Sentiment Analysis on Short Texts: The Case of the Romanian Language

Ioana-Roxana Băghiuț
Business Informatics Research Center
Faculty of Economics and Business Administration
Cluj-Napoca, Romania
baghiut.ioana@gmail.com

***Abstract*— This paper tests several methods for solving the polarity inference problem on product reviews collected from the biggest online shopping platform operating in Romania. We specifically address two scientific objectives: to build a good classifier to automatically determine the polarity of a given review, and to evaluate the maturity of existing Romanian language resources, questioning whether they could enhance the performance of various deep learning methods.**

We try the most popular document representation strategies: Bag-of-Words and word embeddings, and we test various classical Machine Learning methods like Bernoulli Naïve Bayes, Random Forest and Support Vector Machines and novel Deenovelsrning methods like Densely Connected Neural Networks, Convolutional Neural Networks, and Recurrent Neural Networks (LSTM and GRU). We also try a pre-trained word embedding developed for the Romanian language – fastText, and the Romanian DistilBERT, evaluating the degree they enhance the performance of the resulting CNN, LSTM or GRU models over our dataset.

We contribute to the scientific literature by providing a novel dataset in the Romanian language of reasonable size to be used in further NLP analyses, similar to other existing ones in English, and by evaluating the maturity of some pre-trained embeddings resources for the Romanian language.

Keywords—Sentiment Analysis, TF-IDF, Word2Vec, fastText, DistilBERT, Random Forest, SVM, Bernoulli Naïve Bayes, Deep Neural Networks, LSTM, GRU, CNN

I. Introduction

Sentiment analysis (or opinion mining), as the name suggests, is the identification of the sentiment behind a given instance and is one of the most valuable decision-making methods. It is the field of study that analyses people's opinions, feelings and emotions towards an entity (services, products, organizations, events or various topics). As current research and applications of sentiment analysis have mainly focused on written texts, it has become an active field of Natural Language Processing (NLP) research [19].

In general, sentiment analysis is a matter of semantic analysis (a meaning analysis), but it is quite limited since a sentiment analysis system does not need to fully "understand" every sentence in a document, but only the most important aspects of it, e.g., positive or negative opinions and their targets.

Sentiment analysis in reviews is the process of mining reviews of products or services on the Internet in order to determine the opinions and feelings of users. The usefulness of this type of research is endless, from business analysis to social media monitoring, brand reputation monitoring, Voice of the Consumer (VoC) listening, product analysis, etc. [1]. According to [18], 93% of customers read online reviews before buying a product. Not only people's buying decisions are heavily influenced by the opinions of others, but also the survival and profits of businesses.

The sentiment analysis process can be carried out using two approaches: rule-based and machine learning-based, or a hybrid of these two [2].

Rule-based sentiment analysis is a traditional approach where sentiments are classified based on manually created rules without training or using machine learning models. This approach includes NLP techniques such as lexicons (word lists), text cleaning, tokenization, tagging parts of speech, removing stopwords and extracting roots from words. At the end, the computer counts positive and negative words in the text and calculates the overall sentiment (a score) of the text. Different rules can be created for calculating this score, for example for the text "not good", the words are counted as opposites. The disadvantage is that this method of analysis is rather limited because it does not take the sentence as a whole, as human language is far too complex to create generally valid rules for identifying complex negations or metaphors [17].

Machine Learning-based sentiment analysis refers to the classification of sentiment based on machine learning techniques. In this case, a machine learning model is trained to classify sentiment based on the words and their order in the text. The success of this approach depends on the quality of the training set and the chosen algorithm. In this paper, we approach the machine learning-based approach for sentiment analysis, in a three-classes classification problem.

At the same time, unlike English which presents a fairly diverse suite of resources, machine learning for Romanian can be challenging because it does not include as many resources and, in particular, because we do not know whether existing resources (such as BERT-like pre-trained models and fastText pre-trained word embeddings) are mature enough.

In this paper, we perform sentiment analysis emerging from the opinions of users from the online environment. We define our problem as a three-class polarity detection (positive, neutral and negative) from reviews. For this purpose, we initially performed the automatic data extraction necessary for the sentiment analysis. Next, we apply a full NLP processing methodology consisting of text preprocessing, feature extraction and model learning. We try various alternatives of ML classifiers and test the proposed feature extraction techniques.

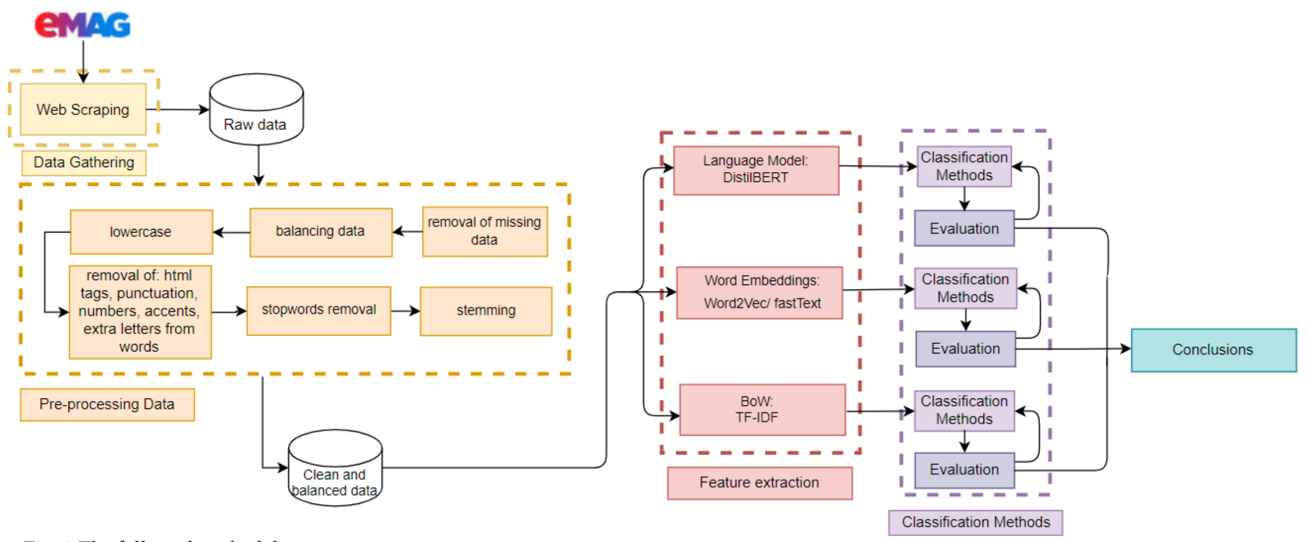


Fig. 1 The followed methodology

II. Related Works

As far as we know, there is only two other works that try to classify Romanian texts into three classes (positive, negative and neutral). [20] presents a BoW approach that does not require training, as it is based on an improved term-counting method that detects dependencies between words in a text, as well as considering the various valence shifters. The method was built on a dataset of 2521 samples (reviews extracted from a tourism website[1]), and the results showed an overall F-score of 0.83.

[21] proposes a semi-supervised method for sentiment analysis in Romanian using 2 large datasets (a 25.841 samples news dataset and a 42.497 samples Romanian Dictionary), the results varying depending on the size of the train and test sets used, the maximum F-score was 0.89.

Neither of them approached word embeddings or contextualized word embeddings. Only classical BoW were used.

III. Methodology

Figure 1 presents our overall data processing methodology. The first step is data extraction, which is done by web scraping. This step results in a raw, noisy and unbalanced dataset. In the next step, this dataset is first cleaned of any missing data, in this case only one record had missing data. Next, we balanced the dataset, then the texts (the extracted reviews) are cleaned of any unnecessary information (HTML tags, numbers, punctuation, stopwords, etc.). Finally, we normalized the words using stemming. To be able to work with text data, it is important to transform the raw text into a form that can be understood and used by machine learning algorithms, which is actually the third step, feature extraction (or text vectorization). Three techniques were used here: classical vectorization using TF-IDF, word embeddings using two modalities: Word2Vec (trained on the dataset) and fastText, and a pre-trained model on Romanian – DistilBERT. The next step includes performing experiments using the various machine learning and deep learning classification methods using the features extracted above. Each model produced is evaluated and tested in the second to last step. Based on the results the models were adjusted in order to find the most capable model to classify the reviews. At the end, based on all the results, conclusions and possibly further development directions were determined. All these steps will be presented in detail in the next sections of the paper.

IV. Dataset

A. Data Gathering

The dataset was created using data extracted in January 2022 and contains information on reviews given by eMag shoppers for their purchased phones.

Data extraction from the eMag website was achieved using web scraping. Web scraping is not a novel process, in previous years this practice was more commonly known as screen scraping, data mining, web harvesting or other similar variants [23].

The reviews are dynamically loaded in each product's page, in the sense that they are arranged in pages of 10, and when the next page of reviews is clicked, only the reviews section is reloaded, not the entire product page. We used the Scrapy[2] framework to crawl all the product pages to extract and assemble the URLs from which all the product reviews can be accessed. Next, the Requests library[3] was used to access the addresses resulting from Scrapy and extract information about the product name, score, title and body of the review.

B. Dataset Description

The extracted dataset consists of 5 attributes and a total of 41,928 reviews with a total size of 13.5 MB. The dataset's attributes are presented in Table I. Only two important columns will be used in the experiments (the ratings and the content of the reviews).

TABLE I. DATASET DESCRIPTION

[1] https://amfostacolo.ro/

[2] https://scrapy.org/

[3] https://pypi.org/project/requests/

Attribute	*Type*	*Description*
phone_id	Text	Refers to the phone's unique ID
phone_name	Text	Refers to the phone's name
review_score	Numeric	Represents the score given by the client. It takes values between 1 and 5.
Review_title	Text	Refers to the title of the review.
Review_content	Text	Refers to the textual content of the review.

V. DATA PRE-PROCESSING

Data pre-processing is the process of cleaning and preparing text for classification. Online texts usually contain a lot of noise and non-informative parts, such as HTML tags or emojis. In addition, many words in the text do not have an impact on the overall orientation of the text, and keeping them would make the classification process more difficult because each word in the text is treated as a dimension.

Preprocessing the data is as important as building any machine learning model. The reliability of a model depends largely on the quality of the data, the performance and accuracy of machine learning models being fundamentally dependent on the quality of the data used.

A. *Data Cleaning*

Before we could apply all the text cleaning steps, we first kept only the columns from the original dataset that are relevant for sentiment classification (review_score, review_content).

Moving to data visualization, the histogram on the left of Figure 2 shows that the dataset is highly unbalanced towards high ratings. Most reviews are rated 5. These are followed by reviews that scored 4, and the remaining reviews scored 3, 2 or 1. We can agree that users perceived the score 4 as the middle (neutral) class.

The aim is thus to achieve three balanced classes by grouping the reviews into the following classes: positive (reviews with the rating 5), neutral (reviews with the rating 4) and negative (reviews with the rating 3, 2 or 1).

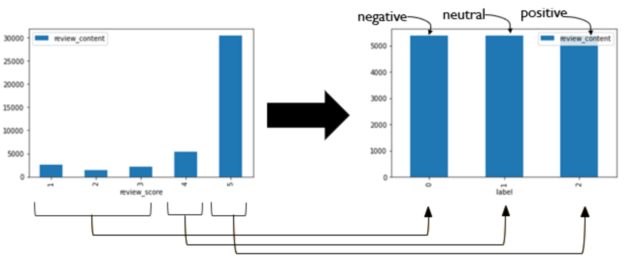


Fig. 2 Class distribution on the original data set and the balanced data set

The final set used in our experiments consists of 16.125 reviews, equally distributed across the 3 classes (5375 per class). Furthermore, in order to have as much information as possible, only the longest reviews were kept, their length varying from 1 to 1132 words. In the table below are presented the number of samples and words for each class.

TABLE II. NUMBER OF SAMPLES AND WORDS IN THE BALANCED DATASET

Class	*Samples*	*Words Before Cleaning* [a.]	*Words After Cleaning* [b.]
0	5375	396.370	190.757
1	5375	356.358	173.658
2	5375	742.095	354.295

[a.] Only html tags and brackets removal were performed

[b.] Additional data cleaning was performed, process explained below

Next, the cleaning texts process was rather a long and extensive process that included several steps such as converting texts to lowercase, removing punctuation, HTML tags, numbers and single letters, extra letters in words. Diacritics and stopwords, and finally text normalization using stemming. Such an example can be seen in the following table:

TABLE III. TEXT CLEANING EXAMPLE

Step	*Text*
Original text	'<a href=""https://www.idevice.ro/2018/05/01/huawei-p20-pro-defectul-umilitor-fabricatie/"" target=""_blank"">www.idevice.ro/2018/05/01/huawei-p20-pro-defectul-umilitor-fabricatie/</a> <br /> <br /> Eu am făcut greșeaaala să-l repar cu 1500 de lei. S-a spart a doua oară . [} %'
Lowerase	'<a href=""https://www.idevice.ro/2018/05/01/huawei-p20-pro-defectul-umilitor-fabricatie/"" target=""_blank"">www.idevice.ro/2018/05/01/huawei-p20-pro-defectul-umilitor-fabricatie/</a> <br /> <br /> eu am făcut greșeaaala să-l repar cu 1500 de lei. S-a spart a doua oară . [} %'
HTML tags removal	' eu am făcut greșeaaala să-l repar cu 1500 de lei. S-a spart a doua oară . [} % '
Punctuation removal	' eu am făcut greșeaaala să l repar cu 1500 de lei s a spart a doua oară '
Stopwords Removal	'făcut greșeaaala l repar 1500 lei s spart doua oară'
Numbers removal	'făcut greșeaaala l repar lei s spart doua oară'
Diacritics Removal[a.]	'facut greseaaala repar lei spart doua oara'
Extra letters removal	'facut greseala repar lei spart doua oara'
Stemming[a.]	'facut greseal repar lei spart dou oar'

[a.] This step was skipped for the DistilBERT case

B. *Train-Test splitting*

The data set was divided into 80% - training set and 20% - test set. Also, 15% out of the training was used as validation set.

VI. CONVERTING TEXTS TO FEATURES

Vectorization is the process of transforming text data into numerical representations so that they can be understood by machine learning algorithms. This process consists of applying a certain tokenization scheme (transformation of text into simple units, called tokens) and then associating numeric vectors with the generated tokens. These vectors are then used in training the models. Now, there are different ways of associating these numeric vectors with the generated tokens.

In this paper were used 3 different methods of vectorization:

1) Bag-of-Words – TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency) evaluates the relevance of a word to a document in a collection of documents [3]. Essentially, TF captures the importance of the word regardless the length of the document, measuring how many times a term is present in a document. Inverse Document Frequency (IDF) assigns a lower weight to frequent words and a higher weight to non-frequent words. Thus, the value of IDF will be very low for the most occurring words such as stopwords.

The downside of this method is that the vectors obtained have a very high dimensionality (the same dimensionality as the number of words in the vocabulary). Moreover, TF-IDF retains the general frequency characteristics of words in the text, but ignores the link between context and semantics of each word. Therefore, this method does not capture the meaning of words, but considers words as separate features.

Here, we used a TF-IDF vocabulary of 8.142 terms.

2) Word Embeddings

Word embeddings are a feature learning technique in which the words in the vocabulary are mapped to numeric vectors that capture the contextual hierarchy [4]. That means that words from the same context which usually appear together in the body of the text will also be close together in the vector space.

a) Word2Vec

It produces a vector space, usually several hundred dimensions in size, with each unique word in the corpus, so that words that share a common context in the corpus are located close to each other in this space.

In this paper, we trained a Word2Vec model and we used the original dataset (all 41.928 reviews) since it needs as many texts as possible in order to form connections between words. The model contains 18.983 words, each with 300 dimensions.

b) fastText

Word2Vec works at the word level, so if there is a word that does not appear in the vocabulary (OOV - Out of Vocabulary), Word2Vec fails to get a vector representation of it. This problem can be solved by fastText [7].

FastText[4] (which is basically an extension of Word2Vec), was first published and created by FAIR (Facebook’s AI research) in 2016 and it treats each word as consisting of n-grams of characters, which is also the key difference between the two models. Thus, the vector for a word is the sum of these n grams of characters. For example, the word "telefon" can be broken up as "tel", "ele", "lef", "efo", "fon" for n=3.

Facebook has published a large number of such pre-trained word vectors based on Wikipedia and Common Crawl including Romanian which can be downloaded from their website.

3) Contextualized Word Embeddings

The problem with word embeddings is that they don't see the differences between words that are spelled the same, but depending on the context a word can have different meanings.

Based on the limitations of these word embeddings, recent research has attempted to create representations of words that are context-sensitive, and in 2017 Google unveiled a new neural network architecture called "transformer", that relies on an attention mechanism to draw global dependencies between input and output [9].

BERT makes use of transformer to learn contextual relations between words in a text.

BERT (Bidirectional Encoder Representations from Transformers) was designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. Therefore, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a various categories of NLP tasks[10].

We used the DistilBERT[5] model pre-trained on the Romanian language.

The Distil-BERT model for Romanian (first published in April 2022) was obtained by distilling the knowledge of the base model BERT-ro. It uses 6 coding layers (only half as many as the base model), 12 heads of attention and a hidden dimension of 768 units [11].

VII. Classification Methods

a) Classical Machine Learning Methods

SVM – Support Vector Machines are one of the most robust and accurate methods among known Machine Learning algorithms. They are supervised machine learning algorithms that can be used for both classification and regression problems. SVM performs classification by constructing large hyper-planes, which are also called decision planes. These hyper-planes help to extract a particular type of data [12].

Random Forest – Random Forest is one of the best classical classification methods. It is an ensemble-based supervised classification algorithm that works based on the decision tree algorithm that can be applied to both regression and classification problems. The trees are constructed using two elements: nodes and branches. At each node, one of the data features is evaluated to create different subsets of the dataset used, each instance belonging to a subset. The final subsets are called terminal nodes or leaf nodes, and the intermediate subsets are called internal nodes or split nodes. To predict the result in each leaf, the results of the training data in the node are averaged [13].

Bernoulli Naïve Bayes – Naive Bayes classifiers are linear classifiers, known for being simple but efficient. The probabilistic model of these classifiers is based on Bayes' theorem.

The features are independent binary/boolean values. This model is used for classification problems, where the (binary)

[4] https://fasttext.cc/

[5] https://huggingface.co/racai/distilbert-base-romanian-cased

features show the occurrence of terms, i.e., they show whether or not a word appeared in a document [14].

b) Advanced Machine Learning Methods

LSTM – Long-short term memory is a recurrent neural network that is used to solve text classification problems. It consists of three gates (forget gate, input gate, and output gate) and a cell state that allow the information to persist in time [15].

GRU – Gated Recurrent Unit is another type of RNN. GRU works on the same principle as LSTM, but is more simplified and thus cheaper to execute (however, it may not have the same representational power). GRU no longer uses the state of a cell to transfer information, but makes use of update and reset gates [16].

CNN – Convolutional Neural Network were invented specifically for image recognition. They work by convolution, in other words, they extract local portions (patches) of data from the input, allowing for modular and efficient data representation, which makes them computationally cheaper. CNNs can be a fast alternative to RNNs for tasks such as text classification. Applied to a text sequence, the 1D convolutional layer will extract 1D slices (or subsequences) from the input sequences. Thus, applied on each batch, it comes to recognize local patterns in an input and will be able to recognize them later in a different position. In this paper, we used a 1D Convnet [6].

VIII. RESULTS AND DISCUSSIONS

A. *Experiments*

We tried the following combinations. All of the conducted experiments can be found on github[6].

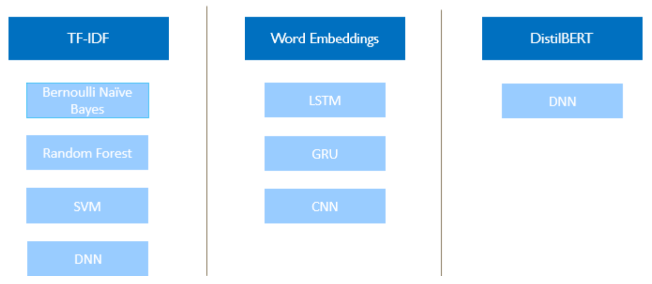


Fig. 3 Tested Methods

The classical ML classification methods were implemented using the sklearn[7] library, while the DL methods used the keras[8] library. The DistilBERT models used the transformers[9] library.

I. *TF-IDF Experiments*

We initially performed a series of experiments a dense neural network of the form: layer + output layer, where we tried several values for the number of units in the first layer and two optimizers:

TABLE IV. VALIDATION ACCURACY FOR DNN + TF-IDF

Network's capacity						
8	***16***	***32***	***64***	***128***	***256***	***512***
Using RMSProp, lr=0.001, 10 epochs:						
0.6992	0.6997	0.7008	0.6951	0.6951	0.6873	0.6848
Using Adam, lr=0.0001, 10 epochs:						
0.6956	0.7101	0.7152	0.7111	0.7044	0.7028	0.7018

Overall, the models do better when using the Adam optimizer, which also has the best accuracy of all the experiments. The performance of this model can be seen in Figure 4.

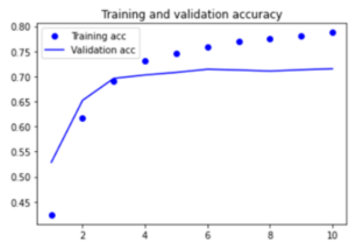


Fig. 4 Accuracy for DNN + TF-IDF

The model overfits, the accuracy on the validation set reaches the 70% threshold in epoch 3 and is maintained, while the performance on the training set continues to increase.

Next, the results on the test set for the classical ML methods and DNN are shown in the table down below:

TABLE V. TEST RESULTS FOR MODELS USING TF-IDF

Method	***Accuracy***	***F1***
SVM	73.12%	73%
BernoulliNB	68.84%	69%
Random Forest	71.72%	71%
DNN	71.81%	72%

TF-IDF showed the best results in combination with the SVM (accuracy of 73.12% and F1 of 73%), and the worst results when used with the Bernoulli model. Random Forest also managed to reach a performance very close to that of the neural network.

II. *Word Embeddings Experiments*

We tried LSTM, GRU and Convnet1D with both Word2Vec trained on the original dataset and the pre-trained fastText on the Romanian language.

Each RNN network was configured with the following structure: EmbeddingLayer + RnnLayer + OutputLayer

Each CNN network was configured with the following structure: EmbeddingLayer + ConvLayer +MaxPoolLayer + ConvLayer + GlobalMaxPool + OutputLayer.

In both RNN and CNN cases we used the RMSProp optimizer with a learning rate of 0.001 and 10 epochs.

In the case of Word2vec, the models showed the following accuracy values on the validation set:

TABLE VI. VALIDATION ACCURACY FOR DL MODELS USING WORD2VEC

Network's capacity						
8	***16***	***32***	***64***	***128***	***256***	***512***
LSTM						

[6] https://github.com/ioana0901/Sentiment-Analysis-on-Short-Texts-The-Case-of-the-Romanian-Language.git
[7] https://scikit-learn.org/stable/user_guide.html
[8] https://keras.io/guides/
[9] https://huggingface.co/docs/transformers/index

0.7276	0.7147	0.7121	0.7344	0.7416	0.7261	0.7442
GRU						
0.7339	0.7127	0.7225	0.7426	0.7297	0.7359	0.7225
Convnet1D						
0.5840	0.6801	0.6791	0.6982	0.6873	0.6873	0.6791

Both RNNs worked similarly well with Word2Vec. Convnet performed worse than the recurrent networks (which was expected), with the highest accuracy being only 69.82%.

The performance of the best LSTM and GRU models can be seen in Fig.5:

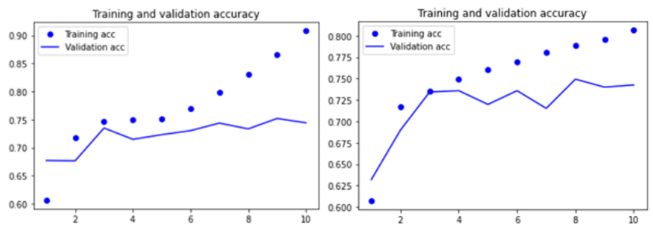


Fig. 5 Accuracy for LSTM & GRU + Word2Vec left – LSTM, and right – GRU

Going further, we tried adding an extra layer in the Convnet and increased the number of training epochs to 20 in an attempt to increase the accuracy of the model:

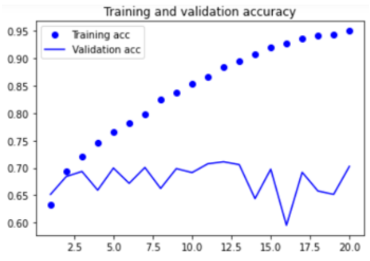


Fig. 6 Accuracy for convnet1D + Word2Vec

The validation accuracy of the CNN model reached 0.70, but this is not much of an improvement over the smaller model.

In the case of fastText, the models showed the following accuracy values on the validation set:

TABLE VII. VALIDATION ACCURACY FOR DL MODELS USING FASTTEXT

Network's capacity						
8	***16***	***32***	***64***	***128***	***256***	***512***
LSTM						
0.6956	0.6873	0.7054	0.7075	0.6641	0.6853	0.6052
GRU						
0.7003	0.7059	0.6982	0.7039	0.6946	0.6868	0.6858
Convnet1D						
0.6196	0.6305	0.6558	0.6636	0.6708	0.6072	0.6532

GRU worked better with fastText, while LSTM and the convnet didn't perform as well. RNN's best models' performance can be seen below, in figure 7.

The graphs show that the accuracy on the validation set did not improve compared to the model in the table.

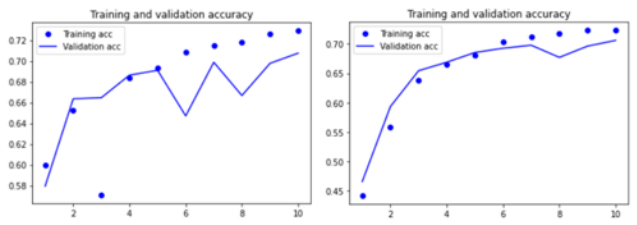


Fig. 7 Accuracy for LSTM & GRU + fastText left – LSTM, and right – GRU

Similar to the previous case, we tried adding an extra layer to the Convnet and trained it on 20 epochs:

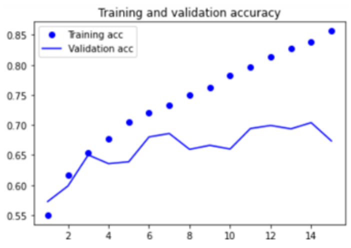


Fig. 8 Accuracy for convnet1D + fastText

Tested on the test set, the best models we have so far show the following results:

TABLE VIII. TEST RESULTS FOR DL MODELS USING WORD EMBEDDINGS

	Word2vec		***fastText***	
	Acc	***F1***	***Acc***	***F1***
GRU	73.12%	74%	70.54%	71%
LSTM	71.26%	71%	68.65%	69%
Convnet1D	70.97%	72%	62.63%	69%

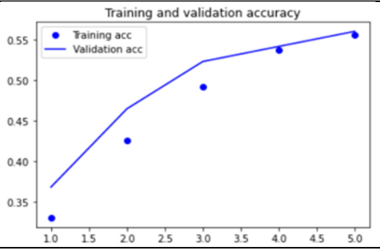
From Table VIII, it can be seen that the models that used fastText performed much worse than Word2Vec (for CNN there was a difference of more than 8%).

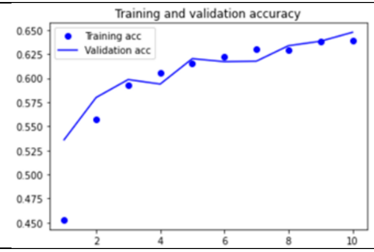
Overall, GRU performed well in both cases, and had the best results. However, what is quite interesting is that the SVM + TF-IDF (both computationally cheaper) had the same accuracy as GRU + Word2Vec, the latter managing to beat SVM by a fairly small difference (of 1%), with an F1 score of 74%.

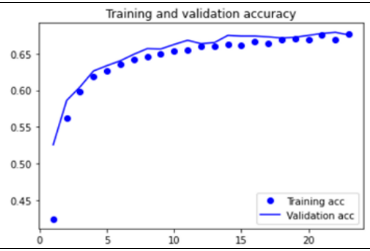
III. DistilBERT

In the case of DistilBERT, we tested the model by adding a dense layer to carry out classification and distinguish between reviews. We tried different combination of learning rates and epochs:

TABLE IX. DISTILBERT + DENSE PERFORMANCE

	Performance
Lr=3e-5 Epochs=5	

Lr=5e-5
Epochs=10
	

Lr=1e-5
Epochs=23
	

From the table above, it can be noted that the model fails to reach overfitting, regardless of the learning rate used and the number of epochs. This might be due to the fact that a larger number of epochs is needed. At the same time, the accuracy remains below 0.70, weaker than most of the methods addressed so far.

Tested, the last model shown in the table has an accuracy of 67.19% on the test set.

At the same time, the classification models created using it failed to reach a point of overfitting, compared to the other DL models that used word embeddings and BOW, which reached overfitting quite quickly (from the first 4 training epochs). This suggests that the model still has some learning to do, or needs to be trained over more epochs and possibly increase its complexity.

B. *Discussions*

All of the results of the best classification models carried out so far can be seen in the table X below.

At first sight, we can see that the best classification method was the GRU model combined with the Word2Vec trained on the original dataset, with an accuracy of 73.12% and a F1 score of 74%. Very close to this was the classical SVM combined with the TF-IDF vectors, which had the same accuracy and only a small difference of 1% in regards to the F1 score.

On the other hand, the worst results were given by the Convnet combined with the fastText vectors with an accuracy of 62.63% and an F1 score of 69%, followed by the DNN model with DistilBERT which had a slightly higher accuracy of 67.19% and the same F1 score.

In terms of classification methods, GRU has managed to do the best with each proposed embedding technique, while LSTM and Convnet did best when using Word2Vec vectors. Also, the DNN worked pretty well with the TF-IDF but not as good with the DistilBERT model.

In terms of feature extraction methods, the pre-trained embedding methods (fastText and DistilBERT) did not help much in the classification process, and even showed some of the worst results.

TABLE X. TESTED MODELS RESULTS

		Accuracy				**F1**			
		TF-IDF	***Word2vec***	***fastText***	***DistilBERT***	***TF-IDF***	***Word2vec***	***fastText***	***DistilBERT***
Classical ML methods	**BernoulliNB**	68.40%	-	-	-	69%	-	-	-
	SVM	73.12%	-	-	-	73%	-	-	-
	Random Forest	71.72%	-	-	-	72%	-	-	-
DL methods	**DNN**	71.81%	-	-	67.19%	72%	-	-	67%
	LSTM	-	71.26%	68.65%	-	-	71%	69%	-
	GRU	-	73.12%	70.54%	-	-	74%	71%	-
	Convnet1D	-	70.97%	62.63%	-	-	72%	69%	-

IX. CONCLUSIONS

In this paper, we have discussed various machine learning methods in order to perform sentiment analysis of online user reviews. We have managed to create a useful dataset containing the reviews and their respective sentiment that can be used in future experiment. We used the data extracted through web scraping in training several Machine Learning and Deep Learning models, and expressed their performance using accuracy and F1 score on the test data. Three feature extraction methods were treated: a classical method (TF-IDF), a more modern method - word embeddings (Word2Vec trained on the original dataset and fastText pre-trained on the Romanian language) and a recent method - contextual word embeddings based on transformers (DistilBERT pre-trained on the Romanian language).

The results indicate that classical ML methods (such as SVM and Random Forest) manage to meet, if not exceed, the performance of more advanced classification methods (such as LSTM and CNN). We can also conclude that pre-trained models (fastText and DistilBERT) did not show very good results, they did not help as expected. Of course, one can still experiment with them.

Based on what has been achieved so far, a number of future developments can be initiated such as:

- training the models on a larger dataset;
- testing combined DL classification methods;
- adding additional layers (LSTM, GRU, CNN…) in the case of classification using the DistilBERT model;
- trying other pre-trained word embeddings (Word2Vec, BERT and ROBERT pre-trained on Romanian or Google's MultiBERT).

REFERENCES

[1] P. Teodorescu, Extragerea unui sentiment uman dintr-un text folosind o rețea neuronală recurentă și biblioteca Keras, *Romanian Journal of Information Technology and Automatic Control*, 30(3), p.119-132, 2020

[2] Maynard, D., & Funk, A., Automatic detection of political opinions in Tweets. *Lecture Notes in Computer Science*, 88–99, 2012, https://doi.org/10.1007/978-3-642-25953-1_8

[3] Y. Hamdaoui, TF(Term Frequency)-IDF(Inverse Document Frequency) from scratch in python, Medium, 2021 https://towardsdatascience.com/tf-term-frequency-idf-inverse-document-frequency-from-scratch-in-python-6c2b61b78558.

[4] A. Pai, An Essential Guide to Pretrained Word Embeddings for NLP Practitioners, Analytics Vidhya, 2020, https://www.analyticsvidhya.com/blog/2020/03/pretrained-word-embeddings-nlp/.

[5] Eh. M. Dharma, F.L. Gaol , H. L. H. Spits Warnars, S. Benfano, The accuracy comparison among word2vec, glove, and fasttext towards convolution neural network (cnn) text classification, *Journal of Theoretical and Applied Information Technology*, 100, p. 349-357, 2022

[6] F. Chollet, Deep learning with python, Manning, Shelter Island, 2018

[7] T. H. Mir, A quick overview of the difference between word2vec and FastText. Medium, 2022, https://medium.com/swlh/a-quick-overview-of-the-main-difference-between-word2vec-and-fasttext-b9d3f6e274e9

[8] S. F. Zaidi, F. M. Awan, M. Lee, H. Woo, C.-G. Lee, Applying convolutional neural networks with different word representation techniques to recommend bug fixers, IEEE Access, 8, 2020

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, arXiv.org, 2017.

[10] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional Transformers for language understanding. arXiv.org, 2019

[11] A.-M. Avram, D. Catrina, D.-C. Cercel, M. Dascălu, T. Rebedea, V. Păiş, D. Tufiş, Distilling the knowledge of Romanian berts using multiple teachers, arXiv.org, 2022

[12] S. Ray, SVM: Support Vector Machine Algorithm in machine learning. Analytics Vidhya, 2021, https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[13] T. Yiu, Understanding random forest, Medium2021, https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[14] S. Raschka, Naive Bayes and text classification I - introduction and theory. arXiv.org, 2017

[15] G. Singhal, Introduction to LSTM Units in RN, 2020: https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn

[16] Phi Michael , Illustrated Guide to LSTM's and GRU's: A step by step explanation, 2018, https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[17] H. Bonthu, Rule-based sentiment analysis in Python for Data Scientists, Analytics Vidhya, 2021, https://www.analyticsvidhya.com/blog/2021/06/rule-based-sentiment-analysis-in-python/

[18] D. Kaemingk, Online reviews statistics to know in 2022, Qualtrics XM, 2021, https://www.qualtrics.com/blog/online-review-stats/

[19] B. Liu, Sentiment analysis mining opinions, sentiments, and emotions, Cambridge University Press, Avenue of the Americas, 2015

[20] C. Bădică, M. Colhon, A. Șendre, Sentiment Analysis on tourist reviews: Data Preparation and Preliminary Results, Proceedings Of The 10 Th International Conference "Linguistic Resources And Tools For Processing The Romanian Language", p. 135-142, 2014, http://consilr.info.uaic.ro/2014/Consilr_2014.pdf

[21] M. Buzea, Ş. Trăuşan-Matu and T. Rebedea, "A Three Word-Level Approach Used in Machine Learning for Romanian Sentiment Analysis," 2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet), 2019, pp. 1-6, doi: 10.1109/ROEDUNET.2019.8909458

[22] E.M. Dharma, F.L. Gaol, H.L.H. Spits Warnars, S. Benfano, The accuracy comparison among word2vec, glove, and fasttext towards convolution neural network (cnn) text classification, Journal of Theoretical and Applied Information Technology, 100, p. 349-357, 2022, http://www.jatit.org/volumes/Vol100No2/5Vol100No2.pdf

[23] Mitchell, Ryan, Web Scraping with Python, O'Reilly Media, 2015