	0.5

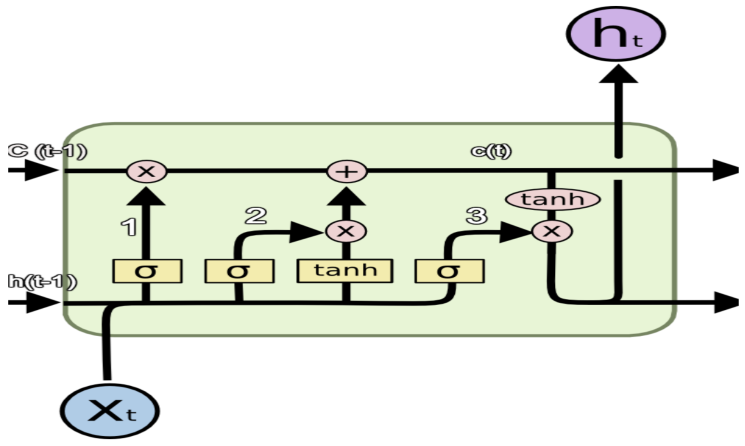
3.3.4 Temporal and Neural Approaches

3.3.4.1 Recurrent Neural Networks (RNN) / Long Short-Term Memory (LSTM)

In a traditional neural networks we assume that all inputs (and outputs) are independent of each other. But for many tasks that's rarely the case. If you want to predict the next word in a sentence for example you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. A typical RNN looks like :



In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps because of the exploding and vanishing gradient problems that can be encountered. LSTMs were developed to deal with this, they are an improvement of RNNs. They are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.



The symbols used have the following meanings :
- x : Scaling of information
- + : Adding information
- σ : Sigmoid layer
- $tanh$: $tanh$ layer
- h_{t-1} : Output of last LSTM unit
- c_{t-1} : Memory from last LSTM unit
- X_t : Current input
- c_t : New updated memory
- h_t : Current output

TUA1 at eRisk 2018[8] team has implemented TF-IDF + embedding + LSTM .

3.4 Results

Here are some of the results obtained applying different methods from those presented above.

Some teams calculated them for each chunk and others gave the result of the combination of all the chunks.

Teams (Paper Title)	Representation	Classifier	P	R	F1
Early Detection of Signs of Anorexia andDepression Over Social Media using EffectiveMachine Learning Frameworks	BOW	Logistic Regression	**0.75**	**0.73**	**0.74**
		SVM	**0.72**	**0.71**	**0.72**
		Random Forest	**0.71**	**0.74**	**0.73**
	UMLS	Ada Boost	0.41	0.50	0.45
		Logistic Regression	0.46	0.43	0.37
		SVM	0.48	0.46	0.41
		Random Forest	0.46	0.43	0.4
Temporal Mood Variation: at the CLEFeRisk-2018	DOC2VEC	Bayesian Inversion+CNN	0.38	0.68	0.49
	WORD2VEC	Bayesian Inversion+RF	0.23	**0.71**	0.35
		Bayesian Inversion+Moving Average	0.29	0.52	0.37
A Neural Network Approach to Early Risk Detection of Anorexia on Social Media Text	TFIDF+ REMOVE STOP WORDS	CNN	**0.75**	0.51	0.61
	TFIDF+ SENTENCE EMBEDDING		**0.87**	**0.83**	**0.85**
	TFIDF+ Keyword Selection		**0.75**	**0.88**	**0.81**
TUA1 at eRisk 2018	TFIDF	CNN+LSTM	0.31	0.22	0.29
	TFIDF + EMBEDDING		0.25	0.28	0.27
Using Topic Extraction on Social Media Content	LDA	CNN	0.64	0.65	0.64
PEIMEX at eRisk2018	BOW	SVM	0.39	0.56	0.46
		Ada Boost	**0.75**	**0.76**	**0.75**
	TFIDF	SVM	0.37	0.51	0.43
	BOW + TFIDF		**0.61**	**0.73**	0.67

		Chunk 1			Chunk 2			Chunk 3			Chunk 4			Chunk 5		
Teams	Methods	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
UPF's Participation at the CLEF eRisk 2018:Early Risk Prediction on the Internet	Linear Regression+LIWC+unigrams	0.43	0.25	0.32	0.49	0.34	0.40	0.54	0.43	0.48	0.54	0.43	0.48	0.55	0.47	0.51
	Linear Regression+LIWC+bigrams	0.74	0.34	0.47	0.75	0.44	0.55	0.75	0.51	0.61	0.75	0.51	0.61	0.73	0.54	0.62
TUA1 at eRisk 2018	CNN+LSTM	0.20	0.44	0.27	0.18	0.56	0.28	0.17	0.63	0.27	0.17	0.63	0.27	0.17	0.66	0.27

		Chunk 6			Chunk 7			Chunk 8			Chunk 9			Chunk 10		
Teams	Methods	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
UPF's Participation at the CLEF eRisk 2018:Early Risk Prediction on the Internet	Linear Regression+LIWC+unigrams	0.55	0.48	0.51	0.55	0.49	0.52	0.53	0.49	0.51	0.54	0.51	0.52	0.56	0.54	0.55
	Linear Regression+LIWC+bigrams	0.74	0.56	0.64	0.76	0.63	0.69	0.76	0.68	0.72	0.76	0.68	0.72	0.76	0.71	0.73
TUA1 at eRisk 2018	CNN+LSTM	0.16	0.68	0.26	0.16	0.71	0.26	0.15	0.71	0.25	0.15	0.73	0.25	0.15	0.76	0.25

4 Experiments

4.1 Methodology

Our experiments were done on a data set of 152 users in the train corpus and 320 users in the test corpus.

We had a processing phase to make a numerical representation of the data, then a learning one, in witch we fed the representations to different classifiers.

We evaluated our models by calculating precision, recall, F-measure and in some cases ERDE.

4.2 Scenarios

4.2.1 Simple Binary Classification Problem

At first, we have decided to ignore the fact that the prediction has to be early. We implemented our models to answer the question "is a person anorexic or not" after gathering all of his writings.

Formally said, we have couples (X,y) where X is a text written by a user, and y the label associated to this user, y $\in$ {-1,1}.

We have also thought of taking advantage from the format of the given data. Thus, we considered different approaches to represent our X and to train our models. More detail about these approaches will be given later in this paper.

the harshness of natural language processing is due to the complexity of our language. It is very difficult to analyze the semantics of sentences and words. Thus, before applying any classifier we have to give the text a numerical and significant representation. For that, we used TF-IDF, LSA, LDA and Word Embedding ($DOC2VEC$).

We then applied some of sklearn[12] classifiers which are Linear SVM, Logistic Regression, Multi-layer Perceptron and Random Forest. We have also tried Multinomial Naive Bayes but the results were mediocre that we are not going to present them.

To evaluate our models we have considered the famous metrics : Precision, Recall and F-measure.

4.2.1.1 Train Only On Chunk t, Concatenate Submissions

As a first approach, we have decided to proceed based on the hypotheses described below.

To predict the label of a user given his writings on a chunk i, we train our model only on the chunk i of the train corpus.

We consider our vector X as the concatenation of all user's writings in this chunk. As a result, length of X is equal to the number of users.

4.2.1.2 Train On Chunks From 1 To t, Concatenate Submissions

As a second approach we assume that to predict the label of a user given his writings on a chunk i, we consider that we have all of his writings from the first chunk to the chunk i (which is logical as the chunks are chronologically ordered), we then train our model on the chunks from 1 to i of the train corpus.

We consider our vector X as the concatenation of all user's writings in this chunk. As a result, length of X is equal to the number of users.

4.2.1.3 Train Only On Chunk t, Distinguish Submissions
As a third approach, we took an hypotheses from the first approach which is : To predict the label of a user given his writings on a chunk i, we train our model only on the chunk i of the train corpus.

But we have decided to distinguish the submissions. We consider our vector X as one writing (one submission). As a result, length of X is equal to the number of users multiplied by the mean number of writings.

4.2.1.4 Train On Chunks From 1 To t, Distinguish Submissions
The fourth approach, assumes that to predict the label of a user given his writings on a chunk i, we consider that we have all of his writings from the first chunk to the chunk i (which is logical as the chunks are chronologically ordered), we then train our model on the chunks from 1 to i of the train corpus.

We consider our vector X as one writing (one submission). As a result, length of X is equal to the number of users multiplied by the mean number of writings.

4.2.2 Temporal Classification

The data set that we had was temporal as the chunks were chronologically ordered. That's why we built an approach that deals with it as time series.

4.2.2.1 Time Series with Deep Learning
At first, we considered Deep Learning. We built a $Keras\ LSTM$ model that takes in parameters an M-Dimensional Time Series. with $M \in 1...10$ as we have 10 chunks.

The initial time series that we have are textual, thus, we applied LSA_TFIDF, LDA_TFIDF and $DOC2VEC$ to build numerical time series, that we fed to our model.

We fixed the function to optimize as the mean squared error (MSE) defined as :

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} (y_i - yhat_i)^2$$

4.2.2.2 Auto-Regressive Integrated Moving Average (ARIMA)
We have implemented the $ARIMA$ model. We trained our model using some usual classifiers(logistic regression). Their resulting probability distributions were partitioned in two tables, one table contained the positive probabilities and the other one contained the negative ones. We fed these tables to the ARIMA model. It's $forecast$ method predicted the behaviour of the series until the tenth chunk.

4.2.2.3 Threshold Function
To implement this functionality, we introduced a threshold probability. At a chunk i, if the probability of belonging to the majority class is greater than the threshold then a decision is issued -1,1. Otherwise, the models returns 0 till the last chunk.

4.3 Results

4.3.1 Simple Binary Classification Problem

The tables below shows the values of the measures (Precision, Recall and F1) at each chunk applying $TFIDF$, LSA, LDA and $DOC2VEC$ representations combined with SVM, $Logistic\ Regression$, $Random\ Forest$ and CNN on each approach.

4.3.1.1 Train Only On Chunk t, Concatenate Submissions

We remind that in this first approach we train only on the current chunk in which we concatenate all the user's submissions.

Approach 1		Chunk 1			Chunk 2			Chunk 3			Chunk 4			Chunk 5		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TF_ IDF	SVM	**1.0**	0.05	0.09	**1.0**	0.17	0.29	**1.0**	0.05	0.09	**1.0**	0.12	0.22	**1.0**	0.07	0.14
	LOGISTIC REGRESSION	0.53	0.2	0.29	0.78	0.34	0.47	0.38	0.15	0.21	0.25	0.37	0.29	**1.0**	0.17	0.29
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	**1.0**	0.02	0.05	nan	0.0	nan	nan	0.0	nan
	CNN	0.44	0.29	0.35	0.56	0.37	0.44	0.45	0.56	0.5	0.36	**0.71**	**0.48**	0.86	0.59	**0.7**
TF _ IDF + LSA	SVM	0.8	0.29	0.43	0.82	0.34	0.48	0.62	0.32	**0.42**	0.7	0.34	0.46	0.75	0.29	0.42
	LOGISTIC REGRESSION	0.19	0.12	0.15	0.29	0.24	0.26	0.2	0.15	0.17	0.23	0.17	0.2	0.76	0.32	0.45
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.28	0.2	0.23	0.35	0.34	0.35	0.29	0.29	0.29	0.35	0.34	0.35	0.75	0.66	**0.7**
TF _ IDF + LDA	SVM	0.6	0.15	0.24	**1.0**	0.29	0.45	0.64	0.17	0.27	**1.0**	0.29	0.45	0.12	0.32	0.17
	LOGISTIC REGRESSION	0.53	0.41	**0.47**	0.59	**0.49**	0.53	0.42	0.37	0.39	**1.0**	0.32	**0.48**	0.13	0.29	0.18
	RANDOM FOREST	nan	0.0	nan	**1.0**	0.05	0.09	nan	0.0	nan	**1.0**	0.17	0.29	0.16	0.71	0.26
	CNN	0.64	0.34	0.44	0.69	0.44	**0.54**	0.52	0.34	0.41	0.86	0.46	0.6	0.15	**0.78**	0.26
DOC TO VEC	LOGISTIC REGRESSION	0.24	0.39	0.3	0.44	0.39	0.42	0.37	**0.49**	**0.42**	0.35	0.68	0.46	0.37	0.56	0.44
	SVM	0.29	0.46	0.36	0.56	0.37	0.44	0.41	0.41	0.41	0.4	0.51	0.45	0.48	0.56	0.52
	RANDOM FOREST	0.37	**0.68**	0.48	0.43	0.49	0.46	0.55	0.71	0.62	0.27	0.29	0.28	0.47	0.54	0.5
	CNN	**1.0**	0.15	0.26	0.19	0.07	0.11	**1.0**	0.02	0.05	0.14	0.05	0.07	**1.0**	0.1	0.18

Approach 1		Chunk 6			Chunk 7			Chunk 8			Chunk 9			Chunk 10		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TF_ IDF	SVM	**1.0**	0.1	0.18	0.88	0.17	0.29	**1.0**	0.15	0.26	**1.0**	0.12	0.22	**1.0**	0.12	0.22
	LOGISTIC REGRESSION	**1.0**	0.2	0.33	0.73	0.2	0.31	**1.0**	0.22	0.36	0.88	0.17	0.29	**1.0**	0.22	0.36
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	**1.0**	0.05	0.09
	CNN	**1.0**	0.29	0.45	0.8	0.29	0.43	0.85	0.54	**0.66**	nan	0.0	nan	0.93	0.61	**0.74**
TF _ IDF + LSA	SVM	0.86	0.46	0.6	0.74	0.34	0.47	0.86	0.46	0.6	0.54	0.34	0.42	0.83	0.61	0.7
	LOGISTIC REGRESSION	0.6	0.51	**0.55**	0.68	0.37	0.48	0.65	0.59	0.62	0.52	0.56	**0.54**	0.67	0.73	0.7
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.4	0.73	0.52	0.61	**0.61**	**0.61**	0.42	**0.83**	0.56	0.58	0.63	0.6	0.57	0.8	0.67
TF _ IDF + LDA	SVM	0.17	0.41	0.24	0.14	0.27	0.19	0.14	0.41	0.2	0.14	0.37	0.2	0.18	0.24	0.2
	LOGISTIC REGRESSION	0.16	0.32	0.22	0.13	0.32	0.18	0.14	0.39	0.21	0.15	0.32	0.2	0.16	0.2	0.17
	RANDOM FOREST	0.18	**0.88**	0.3	0.16	0.66	0.26	0.17	0.76	0.27	0.16	**0.71**	0.26	0.15	0.56	0.23
	CNN	0.18	0.85	0.3	0.16	0.8	0.26	0.17	0.76	0.27	0.16	**0.71**	0.25	0.16	**0.68**	0.26
DOC TO VEC	LOGISTIC REGRESSION	0.35	0.54	0.43	0.33	0.39	0.36	0.48	0.71	0.57	0.4	0.46	0.43	0.44	0.34	0.38
	SVM	0.38	0.59	0.46	0.32	0.39	0.35	0.47	0.63	0.54	0.45	0.41	0.43	0.41	0.32	0.36
	RANDOM FOREST	0.41	0.54	0.46	0.54	0.51	0.52	0.5	0.68	0.58	0.51	0.61	0.56	0.38	0.59	0.46
	CNN	**1.0**	0.1	0.18	**1.0**	0.12	0.22	**1.0**	0.15	0.26	**1.0**	0.17	0.29	0.89	0.2	0.32

When $TFIDF$ was applied alone on the data, the best results were reached using a CNN. With this classifier there is a compromise between precision and recall.

When $TFIDF$ was combined with LSA, all of SVM, CNN and $Logistic\ Regression$, gave good and close results. Which were better than the results of $word\ embedding$.

For this approach, combining $TFIDF$ with LDA was not interesting, for most chunks, the F-$measure$ was between 20 and 30%.

4.3.1.2 Train On Chunks From 1 To t, Concatenate Submissions

In this second approach, as described above, to predict in the current chunk, we train our model from the first chunk till the current one. User's submissions were concatenated for each one of them.

Approach 2		Chunk 1			Chunk 2			Chunk 3			Chunk 4			Chunk 5		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TF_ IDF	SVM	**1.0**	0.05	0.09	**1.0**	0.15	0.26	0.83	0.12	0.21	0.89	0.2	0.32	0.91	0.24	0.38
	LOGISTIC REGRESSION	0.53	0.2	0.29	**1.0**	0.27	0.42	0.86	0.29	0.44	0.93	0.32	0.47	0.89	0.39	0.54
	RANDOM FOREST	nan	0.0	nan	**1.0**	0.02	0.05	**1.0**	0.05	0.09	nan	0.0	nan	nan	0.0	nan
	CNN	0.44	0.29	0.35	0.91	0.49	**0.63**	0.92	0.56	0.7	0.9	0.46	0.61	0.79	0.56	0.66
TF _ IDF + LSA	SVM	0.81	0.32	0.46	0.83	0.49	0.62	0.84	0.51	0.64	0.77	0.49	0.6	0.85	0.56	0.68
	LOGISTIC REGRESSION	0.17	0.12	0.14	0.71	0.49	0.58	0.75	0.59	0.66	0.67	0.59	0.62	0.76	0.61	0.68
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.26	0.22	0.24	0.74	0.41	0.53	0.76	0.68	**0.72**	0.74	0.63	0.68	0.71	**0.78**	**0.74**
TF _ IDF + LDA	SVM	0.6	0.15	0.24	0.18	0.39	0.24	0.14	0.37	0.2	**1.0**	0.1	0.18	**1.0**	0.15	0.26
	LOGISTIC REGRESSION	0.53	0.41	**0.47**	0.15	0.27	0.2	0.14	0.37	0.2	**1.0**	0.1	0.18	**1.0**	0.15	0.26
	RANDOM FOREST	nan	0.0	nan	0.19	0.83	0.3	0.16	0.49	0.24	0.91	0.51	**0.66**	0.92	0.56	0.7
	CNN	0.64	0.34	0.44	0.16	**0.85**	0.27	0.17	**0.85**	0.29	0.93	0.34	0.5	0.92	0.56	0.7
DOC TO VEC	LOGISTIC REGRESSION	0.26	**0.51**	0.34	0.38	0.37	0.37	0.54	0.54	0.54	0.45	0.56	0.5	0.61	0.61	0.61
	SVM	0.27	0.46	0.34	0.43	0.46	0.45	0.54	0.54	0.54	0.42	0.56	0.48	0.61	0.61	0.61
	RANDOM FOREST	0.36	0.63	0.46	0.53	0.59	0.56	0.61	0.68	0.64	0.47	0.59	0.52	0.52	0.8	0.63
	CNN	**1.0**	0.12	0.22	**1.0**	0.07	0.14	**1.0**	0.12	0.22	**1.0**	0.12	0.22	**1.0**	0.12	0.22

Approach 2		Chunk 6			Chunk 7			Chunk 8			Chunk 9			Chunk 10		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TF_ IDF	SVM	0.92	0.27	0.42	0.9	0.22	0.35	0.9	0.22	0.35	0.91	0.24	0.38	0.92	0.29	0.44
	LOGISTIC REGRESSION	0.89	0.41	0.57	0.94	0.39	0.55	0.89	0.39	0.54	0.94	0.37	0.53	0.94	0.41	0.58
	RANDOM FOREST	**1.0**	0.1	0.18	**1.0**	0.05	0.09	**1.0**	0.05	0.09	nan	0.0	nan	**1.0**	0.02	0.05
	CNN	0.92	0.56	0.7	0.89	0.61	**0.72**	0.92	0.54	0.68	0.92	0.56	0.7	**0.93**	**0.63**	**0.75**
TF _ IDF + LSA	SVM	0.86	0.61	**0.71**	0.84	0.63	0.72	0.8	0.68	**0.74**	0.85	0.71	**0.77**	**0.88**	**0.71**	**0.78**
	LOGISTIC REGRESSION	0.7	0.63	0.67	0.72	0.71	0.72	0.76	0.71	0.73	0.81	0.73	0.77	**0.81**	**0.71**	**0.75**
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.66	0.66	0.66	0.67	0.68	0.67	0.7	0.73	0.71	0.69	**0.83**	0.76	**0.86**	**0.76**	**0.81**
TF _ IDF + LDA	SVM	**1.0**	0.07	0.14	**1.0**	0.05	0.09	**1.0**	0.07	0.14	**1.0**	0.07	0.14	**1.0**	0.1	0.18
	LOGISTIC REGRESSION	**1.0**	0.1	0.18	**1.0**	0.12	0.22	**1.0**	0.12	0.22	**1.0**	0.17	0.29	**1.0**	0.17	0.29
	RANDOM FOREST	**1.0**	0.22	0.36	**1.0**	0.15	0.26	**1.0**	0.12	0.22	**1.0**	0.2	0.33	**1.0**	0.15	0.26
	CNN	**1.0**	0.39	0.56	**1.0**	0.15	0.26	**1.0**	0.24	0.39	**1.0**	0.24	0.39	0.94	0.37	0.53
DOC TO VEC	LOGISTIC REGRESSION	0.64	**0.68**	0.66	0.67	**0.76**	0.71	0.6	**0.83**	0.69	0.52	0.8	0.63	**0.62**	**0.8**	**0.7**
	SVM	0.6	**0.68**	0.64	0.63	**0.76**	0.69	0.57	0.8	0.67	0.5	0.83	0.62	**0.59**	**0.78**	**0.67**
	RANDOM FOREST	0.64	**0.73**	0.68	0.55	**0.71**	0.62	0.59	0.8	0.68	0.55	0.85	0.67	**0.65**	**0.85**	**0.74**
	CNN	**1.0**	0.1	0.18	**1.0**	0.24	0.39	**1.0**	0.07	0.14	**1.0**	0.24	0.39	**1.0**	0.2	0.33

The best combinations for this approach were : $TFIDF$ with CNN, LSA_TFIDF with CNN, $Logistic\ Regression$ and SVM, $DOC2VEC$ with $Logistic\ Regression$ or SVM.

We can notice that using this approach we had very high precision and the best value of F-measure for all the implemented methods.

4.3.1.3 Train Only On Chunk t, Distinguish Submissions

In this approach we also train only on the current chunk but this time, we do not concatenate the submissions.

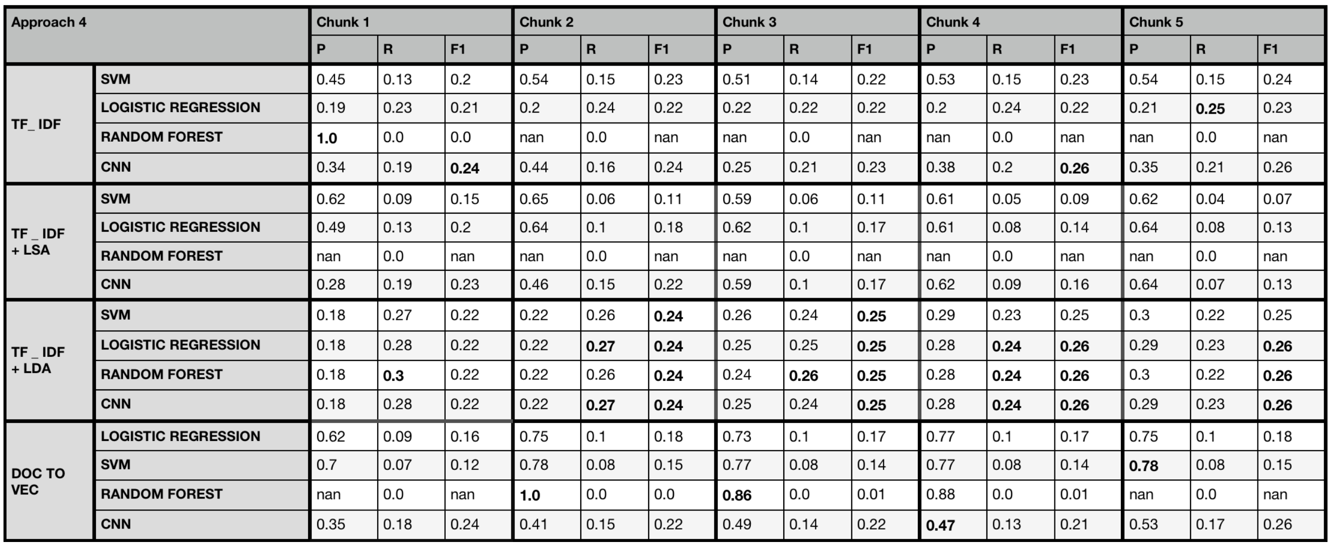
Approach 3		Chunk 1			Chunk 2			Chunk 3			Chunk 4			Chunk 5		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TF_ IDF	SVM	0.45	0.13	0.2	0.53	0.13	0.2	0.44	0.1	0.17	0.48	0.13	0.2	0.46	0.11	0.18
	LOGISTIC REGRESSION	0.19	0.23	0.21	0.19	0.23	0.21	0.17	0.19	0.18	0.18	0.21	0.19	0.18	0.22	0.2
	RANDOM FOREST	**1.0**	0.0	0.0	nan	0.0	nan	nan	0.0	nan	1.0	0.0	0.0	nan	0.0	nan
	CNN	0.34	0.19	**0.24**	0.21	0.22	0.22	0.2	0.15	0.17	0.2	0.2	0.2	0.5	0.15	**0.24**
TF _ IDF + LSA	SVM	0.62	0.08	0.14	0.7	0.06	0.11	0.6	0.05	0.09	0.63	0.04	0.07	0.61	0.03	0.07
	LOGISTIC REGRESSION	0.52	0.13	0.2	0.62	0.11	0.19	0.52	0.08	0.13	0.66	0.08	0.15	0.56	0.07	0.12
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.3	0.19	0.23	0.37	0.18	0.24	0.46	0.09	0.15	0.52	0.12	0.19	0.47	0.09	0.15
TF _ IDF + LDA	SVM	0.18	0.27	0.22	0.19	0.28	**0.23**	0.19	0.2	**0.2**	0.23	0.24	0.23	0.21	0.22	0.22
	LOGISTIC REGRESSION	0.18	**0.28**	0.22	0.19	**0.29**	**0.23**	0.19	**0.22**	**0.2**	0.23	**0.25**	**0.24**	0.21	**0.24**	0.22
	RANDOM FOREST	0.18	0.3	0.22	0.19	**0.29**	**0.23**	0.19	**0.22**	**0.2**	0.23	0.24	**0.24**	0.21	0.22	0.22
	CNN	0.18	**0.28**	0.22	0.19	**0.29**	**0.23**	0.19	**0.22**	**0.2**	0.23	0.24	0.23	0.21	**0.24**	0.22
DOC TO VEC	LOGISTIC REGRESSION	0.65	0.08	0.14	0.74	0.11	0.19	0.66	0.1	0.18	0.68	0.08	0.15	0.61	0.09	0.15
	SVM	0.71	0.06	0.11	**0.78**	0.08	0.15	**0.71**	0.07	0.13	**0.77**	0.06	0.11	**0.72**	0.06	0.11
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	**1.0**	0.01	0.01	**1.0**	0.0	0.0	**0.86**	0.0	0.01
	CNN	0.26	0.24	0.25	0.35	0.21	0.26	0.3	0.18	0.23	0.41	0.11	0.18	0.24	0.2	0.22

Approach 3		Chunk 6			Chunk 7			Chunk 8			Chunk 9			Chunk 10		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TF_ IDF	SVM	0.46	0.1	0.16	0.39	0.09	0.14	0.45	0.1	0.16	0.43	0.11	0.18	0.45	0.12	0.2
	LOGISTIC REGRESSION	0.18	**0.23**	0.2	0.18	0.19	**0.19**	0.2	**0.27**	**0.23**	0.18	**0.24**	**0.21**	0.19	**0.24**	0.21
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.3	0.17	**0.22**	nan	0.0	nan	0.25	0.19	0.22	0.18	0.2	0.19	nan	0.0	nan
TF _ IDF + LSA	SVM	0.58	0.04	0.07	0.39	0.04	0.07	0.61	0.05	0.1	0.55	0.04	0.08	0.61	0.06	0.1
	LOGISTIC REGRESSION	0.56	0.08	0.14	0.39	0.06	0.11	0.57	0.07	0.13	0.51	0.08	0.14	0.57	0.09	0.16
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.49	0.1	0.17	0.25	0.15	0.18	0.56	0.09	0.16	0.31	0.13	0.18	0.42	0.13	0.2
TF _ IDF + LDA	SVM	0.21	0.22	0.21	0.19	0.18	0.18	0.22	0.22	0.22	0.22	0.21	**0.21**	0.23	0.23	**0.23**
	LOGISTIC REGRESSION	0.21	**0.23**	**0.22**	0.18	**0.2**	**0.19**	0.22	0.23	0.22	0.21	0.21	**0.21**	0.22	**0.24**	**0.23**
	RANDOM FOREST	0.21	**0.23**	**0.22**	0.18	0.18	0.18	0.22	0.22	0.22	0.21	0.21	**0.21**	0.23	0.23	**0.23**
	CNN	0.21	**0.23**	**0.22**	0.18	0.2	**0.19**	0.22	0.23	0.22	0.21	0.21	**0.21**	0.23	**0.24**	**0.23**
DOC TO VEC	LOGISTIC REGRESSION	0.68	0.09	0.16	0.58	0.07	0.13	0.56	0.07	0.12	0.67	0.07	0.13	0.7	0.1	0.17
	SVM	**0.81**	0.07	0.13	**0.63**	0.05	0.09	**0.64**	0.05	0.09	**0.79**	0.06	0.1	0.78	0.07	0.13
	RANDOM FOREST	**1.0**	0.01	0.01	**0.78**	0.0	0.01	**0.88**	0.01	0.02	**1.0**	0.0	0.0	**1.0**	0.01	0.02
	CNN	0.36	0.12	0.18	0.31	0.16	0.21	0.34	0.14	0.2	0.35	0.15	0.21	0.29	0.15	0.2

This approach was not interesting, we didn't have high results. The best that we had didn't exceed 25% and that when we used LDA.

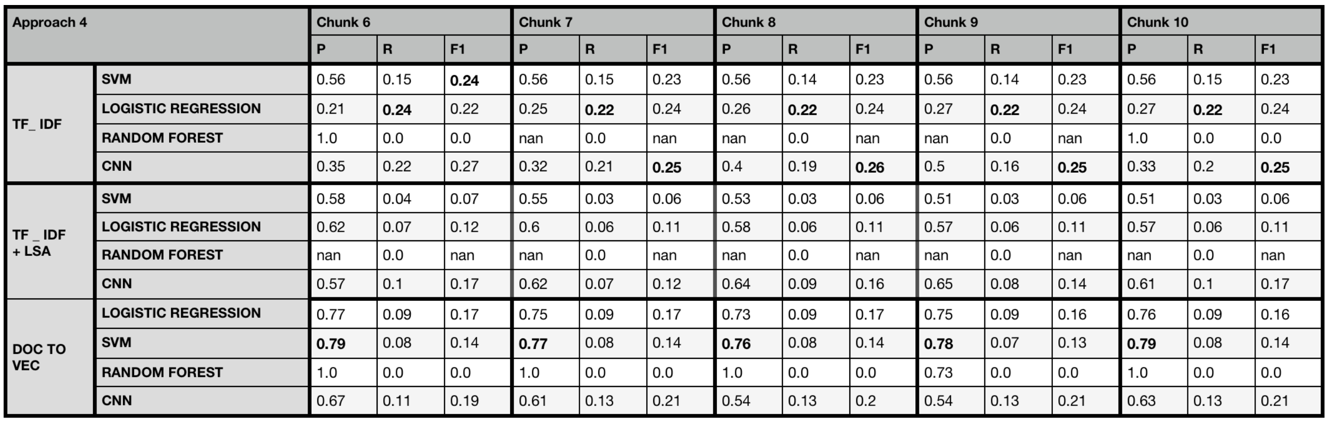
4.3.1.4 Train On Chunks From 1 To t, Distinguish Submissions

We remind that in this approach to predict in the current chunk, we train our model from the first chunk till the current one keeping the submissions as distinct.



Approach 4		Chunk 1			Chunk 2			Chunk 3			Chunk 4			Chunk 5		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TF_ IDF	SVM	0.45	0.13	0.2	0.54	0.15	0.23	0.51	0.14	0.22	0.53	0.15	0.23	0.54	0.15	0.24
	LOGISTIC REGRESSION	0.19	0.23	0.21	0.2	0.24	0.22	0.22	0.22	0.22	0.2	0.24	0.22	0.21	**0.25**	0.23
	RANDOM FOREST	**1.0**	0.0	0.0	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.34	0.19	**0.24**	0.44	0.16	0.24	0.25	0.21	0.23	0.38	0.2	**0.26**	0.35	0.21	0.26
TF _ IDF + LSA	SVM	0.62	0.09	0.15	0.65	0.06	0.11	0.59	0.06	0.11	0.61	0.05	0.09	0.62	0.04	0.07
	LOGISTIC REGRESSION	0.49	0.13	0.2	0.64	0.1	0.18	0.62	0.1	0.17	0.61	0.08	0.14	0.64	0.08	0.13
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.28	0.19	0.23	0.46	0.15	0.22	0.59	0.1	0.17	0.62	0.09	0.16	0.64	0.07	0.13
TF _ IDF + LDA	SVM	0.18	0.27	0.22	0.22	0.26	**0.24**	0.26	0.24	**0.25**	0.29	0.23	0.25	0.3	0.22	0.25
	LOGISTIC REGRESSION	0.18	0.28	0.22	0.22	**0.27**	**0.24**	0.25	0.25	**0.25**	0.28	**0.24**	**0.26**	0.29	0.23	**0.26**
	RANDOM FOREST	0.18	**0.3**	0.22	0.22	0.26	**0.24**	0.24	**0.26**	**0.25**	0.28	**0.24**	**0.26**	0.3	0.22	**0.26**
	CNN	0.18	0.28	0.22	0.22	**0.27**	**0.24**	0.25	0.24	**0.25**	0.28	**0.24**	**0.26**	0.29	0.23	**0.26**
DOC TO VEC	LOGISTIC REGRESSION	0.62	0.09	0.16	0.75	0.1	0.18	0.73	0.1	0.17	0.77	0.1	0.17	0.75	0.1	0.18
	SVM	0.7	0.07	0.12	0.78	0.08	0.15	0.77	0.08	0.14	0.77	0.08	0.14	**0.78**	0.08	0.15
	RANDOM FOREST	nan	0.0	nan	**1.0**	0.0	0.0	**0.86**	0.0	0.01	0.88	0.0	0.01	nan	0.0	nan
	CNN	0.35	0.18	0.24	0.41	0.15	0.22	0.49	0.14	0.22	**0.47**	0.13	0.21	0.53	0.17	0.26

Starting from the 6th chunk, we didn't have the results of LDA. It took a lot of time and took up a lot of memory.



Approach 4		Chunk 6			Chunk 7			Chunk 8			Chunk 9			Chunk 10		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
TF_ IDF	SVM	0.56	0.15	**0.24**	0.56	0.15	0.23	0.56	0.14	0.23	0.56	0.14	0.23	0.56	0.15	0.23
	LOGISTIC REGRESSION	0.21	**0.24**	0.22	0.25	**0.22**	0.24	0.26	**0.22**	0.24	0.27	**0.22**	0.24	0.27	**0.22**	0.24
	RANDOM FOREST	1.0	0.0	0.0	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	1.0	0.0	0.0
	CNN	0.35	0.22	0.27	0.32	0.21	**0.25**	0.4	0.19	**0.26**	0.5	0.16	**0.25**	0.33	0.2	**0.25**
TF _ IDF + LSA	SVM	0.58	0.04	0.07	0.55	0.03	0.06	0.53	0.03	0.06	0.51	0.03	0.06	0.51	0.03	0.06
	LOGISTIC REGRESSION	0.62	0.07	0.12	0.6	0.06	0.11	0.58	0.06	0.11	0.57	0.06	0.11	0.57	0.06	0.11
	RANDOM FOREST	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan	nan	0.0	nan
	CNN	0.57	0.1	0.17	0.62	0.07	0.12	0.64	0.09	0.16	0.65	0.08	0.14	0.61	0.1	0.17
DOC TO VEC	LOGISTIC REGRESSION	0.77	0.09	0.17	0.75	0.09	0.17	0.73	0.09	0.17	0.75	0.09	0.16	0.76	0.09	0.16
	SVM	**0.79**	0.08	0.14	**0.77**	0.08	0.14	**0.76**	0.08	0.14	**0.78**	0.07	0.13	**0.79**	0.08	0.14
	RANDOM FOREST	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.73	0.0	0.0	1.0	0.0	0.0
	CNN	0.67	0.11	0.19	0.61	0.13	0.21	0.54	0.13	0.2	0.54	0.13	0.21	0.63	0.13	0.21

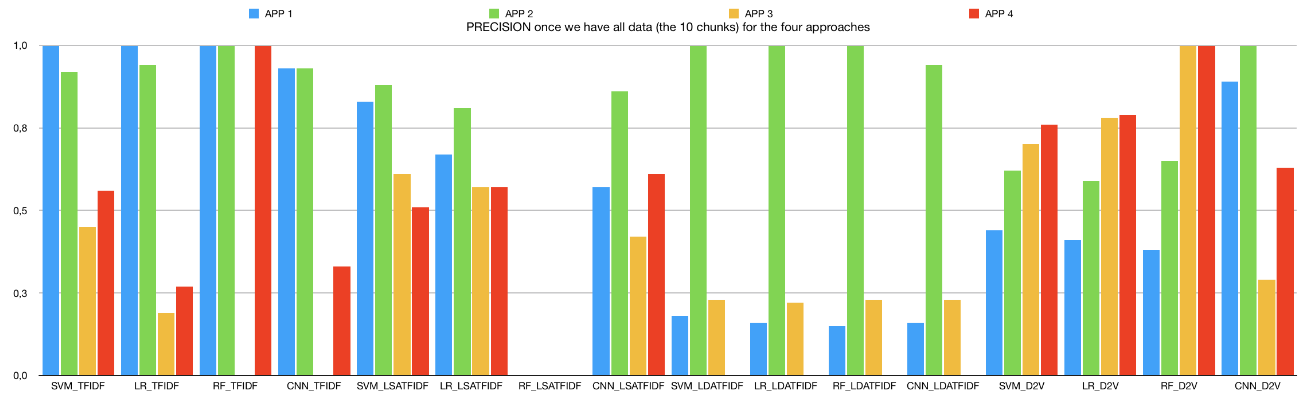
Notice that using this approach with $DOC2VEC$ representation we had high precision in all chunks. The other metrics were between 20 and 25%. They attended their highest degrees with LDA-$TFIDF$ or $TFIDF$ only.

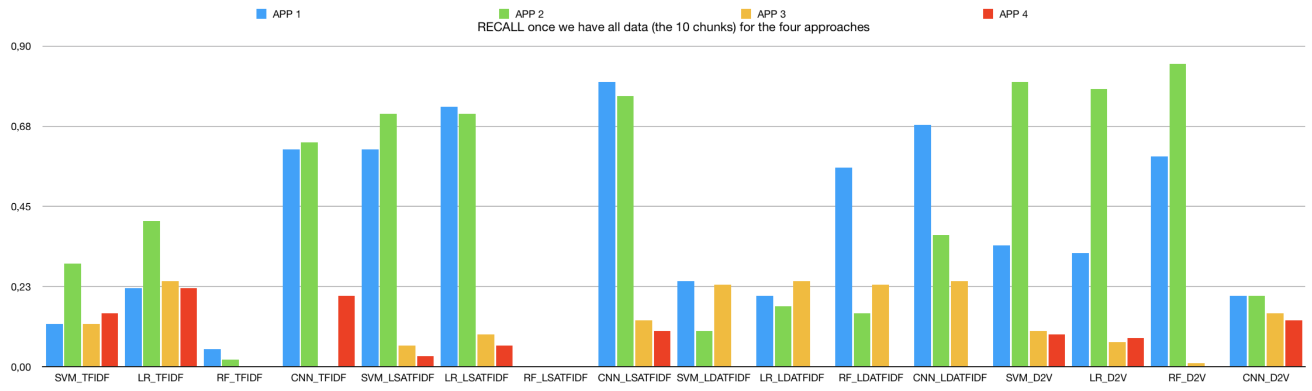
4.3.1.5 Analyses

In all the tables above, we can see that the values of the metrics are much better in advanced chunks than in the first ones. This is normal, as in the latest ones we have more information about the users and the degree of anorexia might be higher, as a result, easily detectable than at its first stage.

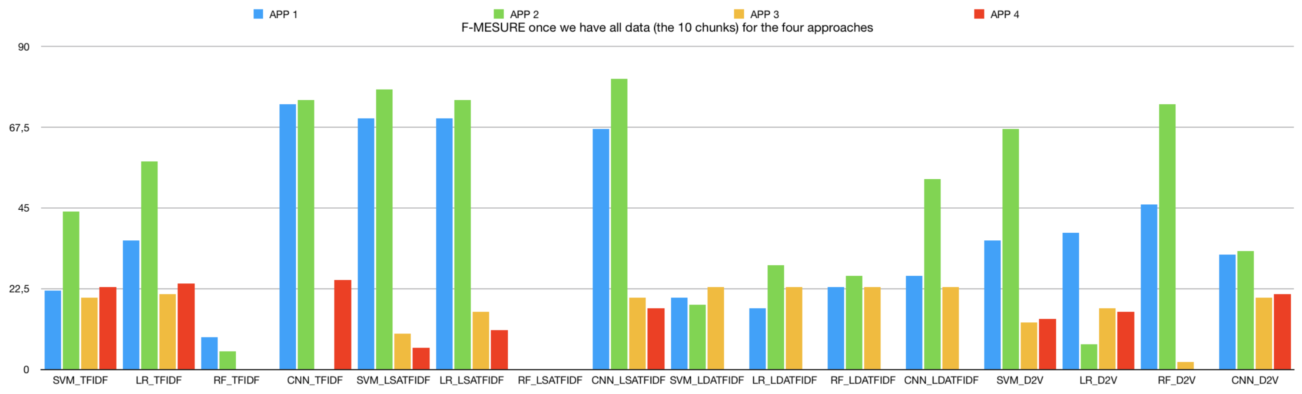
The following figures aims to compare between the approaches and the models in terms of precision and recall.

To draw them we considered that all the data was available (we had the 10 chunks).





A high F-Measure value allows for a compromise between Precision and Recall. So, a model that maximize this metric will be quite interesting for us. The figure below will help us decide which approach is the best.

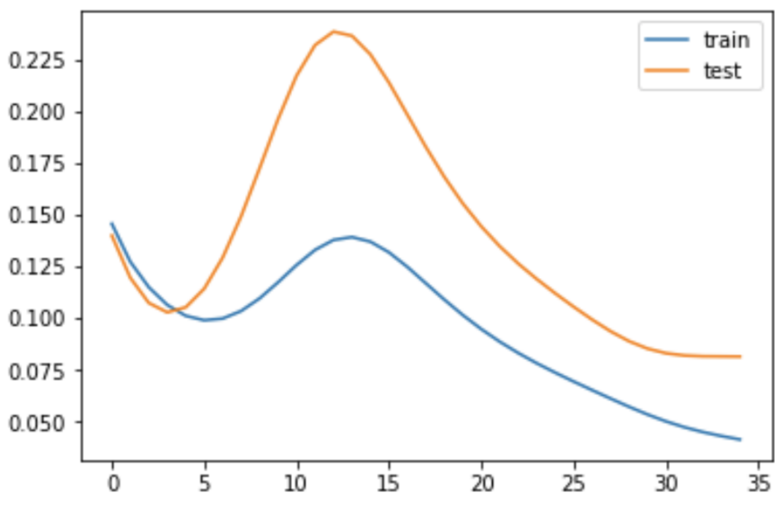


According to these observations, the best approach is the second one. Thus, we will explore it deeper in the models that follows.

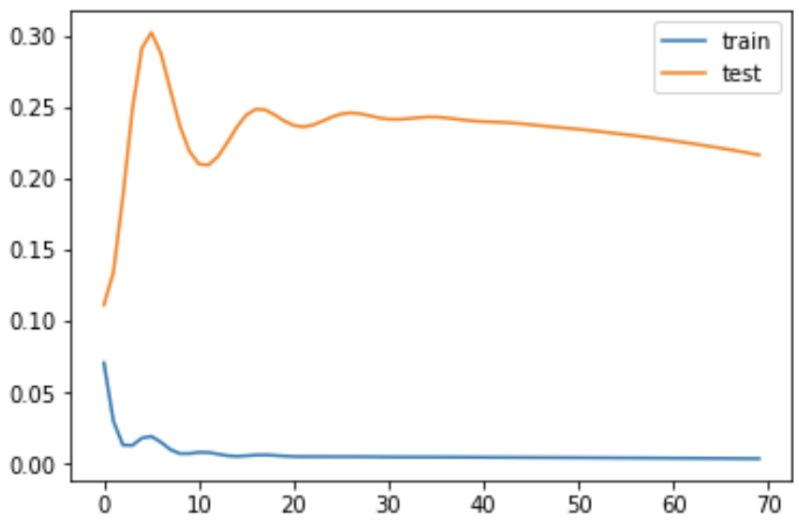
4.3.2 Temporal Classification

4.3.2.1 Time Series with Deep Learning (LSTM)

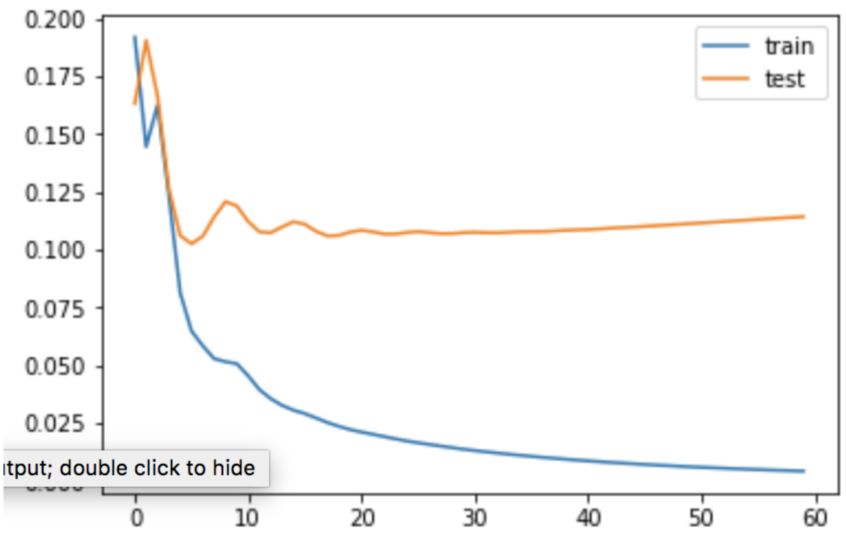
When we trained the Sequential $LSTM$ model of $Keras$ on a 10 dimensional time series expressed with the results of LSA and $TFIDF$ (with 100 component), the MSE evolution was the following :



When the same model was trained on a time series expressed with the results of Latent Dirichlet Allocation and $TFIDF$ (also with 100 component), the MSE evolution was the following :



Finally, When the time series was expressed with $DOC2VEC$ representation, the MSE evolution was the following :



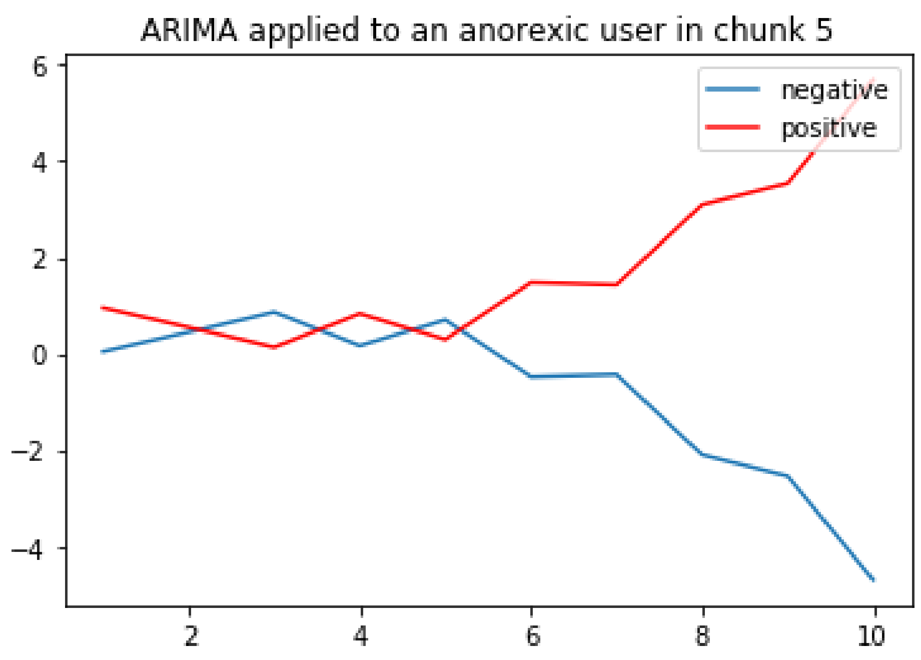
Now when we measured Precision, Recall and F1, the following table sums up the results. The chunk number matches the time series's length (the timestamp).

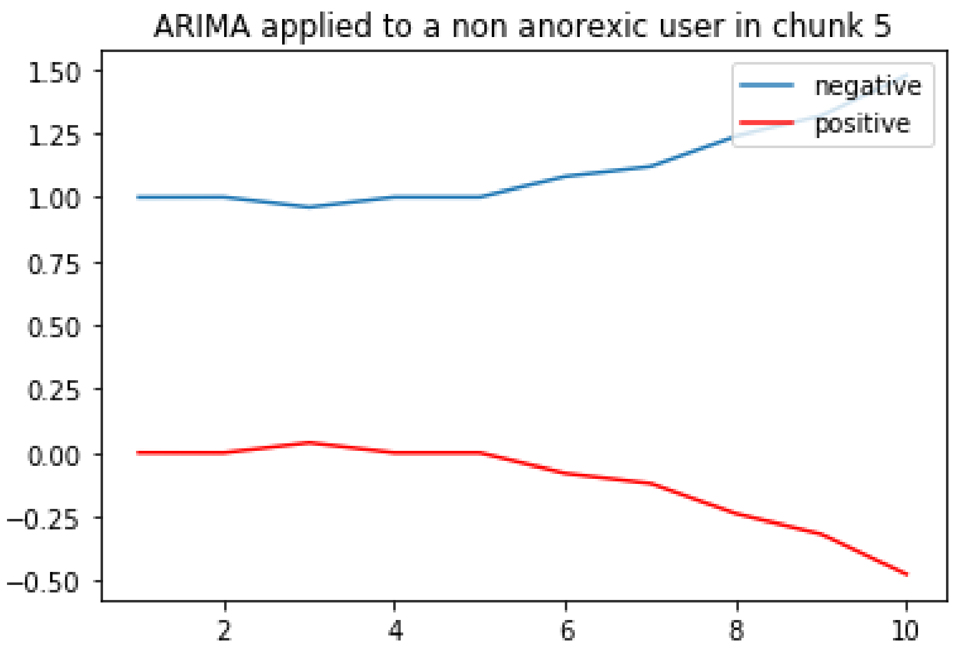
Approach 2	Chunk 1			Chunk 2			Chunk 3			Chunk 4			Chunk 5		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
LSTM / TF-IDF_LSA	**0.75**	0.29	0.42	0.55	**0.54**	0.54	**0.69**	**0.59**	**0.63**	**0.89**	0.41	0.57	**0.95**	0.44	**0.6**
LSTM / TF-IDF_LDA	0.38	**0.49**	**0.43**	**0.6**	0.44	0.51	0.59	0.39	0.47	0.66	**0.56**	**0.61**	0.71	0.49	0.58
LSTM / TF-IDF_DOCTOVEC	0.58	0.34	**0.43**	0.59	**0.54**	**0.56**	0.62	0.51	0.56	0.65	0.41	0.51	0.57	**0.59**	0.58

Approach 2	Chunk 6			Chunk 7			Chunk 8			Chunk 9			Chunk 10		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
LSTM / TF-IDF_LSA	**0.83**	0.37	0.51	**1.0**	0.32	0.48	**0.89**	0.41	0.57	**0.73**	0.27	0.39	**0.83**	0.24	0.38
LSTM / TF-IDF_LDA	0.28	**0.61**	0.39	0.24	0.59	0.34	0.2	**0.66**	0.31	0.18	0.56	0.27	0.17	0.63	0.27
LSTM / TF-IDF_DOCTOVEC	0.58	0.51	**0.55**	0.56	**0.73**	**0.63**	0.57	0.63	**0.6**	0.65	**0.63**	**0.64**	0.63	**0.71**	**0.67**

4.3.2.2 Auto-Regressive Integrated Moving Average (ARIMA)

This example shows the plotting of two users anorexic and non-anorexic to whom $ARIMA$ is applied to the fifth chunk after having first applied the logistic regression by fixing $p = 1, d = 2, q = 0$.



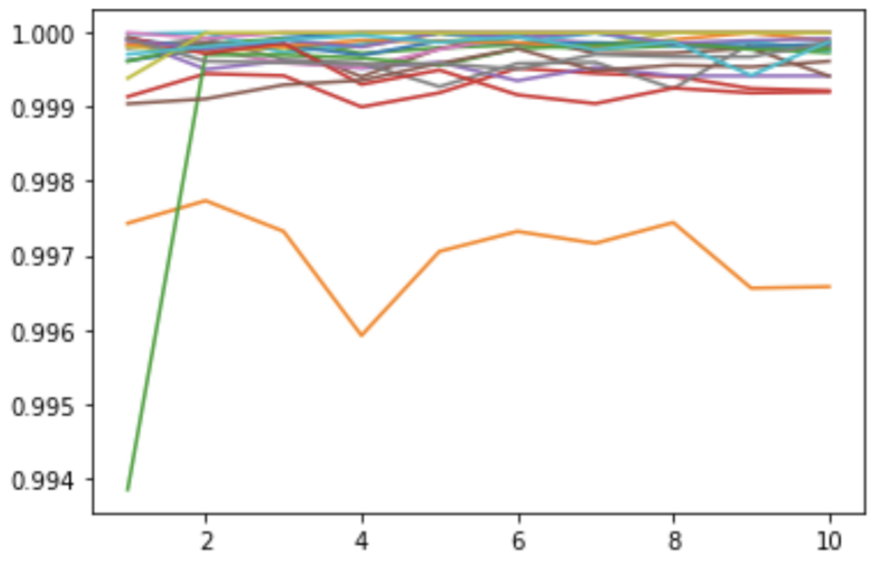


4.3.2.3 Threshold Function

Now that we have seen the comportment of our models, we want to add the early prediction functionality to them. We decided to do this on the approach which gave the results on the simple binary classification problem. It was the second approach.

We first visualized the probabilities returned by the function *predict-proba* of sklearn classifiers. Notice that not all of the classifiers have this function that's why we are not going to test them all. We will only have *Logistic Regression* and CNN. *Random Forest* has also this function but as its results were not good for the second approach we won't apply it here.

We have visualized the probabilities aimed by our models hoping that it follows some famous probability law that will help us fix the threshold but that was not the case. Here is an example of the probabilities returned by *Logistic Regression* applied on LSA_TFIDF for the anorexic users on the train corpus.



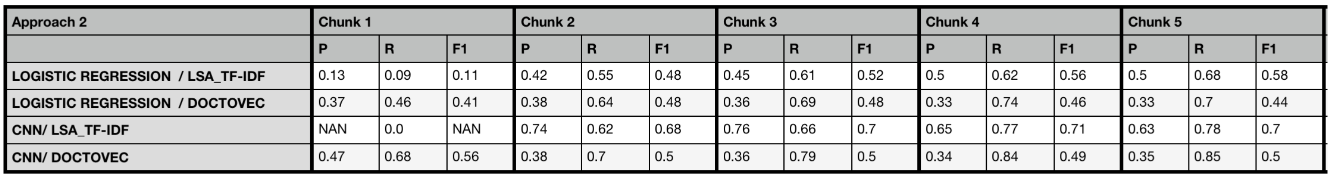
The plots were not much revealing.

We have built a model where the threshold probability equals at each chunk the mean of the probabilities of the majority class minus their variance.

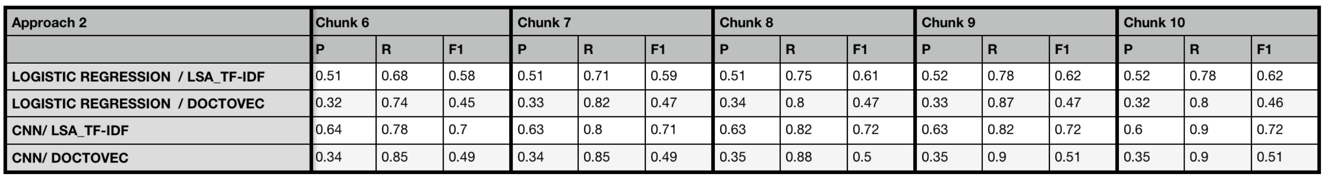
We want to specify that if at a moment 't' (a chunk 't'), a user is diagnosed as anorexic, then, this decision is kept in all the following chunks. We also kept the 't' which will be useful to evaluate the model using $ERDE$ metric.

For the users for whom the model issued a decision, we calculated precision, recall and F-measure at each chunk.

The final results are the following :

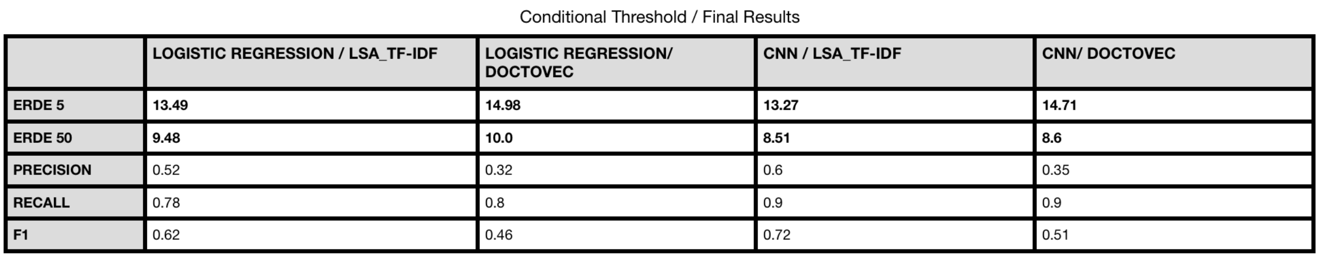


Approach 2	Chunk 1			Chunk 2			Chunk 3			Chunk 4			Chunk 5		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
LOGISTIC REGRESSION / LSA_TF-IDF	0.13	0.09	0.11	0.42	0.55	0.48	0.45	0.61	0.52	0.5	0.62	0.56	0.5	0.68	0.58
LOGISTIC REGRESSION / DOCTOVEC	0.37	0.46	0.41	0.38	0.64	0.48	0.36	0.69	0.48	0.33	0.74	0.46	0.33	0.7	0.44
CNN/ LSA_TF-IDF	NAN	0.0	NAN	0.74	0.62	0.68	0.76	0.66	0.7	0.65	0.77	0.71	0.63	0.78	0.7
CNN/ DOCTOVEC	0.47	0.68	0.56	0.38	0.7	0.5	0.36	0.79	0.5	0.34	0.84	0.49	0.35	0.85	0.5



Approach 2	Chunk 6			Chunk 7			Chunk 8			Chunk 9			Chunk 10		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
LOGISTIC REGRESSION / LSA_TF-IDF	0.51	0.68	0.58	0.51	0.71	0.59	0.51	0.75	0.61	0.52	0.78	0.62	0.52	0.78	0.62
LOGISTIC REGRESSION / DOCTOVEC	0.32	0.74	0.45	0.33	0.82	0.47	0.34	0.8	0.47	0.33	0.87	0.47	0.32	0.8	0.46
CNN/ LSA_TF-IDF	0.64	0.78	0.7	0.63	0.8	0.71	0.63	0.82	0.72	0.63	0.82	0.72	0.6	0.9	0.72
CNN/ DOCTOVEC	0.34	0.85	0.49	0.34	0.85	0.49	0.35	0.88	0.5	0.35	0.9	0.51	0.35	0.9	0.51

The final results with $ERDE5$ and $ERDE50$ are :

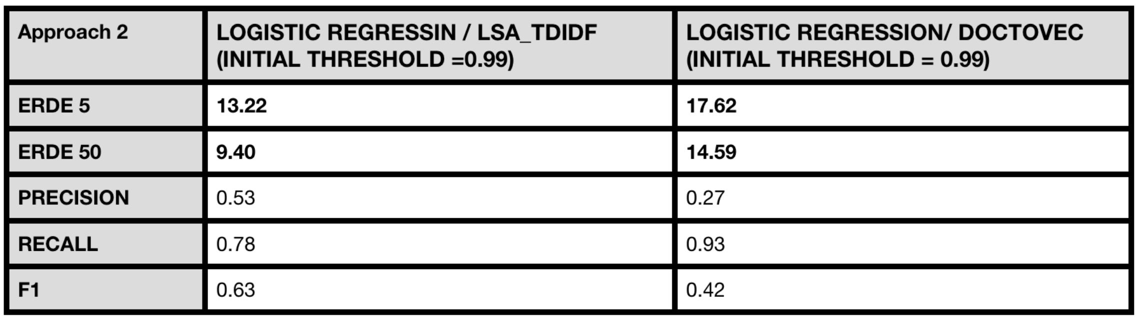


Conditional Threshold / Final Results

	LOGISTIC REGRESSION / LSA_TF-IDF	LOGISTIC REGRESSION/ DOCTOVEC	CNN / LSA_TF-IDF	CNN/ DOCTOVEC
ERDE 5	**13.49**	**14.98**	**13.27**	**14.71**
ERDE 50	**9.48**	**10.0**	**8.51**	**8.6**
PRECISION	0.52	0.32	0.6	0.35
RECALL	0.78	0.8	0.9	0.9
F1	0.62	0.46	0.72	0.51

Finally we have tried a decreasing threshold. We initialized the threshold function to 0.99 and we made it decrease with 0.02 after each chunk. When arriving to the last one we made the threshold equal to 0 in order to make sure that our model has issued a decision for each user at the end.

The results are the following :



Approach 2	LOGISTIC REGRESSIN / LSA_TDIDF (INITIAL THRESHOLD =0.99)	LOGISTIC REGRESSION/ DOCTOVEC (INITIAL THRESHOLD = 0.99)
ERDE 5	**13.22**	**17.62**
ERDE 50	**9.40**	**14.59**
PRECISION	0.53	0.27
RECALL	0.78	0.93
F1	0.63	0.42

5 Conclusion and Future Work

In this report, we have presented the multiple combinations of machine learning approaches that we have tested and those that gave good results in order to reach the main aim of the project which was to treat the writings posted by users anorexic or non-anorexic and learn to detect early signs of anorexia as soon as possible. Our results show that the $Logistic\ Regression$, SVM models and the data represented by the $second\ Approach$ with the $TFIDF$, LSA and $DOCTOVEC$ can effectively cover these task.
As future work, we want to explore and experiment more the auto-regressive methods of time series like ARIMA.

Bibliography

[1] Ratnadip Adhikari et Ramesh K Agrawal. “An introductory study on time series modeling and forecasting”. In : *arXiv preprint arXiv :1302.6613* (2013).

[2] Hayda Almeida, Antoine Briand et Marie-Jean Meurs. “Detecting Early Risk of Depression from Social Media User-generated Content.” In : *CLEF (Working Notes).* 2017.

[3] Bijoyan Das et Sarit Chakraborty. “An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation”. In : *arXiv preprint arXiv :1806.06407* (2018).

[4] Alan A Farias-Anzaldúa et al. “UACH-INAOE participation at eRisk2017”. In : *Proceedings Conference and Labs of the Evaluation Forum CLEF.* T. 1866. 2017.

[5] Dario G Funez et al. “UNSL’s participation at eRisk 2018 Lab”. In : ().

[6] James A Hanley et Barbara J McNeil. “The meaning and use of the area under a receiver operating characteristic (ROC) curve.” In : *Radiology* 143.1 (1982), p. 29–36.

[7] Jan Koeman et William Rea. “How does latent semantic analysis work ? A visualisation approach”. In : *arXiv preprint arXiv :1402.0543* (2014).

[8] Ning Liu et al. “TUA1 at eRisk 2018”. In : *TOKUASHIMA University, Telematics Research Group)* ().

[9] David E Losada, Fabio Crestani et Javier Parapar. “Overview of eRisk : Early Risk Prediction on the Internet”. In : *International Conference of the Cross-Language Evaluation Forum for European Languages.* Springer. 2018, p. 343–361.

[10] Diego Maupome et M Meurs. “Using Topic Extraction on Social Media Content for the Early Detection of Depression”. In : *CLEF (Working Notes)* 2125 ().

[11] Rosa M Ortega-Mendoza et al. “PEIMEX at eRisk2018 : Emphasizing personal information for depression and anorexia detection”. In : ().

[12] Fabian Pedregosa et al. “Scikit-learn : Machine learning in Python”. In : *Journal of machine learning research* 12.Oct (2011), p. 2825–2830.

[13] Waleed Ragheb et al. “Temporal Mood Variation : at the CLEF eRisk-2018 Tasks for Early Risk Detection on The Internet”. In : *CLEF : Conference and Labs of the Evaluation.* 2125. 2018, p. 78.

[14] Faneva Ramiandrisoa et al. “IRIT at e-Risk 2018”. In : *E-Risk workshop.* 2018, p. 367–377.

[15] Marcel Trotzek, Sven Koitka et Christoph M Friedrich. “Word Embeddings and Linguistic Metadata at the CLEF 2018 Tasks for Early Detection of Depression and Anorexia”. In : ().