# Contextual Emotion Detection in Twitter Messages

Diana Lucaci
*University of Ottawa*
Ottawa, Canada
diana.lucaci22@gmail.com

Mozhgan Nasr Azadani
*University of Ottawa*
Ottawa, Canada
mozhgan.nasr91@gmail.com

***Abstract*—With the advent of the Internet and the growth of mobile devices such as cell phones and tablets, more and more people are sharing their feelings, posting their points of views and opinions about certain places they have been to or special services they have received in social media. As a result, the amount of information available on social media is increasing explosively. With the ever-growing amount of information on social media, analyzing such data as an attempt to find some useful information has become a challenging task. Among the existing different types of analysis tasks, we have selected to define a project on emotion detecting. We believe this task has various kinds of applications such as measuring the satisfaction of customers of certain products for companies or determining the actual feeling of people in society at a certain time. In this project, we conduct a number of experiments on the task of emotion detection on Twitter data, aiming to improve the existing results that have been published so far for the chosen dataset. The focus of the project is conducting research as well as implementing different types of feature extraction and selection techniques for text data, using word embeddings for social data and implementing various machine learning and deep learning methods for classifying the conversations containing emotions such as happiness, sadness, anger or other. We investigate the performance of different algorithms including Support Vector Machine (SVM), Random Tree, i.e. an ensemble decision trees, as well as Naive Bayes for this task. The results are evaluated using the micro average F1-measure over the classes happy, sad and angry.**

***Index Terms*—Sentiment analysis, emotion detection, twitter, social media**

## I. Introduction and motivation

Social media has expanded rapidly in recent times, becoming one of the richest sources of information that can be explored and used to improve peoples wellbeing in many different areas such as industry, healthcare, business, politics, and security. Since people increasingly communicate using text messages, a number of research challenges arise in the field of artificial intelligence and more specifically in the world of natural language processing.

With the high number of Tweets being sent every day, as an example, the number of Twitter messages being sent every day is almost over 500 million [17], tasks such as detecting mental health problems, changes in behaviors, rumors, fake news, trends, and predicting users suicide attempts, behavioral changes, or depression, understanding customers opinion mining become rather impractical for humans to process and analyze on a large scale. As a result, although social media has made the task of gathering information from people easier (i.e. there is no need to ask people to do extensive surveys or making telephone contact for each person), on the other hand, it has led to the new challenges of processing this large amount of data. Thus, instead of humans, we need to rely on different automated techniques to help us with respect to this problem. Therefore, artificial intelligence faces new challenges in processing text and turning data into actionable insights. In recent times, there has been great progress in this field:

- **Machine learning** techniques that manage to achieve high accuracies for tasks such as machine translations, question answering systems, language modeling, and numerous others;
- Dense **word vector representations** capable of encapsulating semantics;
- **Statistical analysis** of large corpora to determine patterns or anomalies in large data used for cluster analysis, multidimensional scaling ([3]) or classifiers.

Yet, there are numerous differences in the language style, sentence structure, and content between the literary texts and the text shared on networking platforms, making the corresponding tasks more difficult for social media data. As an example, the tweet *'That youre in lov with meeeeeeeeeee, tnx'* contains misspelling, abbreviation, informal language, and missing punctuations. As illustrated in the above example, capturing and working with the irregularities of the language used on social media (informal language, abbreviations, misspellings, grammatical errors, improper sentence structure, mixed languages, inconsistent punctuation, lack of context and many others) is one of the difficulties encountered by the researchers that try to automate those processes.

Furthermore, detecting peoples emotions and opinions is not always an easy task even for an actual person and can be time-consuming and exhausting due to ambiguity. Different types of emotions can be expressed in social media using the same sentences and terms and identical emotions can be expressed in various ways [14]. Moreover, expressing the emotions can be different from person to person based on various factors including their personalities, their religious beliefs, the way they live and so on. In addition to what mentioned above, it is not easy to infer someones emotion based on a sentence or even a part of a sentence that is usually small and not well-explained.

The remainder of the report is as follows. The dataset and


Fig. 1. Training data distribution

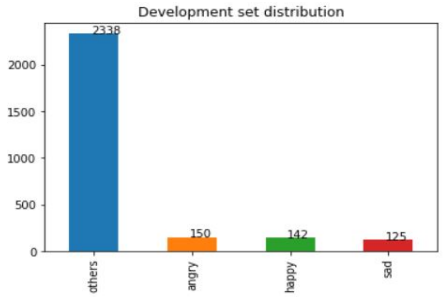


Fig. 2. Development data distribution

the main objectives of the project are discussed in section II and III, respectively. In section IV, we briefly explain the background and previous work. Data analysis and preprocessing techniques, as well as experimental setup, are discussed in sections V and VI, respectively. In section VII, we describe the algorithm. The results of the various types of experiments are reported in section VIII and discussed in section IX. Finally, section X provides the conclusion and future work.

## II. Dataset

To evaluate the implemented ideas and methodology, we will employ the EmoContext dataset. EmoContext is a SemEval task from 2019 [1] that provides a training labeled dataset of approximately 30000 samples of 3-turn Twitter conversations. The label distribution is around 5000 for each of the *angry*, *sad*, and *happy* classes and 15000 for the *others* class, as it can be seen in fig. 1. The development set (2755 samples) and the test set (5509 samples) are also imbalanced (fig. 2, 3), containing 4% of the angry, sad, and happy classes and the rest are labeled as others, thus following a real-life distribution.

## III. Objectives

This project aims to carry out research and implement novel solutions for the task of classifying a sequence of Twitter messages based on the emotion expressed by the authors: sad, happy, angry or others. There are multiple directions that can be exploited when it comes to analyzing social media data and that will be considered during our research for this task:

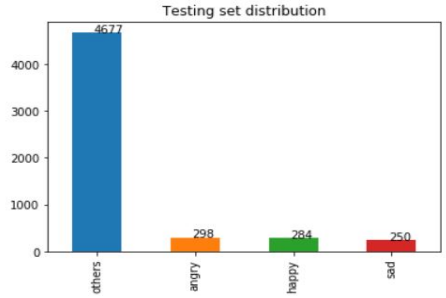


Fig. 3. Testing data distribution

- **Preprocessing and feature extraction and selection** - especially for social media, data graphical representation such as punctuation and emoticons can provide meaningful information about the emotion expressed by the author; such elements are often removed in the preprocessing stages and we think that they can be transformed into useful features; with this in mind, the project report will include multiple experiments that will allow us to draw conclusions based on the results;
- **Dense word embeddings** - pre-trained word embeddings and embeddings learned from specific corpora are two of the most popular representations of words that are considered when using automated methods of processing natural language; the best results for social media data are obtained by adapting pre-trained vectors (such as GloVe - [16]) to the Twitter corpus or by training new embeddings; we plan to research the most successful approaches and to compare the performance of sentiment-specific word embeddings with the baseline of the pre-trained on a general corpus ones;
- **Deep learning** has emerged as a powerful machine learning technique that produces state-of-the-art prediction results ([18]); the project aims to implement such methods and discover the architecture of a model that would lead to high accuracy; we also aim to conduct research whether an ensemble model would improve the baseline accuracy for this task and to provide a report for the results obtained;

## IV. Previous work

In this section, we briefly explain some of the previous works that have been addressed the challenges of the task of emotion detection of text. Emotion recognition from facial expressions and speech are not in the scope of this project and as a result, are not discussed in this report. Different emotion detection approaches exist in previous works such as rule-based methods, neural network-based, reasoning-based, word-level detection, document-level detection, fuzzy logic-based, and so on. However, these approaches can be categorized into four general groups including keyword-based approaches, lexicon-based ones, machine learning ones, and hybrid approaches [14].

Keyword-based approaches such as an early work by Strapparava et al. [15] try to figure out different types of terms and patterns and match them with the emotion keywords. After finding the words expressing someones emotion in the text by POS tagging, these methods match the found list of words with another list of terms providing emotions in accordance with specific emotion models. Those emotions that match with the keywords are considered as the emotions of a particular sentence. There can be situations that a word is matched with more than one type of emotion. In these situations, various approaches can be considered, one of which is to look at the rank of the emotion if there exists a dictionary of keywords containing scores for each emotion. That emotion having the highest score will be chosen as the emotion of the word in this case. Another approach is also to choose the first emotion that matches the word as its corresponding emotion that might not yield the best results. It is worth mentioning that the dictionary to be used varies in different researches depending on the domain, the type of emotions, and the dataset.

In lexicon-based approaches, the task of emotion detection is similar to keyword-based ones, but here an emotion lexicon is considered instead of a list of words [14]. These methods try to classify text employing a knowledge-based repository having labels ready for emotions, that is called a lexicon such as National Research Council of Canada (NRC) or EmoSenticNet (ESN). As an example, Joshi et al. [8] introduced EmoGram in 2016. Their approach is a lexicon-based method which consists of four main parts including a tweet downloader, a tweet emotion scorer, an emotion scorer as well as visualizer. Their method represents the tweets emotions on a timeline in accordance with emotion scores for different labels.

Machine learning approaches have been used in textual emotion detection widely. Most of the work done in this area take advantage of supervised approaches that need a labeled train dataset to be available. Among different types of methods and algorithms, the Support Vector Machine, decision trees, and Naive Bayes are the most commonly employed machine learning classifiers for this task. As an example, Hassan et al. [6], in 2017, proposed an approach to measure emotions of the public and use it to predict significant moments for public events. In their method, they took advantage of Emotex system [5], which is proposed by the same authors earlier in 2014, to do textual emotion detection. This way, their model learns the emotion classification from a large dataset which has been already preprocessed. This study focuses on both negative and positive social events. They analyzed the impact of such events on the public emotion. However, not all the works done using machine learning approaches are supervised. Hajar et al. [4] proposed a new approach using an unsupervised machine learning algorithm to detect the emotions. They computed the probabilities for different words belonging to each emotion class. In this case, the probability of a word is the value of the pointwise mutual information between it and all other words representing the emotion class, and then this probability is averaged for all words in a sentence to get the probability of the complete sentence.

Hybrid approaches, as the name explains, try to combine more than one method aiming at benefiting from the advantages of different methods and achieving higher accuracy. It has been shown that doing so can yield better results than individual methods alone [9].

## V. Data analysis and preprocessing techniques

Twitter data comes along with a number of challenges due to the informal style of the language and the liberty of using slang, words from different languages, misspellings, elongated words, and different symbols (punctuation marks, numbers) that allow people to express feelings, emotions, and sentiments. In order to approach those challenges, in this project we investigated different types of preprocessing techniques as this step is one of the most important parts of a framework. The goal of this step is transforming the data in order to convert the strings to vectors for our different types of classifications. The following preprocessing tasks have been investigated:

- **Stemming.** Stemming is the process of reducing all the words with the same base to a common form. It is commonly used in various tasks of computational linguistics and other NLP tasks.Moreover, different types if word stemmers can be employed such as Lovins Stemmer based on Lovin's algorithm [10].
- **Tokenization.** Different types of tokenizers can be employed at this stage such as a word tokenizer, a Tweet tokenizer or an N-gram tokenizer.
- **Emoji Transformation.** One might think that emojis are not important for classification and as a result they are better to be omitted. However, it is undeniable that they indeed convey people's emotions, feelings, and sentiments. Thus, users use pictorial icons, also known as emoticons (coming from emotional icon), and pictographs of faces, also known as emojis (from Japanese: *e*, "picture", and *moji*, "character") as a way to enhance the expressivity of their messages [2]. Consequently, in the task of sentiment analysis, emotion detection in particular, keeping them is more helpful. However, we cannot keep the emojis as they are and they need to be embedded. To identify these graphical representations, we have used the `emoji` python library and we have transformed them in words as can be seen in fig. 4. For some experiments, we have split the keyword associated to an emoji with its corresponding individual tokens, which lead to a significant improvement, since this method allows us to benefit from the meaning of the corresponding words and not only their co-occurrence and frequency. For example: the emoji id ":thumbs_up:" becomes "thumbs up".
- **StopWords.** They are common words with high frequency such as *'the'* and *'of'* that seem to have no effect on the classification process and are better to be removed.
- **Term frequency.** Evaluating the frequency of the words in the tweets.
- **Capitalization.** Almost every text has different types of capitalization especially for the beginning of the

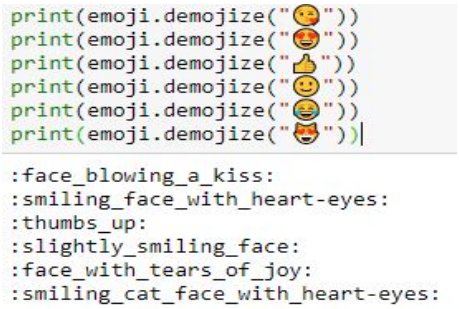


Fig. 4. Emoji translation

sentences and the proper nouns. In our dataset, there are also different types of capital words such as the name of people or some common abbreviations. The most common approach is to make everything lowercase. However, it should be noted that some words may change the meaning.

## VI. Evaluation metrics

In this section, we describe our evaluation metrics. As can be seen in fig. 1, 2, 3, our dataset is highly imbalanced. As a result, the common evaluation metrics such as overall accuracy, precision, and recall over all four class labels, are not appropriate in this case. Therefore, the evaluation is done based on the micro-averaged F1 score for the three emotion minor classes i.e. Happy, Sad and Angry. Doing this, we are able to better evaluate and interpret the results of the classifiers, and also understand how well they are doing. Table I Provides information about different evaluation metrics. As can be seen in Table I, the results are very high for others class as it is the majority class and when we take the weighted average results for different metrics using others as well the results are high. However, this does not show the performance of the classifier regarding the emotions we are interested in i.e. happy, sad, and angry. on the other hand, looking at the micro-averaged F1 scores, it gives us a better understanding of the performance of the classifier in terms of the minority emotions that we are interested in. For instance, the weighted F1-measure is 0.853 for all the four emotions while the micro-average F1 score for the three emotions we are interested in is 0.57, and F1-measure scores are 0.628, 0.554, and 0.457 for emotions angry, happy, and sad, respectively. As a result, we will report the overall F1-measures for different classifiers and deep learning models as well as F1-measure for the three minority classes and the micro-averaged F1 score for the classes happy, sad and angry. The micro-average F1 score is calculated as the harmonic mean of the sum of precision and recall of each class label (happy, sad, and angry).

## VII. Description of the approaches

In this section, we explain various approaches we have employed for the defined task.

### A. Classifiers

We have investigated three different classifiers for this task. These three classifiers are selected from probabilistic models, linear models and ensembles.

- *Naive Bayes Classifier*. The Naive Bayes classifier is a probabilistic model that works based on the Bayes theorem. This classifier has the assumption that the distribution of the training set is the same as the distribution of the test set. In our case, the test set is as imbalanced as the training set. Naive Bayes classifier also works based on the assumption of independence among predictors. It assumes that the presence of a specific attribute in a class is unrelated to the presence of any other attribute. It then calculates the posterior probability using the likelihood and the prior probability. Although it has shown its power for text classification, it does not seem to yield good results in our scenario as can be seen in the results reported in section VIII.
- *Support Vector Machine Classifier*. A support vector machine is a linear classifier that formally tries to find a hyperplane in a k-dimensional space, having k number of attributes, in a way that this hyperplane distinctly classifies the data points. There are a large number of possible hyperplanes that can be selected to separate the classes of the data points. the main objective of the SVM algorithm is to find the best one i.e. the hyperplane that has the maximum margin. It is one of the most well-established learning methods for the classification task. Whenever the input space is not linearly separable, SVM benefits from kernel trick to transform the data into feature space to find a better hyperplane.
- *Random Tree Classifier*. Random tree is an ensemble learning method that takes advantage of using different subsets of the dataset randomly. It is a bagging method which also uses different subsets of the bootstrap samples while training. Since a decision tree may face an overfitting problem over training data, we have selected to compare the results against an ensemble that may lead to better results. However, the result in section VIII will show that even an ensemble is underperforming as a result of overfitting.

All these three methods are tested employing different types of the preprocessing steps mentioned in section V. However, We compared the results against the same type of setting for all methods and report the best ones. For parameter tuning, we have trained the classifiers using 10-fold cross-validation. Moreover, to check the effect of different types of preprocessing steps on the performance of each of the methods, we perform a large number of experiments with different combinations of the steps and report some examples in section VIII. These effects are not always the same for all three classifiers.

### B. Deep learning

**Preprocessing**

TABLE I
A SAMPLE RESULT FOR SVM CLASSIFIER TO SHOW THE DIFFERENCE BETWEEN VARIOUS EVALUATION METRICS FOR OUR IMBALANCED DATASET

| Label | TP rate | FP rate | Precision | Recall | F1-measure | MCC | ROC area | PRC area | Microavg F1(h,a,s) |
|---|---|---|---|---|---|---|---|---|---|
| Others | 0.867 | 0.252 | 0.951 | 0.867 | 0.907 | 0.526 | 0.815 | 0.940 | - |
| Angry | 0.782 | 0.040 | 0.525 | 0.782 | 0.628 | 0.616 | 0.920 | 0.438 | - |
| Sad | 0.688 | 0.038 | 0.464 | 0.688 | 0.554 | 0.540 | 0.905 | 0.356 | - |
| Happy | 0.574 | 0.051 | 0.379 | 0.574 | 0.457 | 0.431 | 0.849 | 0.262 | - |
| Weighted Avg. | 0.839 | 0.221 | 0.876 | 0.839 | 0.853 | 0.527 | 0.827 | 0.852 | 0.57 |

As a preprocessing step for the deep learning algorithms, we have used emoji transformation before embedding the text, as described in section V. Section

**Word Embeddings**

To be able to take advantage of the prediction power of the deep learning models, we need to encode the text with numerical values that can capture the semantics of the words for the context they are used in for each message. To explore the generalization power of different common approaches that proved to lead to high performance for a wide range of NLP tasks, we have performed experiments with different embeddings described in subsection VIII-C.

## VIII. EXPERIMENTAL RESULTS

In this section, we report the results of the experiments conducted employing different classifiers and deep learning methods. The results are more discussed in section IX.

### *A. Different preprocessing techniques results*

Various types of preprocessing steps used to check the performance of the classifiers. Different experiments were conducted employing these different types. However, for the reason of brevity, we selected those using the same setting, but different only in one step to show the effect of each step on the classifiers. Furthermore, We have reported all results on the test dataset.

The effect of the capitalization has been investigated using a setting where there was no term frequency, stemming or stop words removal. Moreover, the word tokenizer has been employed for this experiment. Fig. 5 represents the results for utilizing lowercase in this setting. The results indicates that in our case, making all the words lowercase has slightly improve the performance of all classifiers. As an example for Naive Bayes, micro-average F1 score without lowercase is almost 0.24 while this measure with lowercase is 0.28.

The influence of stemming preprocessing step on the performance of the three classifiers is investigated employing a setting where we have used tweet tokenizer, term frequency and lowercase. the results have been shown in fig. 6. As can be seen in fig. 6, not using stemming have yield better results for Navie Bayes and SVM in our case. For SVM, the reported F1-measure is 0.56 when stemming is not employed while this number is decreased to 0.52 after employing the stemming step. However, although the effect of this step on Random Tree is insignificant, it increased the F1-measure from 0.25 for not stemming to 0.26 with stemming.

We have also investigated the impact of three different kind of tokenizers including a word tokenizer, a tweet tokenizer,

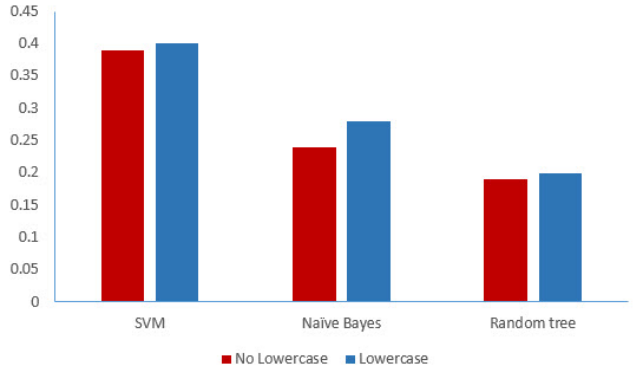


Fig. 5. The effect of Lowercase preprocessing on three classifiers using F1-measure

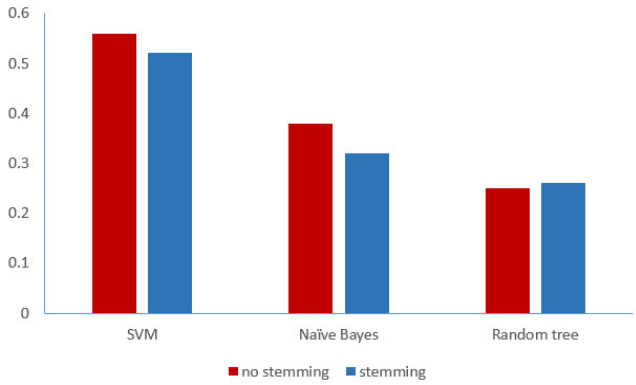


Fig. 6. The effect of stemming preprocessing on three classifiers using F1-measure

and a N-gram tokenizer on the performance of the classifiers. For this experiment, we employed a setting where we have lowercase and term frequency preprocessing, but no stemming. Moreover, each time we employed one of the tokenizer mentioned above. For N-gram tokenizers, we employed a N-gram (1-3) tokenizer. As can be seen in fig. 7, for SVM and Random Tree better results are obtained when we use N-gram tokenizer. As an example, for SVM, the results are 0.51, 0.57, and 0.56 using word tokenizer, N-gram tokenizer, and tweet tokenzier respectively. Not making the results better using a tweet tokenizer may show that particular characteristic of tweet messages such as hashtags and users does not have a significant effect on the task of emotion detection. However, the results indicates that for Naive Bayes, the tweet tokenizer

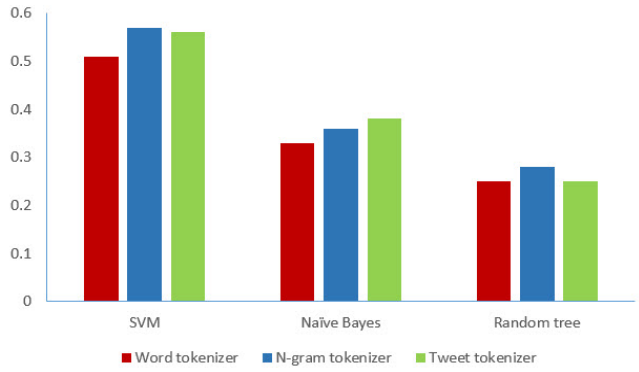


Fig. 7. The effect of three different tokenizers on three classifiers using F1-measure

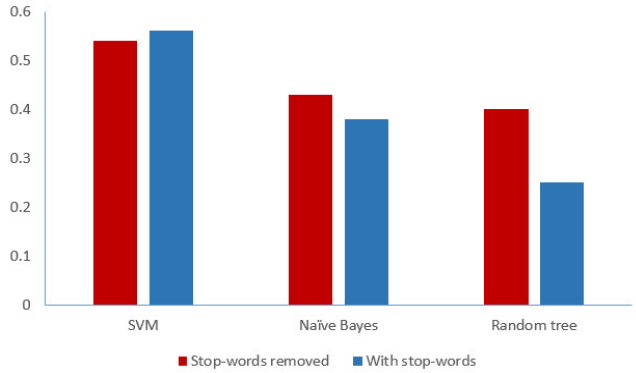


Fig. 9. The effect of stopWords preprocessing on three classifiers using F1-measure

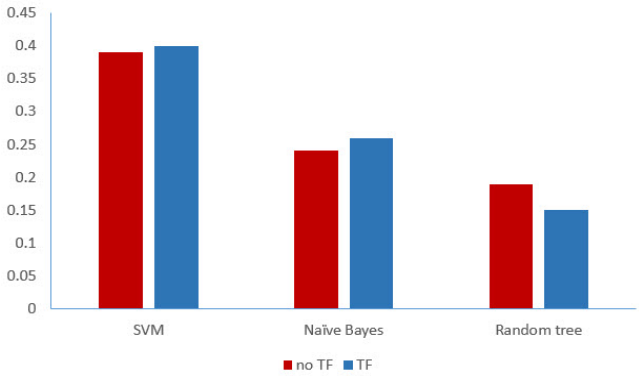


Fig. 8. The effect of term-frequency on three classifiers using F1-measure

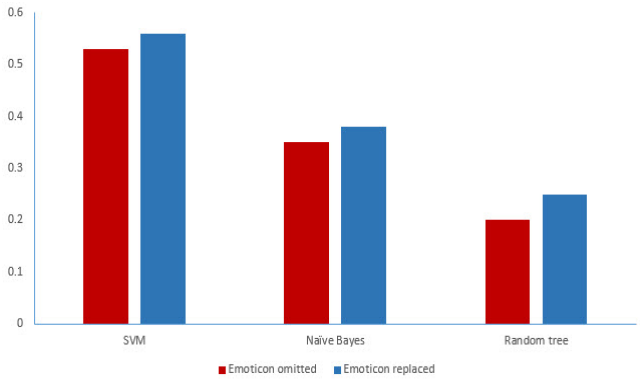


Fig. 10. The effect of emoticon preprocessing on three classifiers using F1-measure

improved the performance. As an instance, micro-average F1 score for this classifier is 0.33 when employing word tokenizer, whereas this measure is 0.38 using tweet tokenizer.

In another experiment, we examine the effect of employing the term frequency on the performance of the classifiers. this experiment was conducted under the setting where there was no lowercase and stemming. Also, we have employed word tokenizer for this experiment. The results shown in fig. 8 indicate that this preprocessing step did not have significant impact on the performance of linear and probabilistic classifiers as the micro-average F1 score for SVM classifer is 0.39 and 0.4 without and with employing term frequency, respectively.

We have also analyzed the impact of removing stopwords in the preprocessing step. The list of stopwords are based on Rainbow [13]. As can be observed in fig. 9, removing stopwords has a significant positive impact on the performance of Random Tree as its micro-average F1 score rose from 0.25 to 0.4. The results indicates that removing stopwords has also a positive impact Naive Bayes classifier. The micro-average F1 score for this classifier increased from 0.38 to 0.43. Focusing on the reported results for SVM, the impact is not significant enough to reach a conclusion.

As discussed in section V, we cannot use the emojis or emoticons in the task of emotion detection as they are and a transformation is needed. In this experiment, we examined the effect of emojis transformation versus ignoring them, removing them from text to be exact. For this experiment, we have employed word tokenizer, lowercase as well as term frequency preprocessing step. As can be seen in fig. 10, transforming the emoticons and emojis has slightly improved the results of the classifiers as they somehow convey the emotions and feelings of people. However, the improvement is not that much significant for SVM and Naive Bayes since the emoji ids are transformed to words based the picture they are showing or the action that is shown in the icons not the real feeling they are meant to convey. The result shown in fig. 10 indicates a rise from 0.54 to 0.56 for SVM classifier, whereas the micro-average F1 score for Random Tree increased from 0.2 to 0.25.

### *B. Classifiers results*

After conducting various types of experiments, we have reported the best results for the three classifers inclusing SVM, Naive Bayes and Random tree over a same setting in table II. The result shown are reported while employing lowercase, term frequency, N-gram tokenizer, and not having stemming or stopwords removal. Moreover, the emojis and emoticons

TABLE II
CLASSIFIERS - RESULTS ON TESTING SET

| Model | F1-all | F1-happy | F1-sad | F1-angry | F1-others | Microavg F1(h,a,s) |
|---|---|---|---|---|---|---|
| SVM | **0.86** | **0.49** | **0.60** | **0.61** | **0.91** | **0.57** |
| Naive Bayes | 0.72 | 0.373 | 0.409 | 0.323 | 0.787 | 0.36 |
| Random Tree | 0.711 | 0.267 | 0.270 | 0.284 | 0.789 | 0.28 |

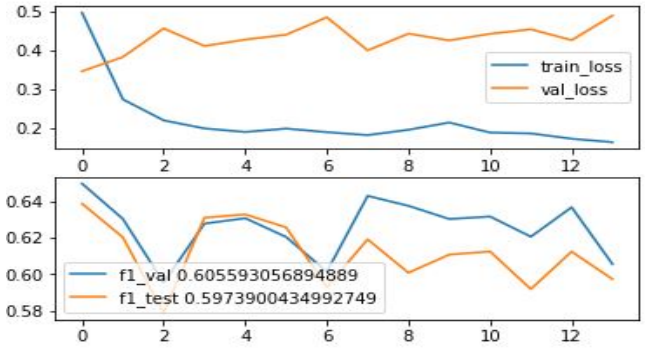


Fig. 11. Semantic enriched model using learned embeddings - loss and F1 score during training

are transformed as explained in section V. As can be seen, SVM outperforms two other classifiers with the micro-average F1 score of 0.57, following by Naive Bayes with the F1 score of 0.36 and Random Tree with the F1 score of 0.28. Moreover, the results of the SVM classifier is significantly higher than those reported for other two classifiers for all measures reported. The results shown in table II are more discussed in Section IX.

### C. Deep Learning

*1) Learned word Embeddings:* Word embeddings learned from text data bring models a performance improvement over sparse representations used in the simpler bag of word model representations. For this reason, we started with a model that learns the embeddings of the words in our corpus from the ids corresponding to them and which are assigned to the words at the preprocessing time. This method aims to capture the meaning of the messages' context specifically from our dataset. This brings the advantage that each token of the training set will have its representation that encapsulates semantics, compared to a pretrained vector representation where all the unknown tokens would be encoded as null vectors or random vectors, which would introduce noise in the model, thus reducing the performance of the system.

The plots of the F1 scores and of the loss value during the training phase (fig. 11) confirms the assumption that the number of instances from the training set is rather small for the size of the deep neural networks that we are using. This is also sustained by Bartlett and Maass' early work on the dimension of neural nets, which states that for a neural network with $P$ parameters, there should be at least $P^2$ training instances. For this reason, we continue the training until the F1 measure starts to vary within a range that does not outperform the

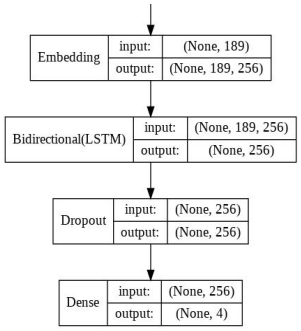


Fig. 12. Architecture of the model using learned embeddings

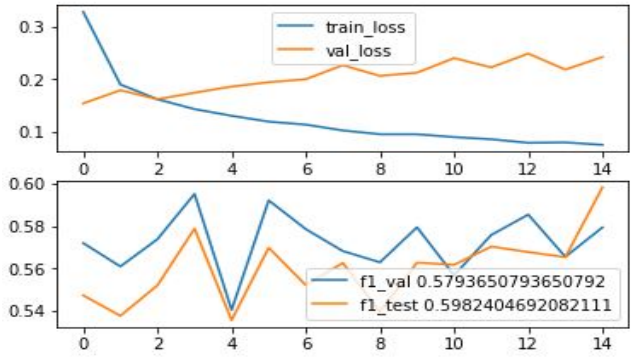


Fig. 13. Model using GloVe - loss and F1 score during training

maximum obtained so far and we report the best ones. The results obtained with a simple architecture (figure 12) can be reviewed in table III.

*2) GloVe Embeddings:* Using pre-trained global vectors for word representation [11] brings along a number of advantages. Firstly, the word vector spaces have linear substructures that capture the meaning of the words that can be easily used for a wide range of natural language processing tasks. Secondly, they are trained on a large corpus, which allows them to take advantage of global count statistics that would not be as representative and able to generalize if they had been trained on a small dataset as the one in discussions. The vectors used in this report are trained on Twitter data, which helps our model achieve better results.

A significant improvement on the performance has the preprocessing technique of tokenizing the emoji ids as explained in section V. This technique allows one to benefit from the GloVe pretrained embeddings for the keywords describing the emojis, which enriches the representation with semantics. The preprocessing technique allows the model to reach its highest performance only after 5 iterations (figure 14). The architecture of the model used with this embedding is shown in figure 15.

*3) ELMO Embeddings:* We are reporting the results of the experiments of a few deep learning architectures using contextualized word representations and bidirectional LSTMs

TABLE III
DEEP LEARNING MODELS - RESULTS ON TESTING SET

| Model | F1-all | F1-happy | F1-sad | F1-angry | F1-others | Microavg F1(h,a,s) |
|---|---|---|---|---|---|---|
| Learned_dense_vectors + BiLSTM+Adam | 0.86 | 0.62 | 0.64 | 0.60 | 0.92 | 0.62 |
| +Rmsprop | 0.80 | **0.67** | 0.37 | 0.58 | 0.88 | 0.50 |
| Learned_dense_vectors + BiLSTM+Adam + emoji_meaning | 0.8742 | 0.6304 | 0.6443 | 0.6263 | 0.9278 | **0.6327** |
| Glove twitter - 100 + BiLSTM(64) | 0.82 | 0.54 | **0.62** | 0.49 | 0.89 | 0.53 |
| Glove twitter - 100 + BiLSTM(128) | 0.85 | 0.55 | 0.60 | 0.54 | 0.92 | 0.56 |
| Glove twitter - 200 + BiLSTM(128) | 0.8600 | 0.5295 | 0.6310 | 0.6129 | 0.9223 | 0.5906 |
| + Dense(100) | 0.82 | 0.54 | **0.62** | 0.49 | 0.89 | 0.53 |
| Glove twitter - 200 + BiLSTM(256) + Dense(256) | 0.8633 | 0.5361 | 0.6639 | 0.6153 | 0.9237 | 0.5982 |
| + **emoji_meaning** | 0.9030 | 0.6686 | 0.7626 | 0.6911 | 0.9446 | **0.7025** |
| **ELMO embedding for each turn** | | | | | | |
| ELMO_AVG_64 | 0.8693 | 0.3864 | 0.6236 | 0.4503 | 0.9308 | 0.4858 |
| ELMO_AVG_128 | 0.8348 | 0.4736 | 0.6298 | 0.4060 | 0.9147 | 0.4799 |
| ELMO_AVG_128 + **emoji_meaning** | 0.8547 | 0.6203 | 0.6224 | 0.4742 | 0.9223 | **0.5480** |
| ELMO_AVG_256 | 0.8647 | 0.5458 | 0.6396 | 0.4716 | 0.9312 | 0.5301 |
| ELMO_AVG_256 + **emoji_meaning** | 0.8825 | 0.6054 | 0.6388 | 0.4916 | 0.9391 | **0.5705** |
| ELMO_CONCAT_rmsprop | 0.8593 | 0.6133 | 0.6776 | 0.4983 | 0.9224 | **0.5765** |
| ELMO_CONCAT_adam | 0.8627 | 0.6167 | 0.6740 | 0.4850 | 0.9276 | 0.5651 |
| ELMO individual processing | 0.8794 | 0.6137 | 0.7050 | 0.6028 | 0.9322 | 0.6346 |
| ELMO individual processing + dropout | 0.8823 | 0.6482 | 0.7202 | 0.5910 | 0.9339 | **0.6449** |
| **Attention layer and glove embeddings:** | | | | | | |
| **LSTM(512) + att + Dense(128) + Dense(64) + rmsprop_optim** | 0.9097 | 0.6854 | 0.7447 | 0.7034 | 0.9488 | **0.7093** |
| LSTM(256) + att | 0.9032 | 0.6643 | 0.7118 | 0.7105 | 0.9448 | 0.6963 |
| LSTM(1024) | 0.9045 | 0.6775 | 0.7563 | 0.6780 | 0.9452 | 0.6998 |
| LSTM(512) + att + adam_optim | 0.8963 | 0.6926 | 0.7067 | 0.6479 | 0.9419 | 0.6782 |
| LSTM(512)+att+Dense(128)+Dense(64)+Dense(4) | 0.8791 | 0.6349 | 0.6792 | 0.6392 | 0.9304 | 0.6494 |
| Att+BiLSTM(256)+Dense(128)+Dense(64)+Dense(4) | 0.8899 | 0.6495 | 0.7219 | 0.6572 | 0.9377 | 0.6723 |
| Att (one-dim) + BiLSTM(256) + Adam | 0.8887 | 0.6593 | 0.7088 | 0.6616 | 0.9359 | 0.6740 |
| Att + BiLSTM(256) + Rmsprop | 0.8774 | 0.6625 | 0.7125 | 0.6166 | 0.9289 | 0.6570 |
| CNN(1) | 0.8950 | 0.6379 | 0.6956 | 0.6767 | 0.9404 | 0.6685 |
| CNN(2) | 0.9063 | 0.6222 | 0.7312 | 0.7266 | 0.9465 | **0.6899** |

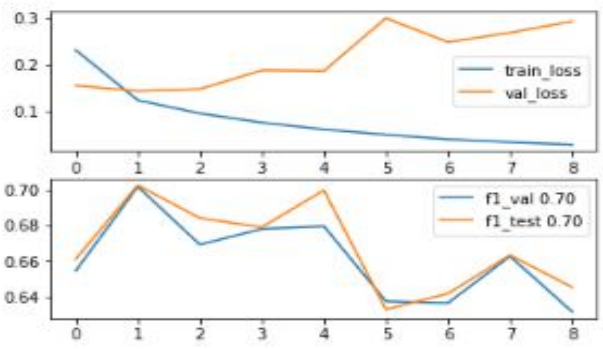


Fig. 14. Semantic enriched model using GloVe - loss and F1 score during training

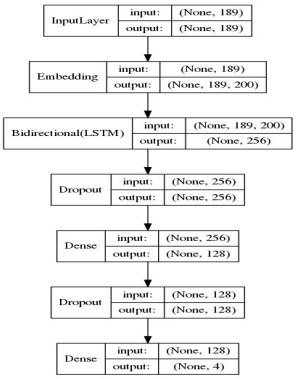


Fig. 15. Model using GloVe embeddings

[12].

**Firstly**, we are using a model that embeds each of the 3 turns and then averages the resulting representation, as it can be observed in figure. 16. For this model, we have reported the results obtained for the batch sizes 64, 128 and 256 in the attempt of detecting the best value of this parameter. The evolution of the micro-average F1 score for the *happy*, *angry*, and *sad* labels for both the validation and testing set can be observed in the right plots of figures 21, 22, and 23. As it can be seen, the models present high variations in terms of F1 score because the precision and recall oscillate from iteration to iteration. The maximum F1 reached is 0.53 on the testing set for the 256 batch size. The least amplitude can be observed for the batch size of 256, which also obtains the best results out of those, as it is also reported in the results table III. The training and validation loss depicted in the left plots of figures 21, 22, and 23 present signs of overfitting since the early epochs of

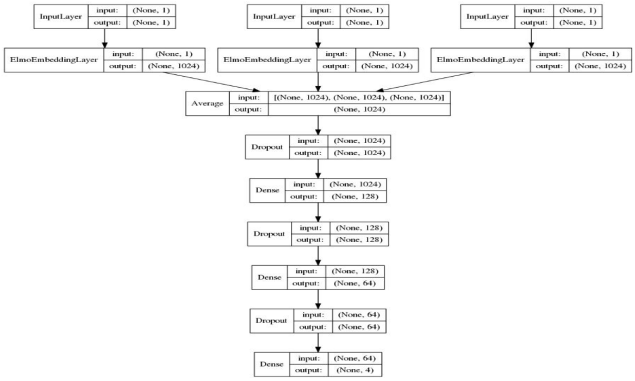


Fig. 16. Model using ELMo layer - averaging the 3 turns

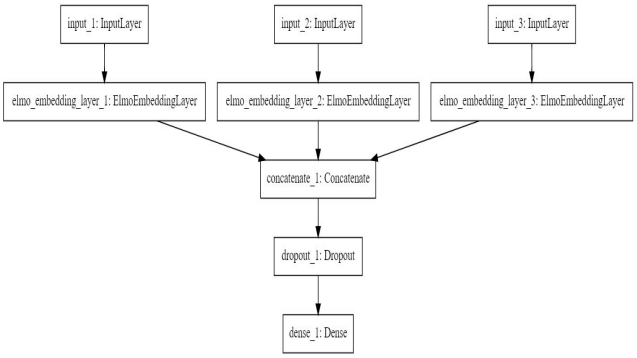


Fig. 17. Model using ELMo layer - concatenating the 3 turns

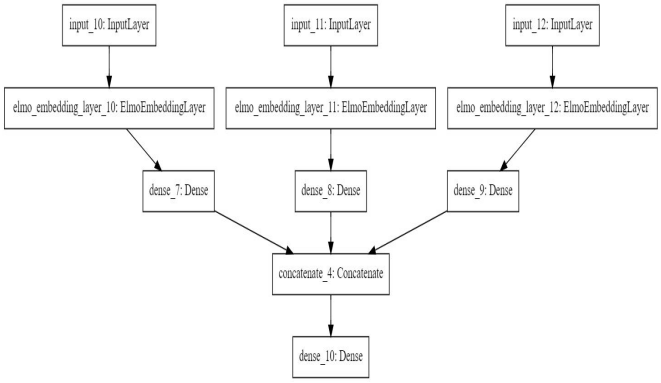


Fig. 18. Model using ELMo layer - individual turns

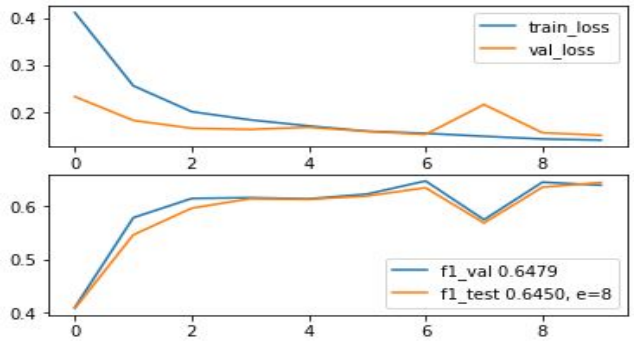


Fig. 19. Training evolution of the model using ELMo layer and dropout-individual turns

the training phase, but the model does not reach its highest potential until later during the training. The model oscillates in the latest epochs between very low values of the F1 score and a maximum of 0.55, without being able to surpass this threshold, which means that on the validation and testing set, the model has some random predictions which means that its generalization power is not very strong.

The reason why F1 scores are lower in comparison with the Glove embeddings might be because of the high dimensionality of the ELMo Embeddings (1024), which is usually suitable for deep networks that are trained on large corpora. In our dataset, we are working with short sentences that are used as inputs for the model. The number of hidden neurons of the Dense layers are 128-64-4 and the dropout rate is 20%. We have used rectified linear unit activation function and softmax for the last layer and RMSprop as optimizer.

The performance is again improved when adding the emoji keywords, as reported in the results table III.

**Secondly**, we have experimented the results of using all 3 embeddings of the turns.

1) **Concatenation.** In an attempt to test ELMo's power of encapsulating the context meaning, we have concatenated the embeddings and added only one output layer using softmax activation function to classify our Twitter messages (fig. 24). These models are comparable better than averaging the turn embeddings since they do not miss any information from each of the turns and preserve the context of them.
2) **Independent turns.** We have discovered that using each turn independently as input to dense layers (fig. 18) leads to more consistent learning, where the F1 score improves with the number of epochs, while the loss function decreases monotonically. As it can be seen in the plot in figure 20, the best F1 score is obtained after the 14th epoch. It can also be observed that overfitting happens

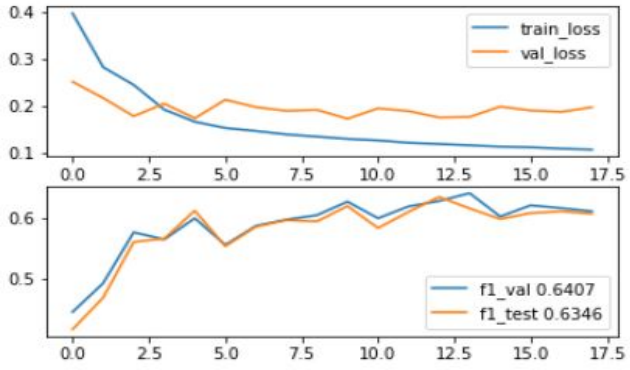


Fig. 20. Training evolution of the model using ELMo layer - individual turns

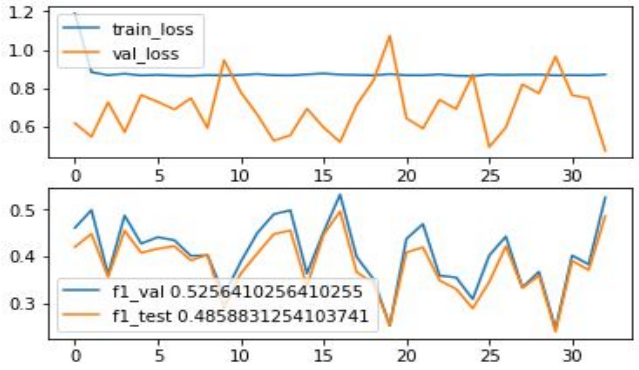


Fig. 21. ELMo Embeddings - averaging the 3 turns - batch size 64

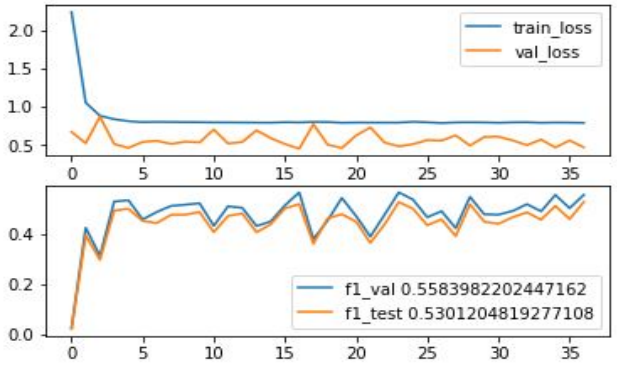


Fig. 23. ELMo Embeddings - averaging the 3 turns - batch size 256

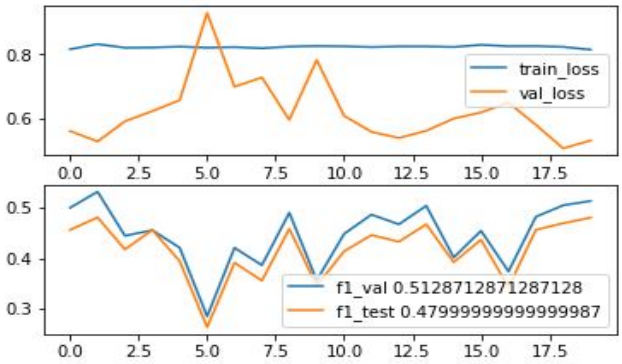


Fig. 22. ELMo Embeddings - averaging the 3 turns - batch size 128

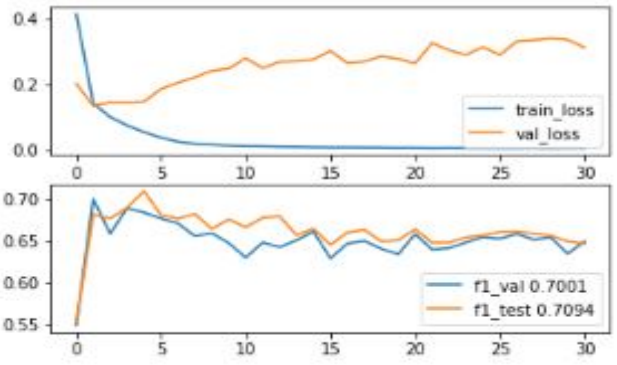


Fig. 24. Learning curve for LSTM + attention layer

in the early stages, even before the best results on the testing set are obtained. To overcome this problem, adding dropout after each dense layer of the turns leads to a more natural learning curve, and a slightly better F1 score, as it can be observed in fig. 19.

3) **Embedding for the concatenated turns.** Since the messages alternate between two authors and the concatenation has a conversation text style rather than a coherent sequence of sentences where ELMo proved to bring the most improvements, this approach obtains an F1-score of 0.4685.

### *D. Attention layers*

Out of all the input consisting of the three turns from Twiiter, some of the words or graphical representations might carry more relevant information than the others. These selected features can help one determine the emotion of the message exchange in a more accurate way. The challenge is to determine what parts of the messages have the most influence in detecting the underlying emotion of the author. To determine this, we use a layer designed to distinguish those features and assign them probabilities by applying the softmax function. There are a number of variations of attention mechanisms used in Natural Language Processing, classified by the number of times the layer uses the same instance to learn the probabilities (single-pass, multi-pass), the number of embedding vectors that encapsulate the information (one-dimension attention - one vector used for the entire input **or** two-dimension attention, where every word of the source has its own embedding vector), and a number of other criteria for classification.

For our project, we have experimented the single-pass attention mechanism, using both one-dimension attention and two-dimension attention mechanism. We have learned that

The attention layer applied after the LSTM layer proved to lead to the best results since it manages to capture the most informative features after the long-term dependencies are captured [7].

### *E. Convolutional Neural Networks*

CNN is a class of deep artificial neural networks that has been initially designed for image processing and that has recently been applied to NLP tasks. In text classification, CNNs proved to bring significant improvements and led to better performance, being capable of capturing the most important features from the word embeddings.

We have experimented 2 architectures depicted in figures 25 and 27. They both reached the highest F1 scores in the early epochs of training. They started to overfit the dataset, as it can be interpreted from the plots in figures 26 and 28, but

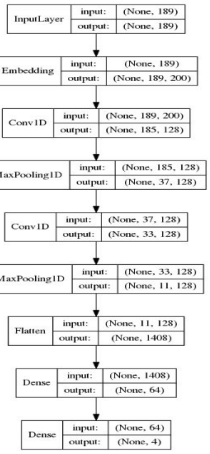


Fig. 25. CNN (1)

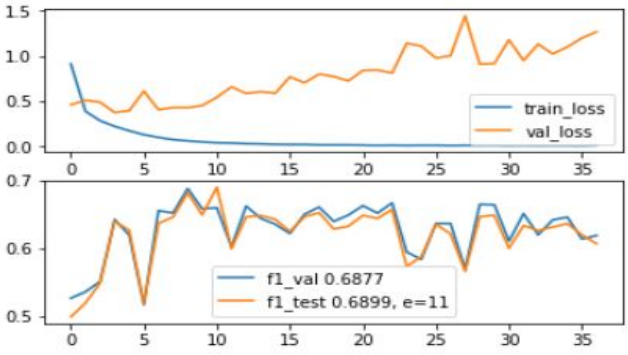


Fig. 26. CNN (1)

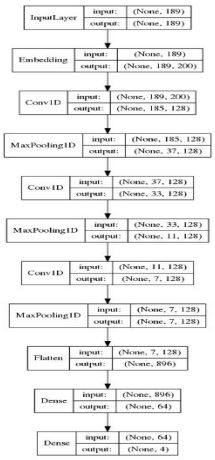


Fig. 27. CNN (2)

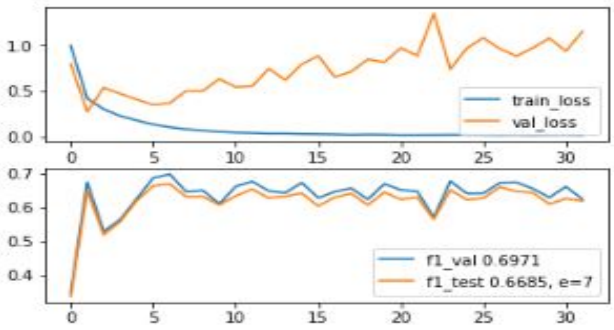


Fig. 28. CNN (2)

the results obtained using these architectures are comparable with the best results obtained using the attention layers.

## IX. Discussion

We have examined different kinds of preprocessing steps, classifiers and deep learning models to perform the task of emotion detection. The results are reported in section VIII and now discussed in this section.

Regarding different types of preprocessing steps, it has been shown that merely taking one step since it is common for other kinds of text classification is not yielding at better performance for the three classifiers. As the results shown in figures 5, 6 suggest employing lowercase and not performing stemming step improves the performance of the classifiers. One reason could be that stemming for sentiment analysis can be somehow problematic since it can destroy some semantic while reducing the words into based forms. Moreover, as represented in fig. 7, Tweet tokenizer does not always improve the performance of the classifiers, whereas the N-gram tokenizer has better results. Using all 1-3 N-grams is giving both more context and larger sample set to build the model on, and can improve the performance of the classifiers. As expected, removing stopwords usually yields better results (as shown in fig. 9, as these high frequency common words do not convey that much feelings and emotions in our senorio. Transforming the emojis and emiticons makes the results better since they contain semantics and are a way to express people's emotions and feeling. However, the improvements are not as high as expected (as shown in fig. 10). One reason could be the way these emojis and emoticon are replaced with words i.e. not every word used to replace an emoji or emoticon is related to the exact feeling and emotion that it conveys. Consequently, although emoji transformation can make the performance better, but the change is not as high as expected.

As can be seen in table II, the Random Tree has the worst F1 scores in comparison with two other classifiers, especially to those of SVM. We explored the results of these algorithms for the training dataset as well and we understood that the Random Tree has the best results. The results on training dataset for all three classifiers have been shown in table IV. The experiment setting is the same as the one for test dataset

TABLE IV
CLASSIFIERS - RESULTS ON TRAINING SET

| Model | F1-all | F1-happy | F1-sad | F1-angry | F1-others | Microavg F1(h,a,s) |
|---|---|---|---|---|---|---|
| SVM | 0.901 | 0.853 | 0.912 | 0.898 | 0.911 | 0.89 |
| Naive Bayes | 0.685 | 0.647 | 0.648 | 0.624 | 0.731 | 0.64 |
| Random Tree | **0.997** | **0.996** | **0.996** | **0.997** | **0.997** | **0.99** |

i.e. employing lowercase, term frequency, N-gram tokenizer, not having stemming or stopwords removal, and transforming emojis and emoticons. As can be observed, the Random Tree performs almost perfect on the training dataset, while it has the worst performance facing the unseen dataset, i.e. test dataset. The results indicate the overfitting problem which is very common in decision tree algorithms. Random Tree tries too hard to have a good performance on the training dataset that cannot perform the same for an unseen dataset which is different from the training dataset. Moreover, Focusing on the results obtained by SVM and Naive Bayes on both training and test dataset, it has been shown that SVM has better results. One reason can be the way these two models deal with the attributes. Naive Bayes algorithm treats the attributes as they are independent, whereas SVM tries to look at the interactions between the attributes to a certain degree.

In terms of deep learning approaches, short text raises a number of challenges for these methods since the complex architecture leads to a large number of parameters which require a large amount of data for training. Since the training set is not large enough for deep neural networks, the performance is affected in a negative way. This is also confirmed by the oscillations of the F1 score and the loss values during the training.

The best F1 measure is obtained in the early stages of the training, which means that the model is struggling to learn from the training data. The testing set contains rather new words or variations of words, presenting different patterns than the training data. Thus, the training set is not only non-representative to the testing set, but it also lacks the diversity and number of examples needed to generalize well enough. This makes the task more difficult. Moreover, the accuracy obtained on the training set reaches a rate of even 99.96%, which is a sign of overfitting, which can also be observed in the loss function plots presented in this report. However, the F1 measure does not have a direct correlation with the decrease of the loss function, as it can be observed from figure 21. The best F1 measure can be obtained even after the loss function for training and validation present the overfitting signs (as it can also be observed in the case of CNNs). Thus, the high accuracy on the training set means that the model reaches its highest potential of learning. The same phenomenon applies when the dimension of the network is drastically reduced (word embeddings of 50 elements, LSTM-64, Dense-16).

Deep architectures using LSTM layers obtain one of the highest micro-average F1 score, followed by the enhanced model using attention layers, which manages to put more weight on the important features, thus leading to the best results. Although we have only performed few experiments with CNN architectures, these models quickly managed to reach a good F1 score, which reinforces the fact that they are suitable for NLP tasks.

## X. Conclusions and future work

The project offers a lot of opportunities in terms of both developing technical skills for automated natural language processing and machine learning and in terms of driving compelling insights from social media text data. In terms of improvements that can be done for the approaches described above, we propose the following:

1) **Studying other deep learning architectures.** CNNs, along with RNNs are popular models that proved to have good results in classification tasks for NLP. It is worth exploring more and interpreting the results obtained with the CNN models, which may lead to discovering the architecture that is the most suitable for the task in discussion.
2) **Preprocessing for deep learning.** From our experiments, we have come to the conclusion that even the word embeddings learned on this dataset can encapsulate the meaning of the words. This was proven by the increase of performance when employing the emoji preprocessing. The models considerably improved the performance where the ids of the emojis were transformed to meaningful tokens, which means that only the statistical information provided by this features is not informative enough. For this reason, we think that using a spell checking module and transforming the elongated words to their dictionary form would improve the performance of the system since the words would benefit from the embeddings previously learned from texts with a more rigorous language style.
3) **Evaluation metrics** are important for deciding for which subtask the model does not perform well. We should, thus, consider looking at the true positive rates for each of the classes of interest (happy, angry, sad), in order to interpret and explain which model is suited for which label of our data.
4) **Adressing the class imbalance problem**. One of the aspects that make this task difficult is that the training set is highly imbalanced, containing as majority class the messages labeled as *others*. The evaluation of the model is focused only on the rest of the classes, which are learned by discrimination between classes. Using methods such as undersampling or ensembles trained on subsets of the data might have a positive impact on the performance.

Determining the most promising features for emotion detection and developing a system that can further become a component of a more complex system (such as a topic popularity prediction system for social media or election prediction systems based on people's opinion expressed on online platforms) will also have an important impact on improving people's wellbeing.

## References


[1] Ankush Chatterjee et al. "SemEval-2019 Task 3: EmoContext: Contextual Emotion Detection in Text". In: *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Minneapolis, Minnesota, 2019.

[2] Cydney Grannan. *What's the Difference Between Emoji and Emoticons?* URL: https://www.britannica.com/story/whats-the-difference-between-emoji-and-emoticons.

[3] Stefan Th. Gries. *Useful statistics for corpus linguistics*. 2019.

[4] Mousannif Hajar et al. "Using youtube comments for text-based emotion recognition". In: *Procedia Computer Science* 83 (2016), pp. 292–299.

[5] Maryam Hasan, Elke Rundensteiner, and Emmanuel Agu. "Emotex: Detecting emotions in twitter messages". In: (2014).

[6] Maryam Hasan et al. "Using social sensing to discover trends in public emotion". In: *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*. IEEE. 2017, pp. 172–179.

[7] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[8] Aditya Joshi et al. "Emogram: an open-source time sequence-based emotion tracker and its innovative applications". In: *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[9] Xiangsheng Li et al. "Hybrid neural networks for social emotion detection over short text". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 537–544.

[10] Julie Beth Lovins. "Development of a stemming algorithm". In: *Mech. Translat. & Comp. Linguistics* 11.1-2 (1968), pp. 22–31.

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[12] Matthew E. Peters et al. "Deep contextualized word representations". In: *Proc. of NAACL*. 2018.

[13] *Rainbow.* https://www.cs.cmu.edu/~mccallum/bow/rainbow/. Accessed: 2019-02-25.

[14] Kashfia Sailunaz et al. "Emotion detection from text and speech: a survey". In: *Social Network Analysis and Mining* 8.1 (Apr. 2018), p. 28.

[15] Carlo Strapparava and Rada Mihalcea. "Learning to Identify Emotions in Text". In: *Proceedings of the 2008 ACM Symposium on Applied Computing*. SAC '08. Fortaleza, Ceara, Brazil: ACM, 2008, pp. 1556–1560. ISBN: 978-1-59593-753-7. DOI: 10.1145/1363686.1364052. URL: http://doi.acm.org/10.1145/1363686.1364052.

[16] Duyu Tang et al. "Learning sentiment-specific word embedding for twitter sentiment classification". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2014, pp. 1555–1565.

[17] *Twitter For Business — Twitter Tips, Tools, and Best Practices.* https://business.twitter.com/. Accessed: 2019-02-25.

[18] Lei Zhang, Shuai Wang, and Bing Liu. "Deep learning for sentiment analysis: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1253.