Machine Learning Project - Twitter Sentiment Analysis

Sami Ben Hassen, Firas Kanoun, Ali Fessi
CS-433 Machine Learning, EPFL, Switzerland

***Abstract*—With faster and more powerful processors, Machine learning[1] has become very important and offers useful tools and techniques to deal with a lot of problems in almost every scientific field. Classification[2] is perhaps one of the most useful tools in machine learning in order to resolve a tremendous number of real world applications. From classifying skin cancers to Amazon reviews, computer scientists are aiming to improve the different algorithms put in place in order to have better prediction while reducing computational costs. Our aim in this paper is to present the different paths one can explore in order to predict the sentiment[3] from a corpus of text.**

I. Introduction

Twitter[4] is one of the most popular social medias. It allows people, through a limited number of characters, to express their opinions in several matters. Each interaction is called a tweet and our objective is to analyze these tweets. Because of the limited number of characters, people tend to use expressive words that will allow them to express their opinion concisely. Therefore it's a great data-set to work on in order to perform Natural Language Processing[5] and text classification[2].

In this paper, we will discuss different machine learning models we have implemented in order to predict if a tweet expresses either a positive or a negative sentiment. The data-set we used consisted of 2.5M tweets that were labeled by 1 if it had the emoji[6] ':)' and by 0 for ':('.

In section II, we will explore the data in order to have a good understanding of the data-set we're dealing with. Section III will be dedicated to the presentation of several tricks to clean our data for Machine Learning[1] algorithm implementations. In section IV we will discuss the embedding of our tweets using different techniques such as TF-IDF and Word2vec. Since Machine Learning algorithms don't understand texts, we needed to quantify each tweet to fit our models, trying as best as we can to keep word semantics. Finally we will explore implementations of several models and tune them to achieve the best predictions.

II. Exploratory Data Analysis

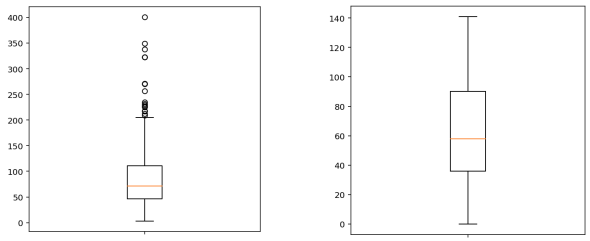
Our data-set contains 2.5M tweets. Often, each tweet will be written in an 'abbreviated English' or 'slang' that may be challenging to understand. It may also contain undesirable characters as the # that is widely used on the platform. Some cleaning is required. But before that, let's have some metrics and insights about the data.

A. Some insights about our data

Looking at Fig.1a we notice that there are a lot of tweets that are out of twitter's character bound. Taking a closer look at the data-set, there were a couple of observations that grabbed our attention :

1) Tagged users are marked as "<user>".
2) There are links in the text showing as "<URL>".
3) A good number of these tags could be chained.

Once we clean everything up we end up with the character distribution shown in the box-plot of fig.1b



(a) Before cleaning (b) After cleaning

Figure 1: Length of the tweets

B. Importance of URL and USER tags

The URL and USER tags are used a lot through the data-set but one can wonder if removing them is actually a good idea. Grouping the tweets that contain these keywords by sentiment and counting the number of occurrences we get the results depicted in the table.I below :

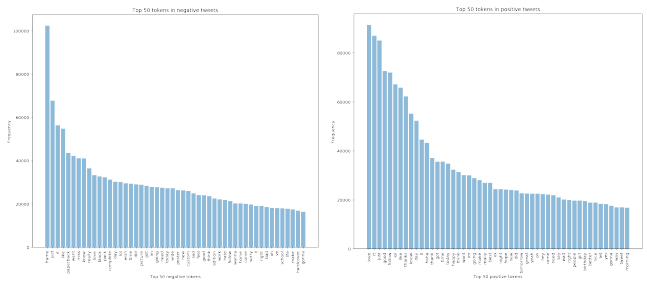
Tag	Sentiment	number of tweets	%
<user>	0	477 397	37
	1	813 397	63
<url>	0	423 924	81.4
	1	97 063	18.6

Table I: User and Url keywords impact on Sentiment

We notice that users tend to tag other ones when they are expressing a positive sentiment while they link to other websites when they are feeling sad.

C. Negative and Positive Words

If we are trying to classify tweets based on words, we have to know which ones are the most used depending on the writer's sentiment. In fig.2 we have counted the top 50 most used words in both of these classes.



(a) Top.50 Negative (b) Top.50 Positive

Figure 2: Top 50 keywords for each class

III. Data Cleaning

As we have mentioned above, our data came with a lot of noise, to clean it up and make it usable we tried different techniques like :

- Truncating words of length 2 or 3.
- Removing digits.
- Replacing punctuation repetition.
- Removing the punctuation entirely.
- Replacing contractions.
- Replacing 'not word' by antonym.
- Replacing elongated words.
- Stemming[7] or Lemmatizing[8].
- Using a library designed for this task: We chose to use **ekphrasis**[9]

Through various trials, we found out that the best results to avoid over fitting were to implement algorithms after doing the following :

1) Replace contractions.
2) Replace punctuation repetition by something like "multistop" or "multiquestion".
3) Remove the remaining punctuation.
4) Use the library **ekphrasis**[9]

IV. Feature engineering

Since we can't run our machine learning algorithm on text, we had to convert all of our data-set rows to a numerical representation. In this section we will first study the importance of stop words[10]. Then we will look into which N-GRAM[11] yields the best results for our task .Finally we will compare the performance of our Machine learning algorithm using different vectorization techniques and see which one performs the best.

A. Stop Words Importance

Stop Words are the words which supposedly do not contain important significance in a sentiment analysis classification task, such as you, a etc... Logically, removing the aforementioned words should increase our model performance. But we wanted to test if this was actually true. Fig.3 shows the results we got for the cases where we:

1) Use stop words
2) Remove all stop words
3) Remove the top 10 most common words in our data

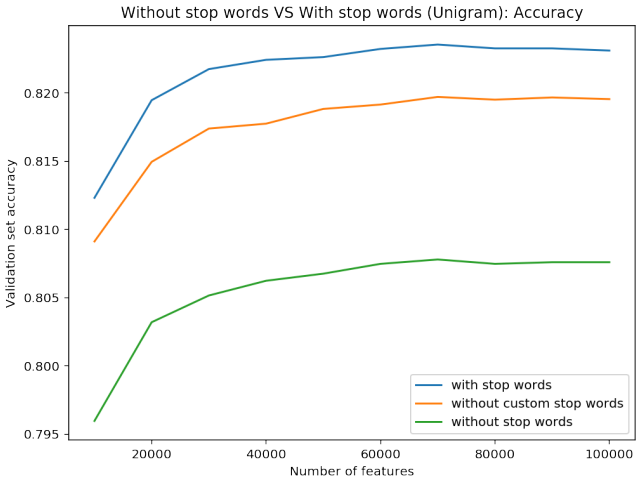


Figure 3: Stop Words Importance

The results we get are surprising to say the least. Dropping stop words actually reduces the performance of our model by a lot. We therefore decided to go against the trend and keep them on our tweets.

B. N-GRAM comparison

An N-GRAM is a contiguous sequence of n items from a given sample of text or speech[11]. This is really important since we want to compare whether taking the word out of its context (Unigram) or taking the words 3 by 3 (Trigram) impacts the performance of a machine learning algorithm model. Fig.4 shows that the results are as expected: Taking the word out of its context is not really a good idea if we want to classify the sentiment of a text writer.

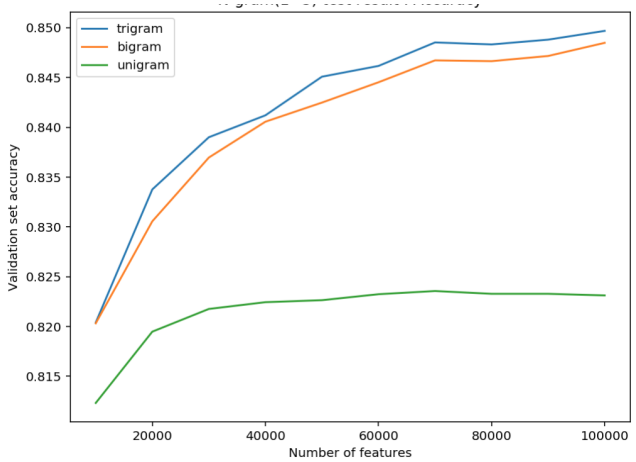


Figure 4: N-GRAM impact of performance

C. Tf-idf Vs. Count Vectorizer

TFIDF[12] or term frequencyinverse document frequency, is a numerical statistic that is intended to reflect how

important a word is to a document in a collection or corpus. The tfidf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. It also gives weight to words that do not appear very often in order not to lose any notion of semantic.

CountVectorizer[13] on the other hand just counts the occurrences of each word in its vocabulary and then computes its frequency of appearance.

Another important feature is the number of words of the vocabulary we choose to use in order to implement the vectorizers we mentioned above.

Fig.5 shows the results of our testing for both of these methods against the number of words used in the dictionary and we see that in all the cases it is Tf-idf that yields the best accuracy. So we will use it for our future comparisons between the different machine learning algorithms.

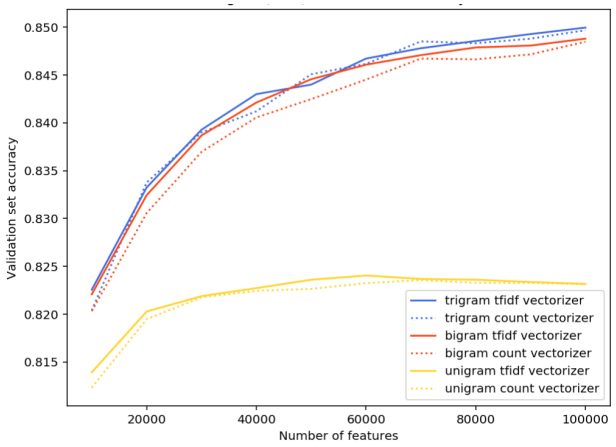


Figure 5: Tf-idf Vs. CountVectorizer

D. Word Embedding

While the two previous techniques are based on one-hot encoding with each tweet represented as a vector of 0 and 1's, word embedding uses a different approach. Each word is positioned into a multi-dimensional space called embedding space and each vector value is represented by its position in this space. Synonyms are found close to each other while words with opposite meanings have a larger distance between them as shown in Figure 6.

1) Word2Vec: To put this into practice, we first separately learned each word embedding by training a Word2vec model on our corpus of tweets and then passed the resulting embedding matrix of words as an input to the first hidden layer of a tuned-parameter neural network model. The challenge when learning the vector values of words was to find the best vector dimension, representing the number of dimension of each word vector and the bets window slide, representing the neighborhood of semantic of the word vector we want to learn, that best fit our model with a wise trade-off between our daily bread, efficiency and time consumption. Since each model took us 3 to 4 hours to train for a vector size of 50 and despite discarding all the words that did not appear very often to speed up the learning, we could not train many models for the latency of the process. The best one giving the best results achieved an accuracy of 83.99% with a convolution neural network of 3 hidden layers and a sigmoid activation with drop out of 20% between layers. This will be discussed more in details in part.VI

2) Glove: Because the training data is never enough to cover all what it said on twitter, we thought that our trained model might not be able to learn the best embeddings for sentiment analysis and therefore decided to load a pre-trained word embeddings trained on a much larger data. For this sake, the GloVe database contains multiple pre-trained word embeddings, and more specific embeddings trained on tweets. So we thought that this might be useful for the task at hand. Using the same training and settings as the model above, the accuracy achieved was of 78.62%. This drop in accuracy can be explained by the fact the trained model is the perfect fit for our data set while the pre trained contains semantics that do not figure in our tweets.

Figure 6: Two dimentional scatter of some word clusters after training a Word2Vec model

V. Machine Learning Algorithms

As seen above, the best results we got were by using TF-IDF with 100,000 words in its dictionary with TriGRAM. In this section we discuss several algorithms we learned during this semester and compare their performance according to their accuracy and the time it takes for them to converge/finish computations.

From the results in tab.II we see that Linear SVM[14] with l1 penalty outperforms all the other ones. Next we wanted to see if creating a Voting Classifier using the top 3 performing algorithms would outperform them. It turns out that it performs as good as the Linear SVM.

We did not experiment with the more computationally expensive models like KNN[15] and random forest[16],

since the data is huge and the operations of the aforementioned algorithms are computationally expensive. Tab.II below shows the results:

Algorithms	Penalty	Accuracy(%)	Time(s)
Linear SVM	**l1**	**85.1**	**980**
Linear SVM	l2	85	451
Logistic Regression	l2	85	480
Ridge Classifier	-	84.9	520
Multinomial NB	-	81.2	415
Bernoulli NB	-	80.1	369
Voting Classifier	-	85.1	-

Table II: Results of Multiple ML Algorithms.

VI. Neural Networks

Neural Networks[17] are a specific set of algorithms that have revolutionized machine learning. They are inspired by biological neural networks and the current so-called deep neural networks have proven to work quite well. Neural Networks are themselves general function approximations, which is why they can be applied to almost any machine learning problem about learning a complex mapping from the input to the output space. In this section we will explore different types of Neural Networks and test their ability to predict twitter sentiments correctly.

A. Convolutional Neural Networks : CNN

A convolutional neural network[18] is a class of deep neural networks, most commonly applied to analyzing visual imagery.

B. Long short-term memory

Long short-term memory (LSTM)[19] units are units of a recurrent neural network (RNN)[20]. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

C. Gated Recurrent Unit

Gated Recurrent Units (GRUs)[21] are improved version of standard recurrent neural network that are made to solve the vanishing gradient problem of a standard RNN. GRU uses an update gate and reset gate that can be trained to keep information from very long ago without vanishing it through time.

D. Results of different combinations of NNs

After running different combinations of NN, we saw results that are clearly better than the usual machine learning algorithms. The results are displayed in tab. III

Fig.7 below shows the evolution of validation accuracy through the epochs for each kind of neural network we have tested.

We can see that the convolutional neural networks combined with Gated recurrent unit yield the best validation accuracy for the problem we are dealing with.

Type	Accuracy(%)
NN + LSTM	85.3
CNN + LSTM	85.6
CNN + GRU	**85.7**
Bidirectional GRU	85.2

Table III: Results of Multiple NN implementations.

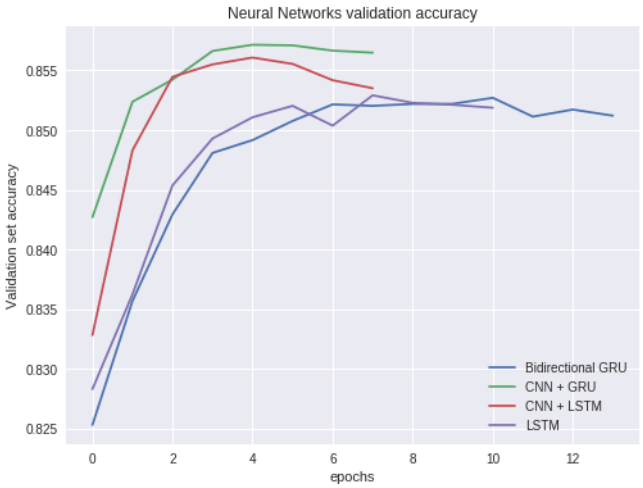


Figure 7: NN performance through the epochs

VII. Conclusion

Through the course of this project, we explored how people write their tweets and discussed different techniques to clean this kind of textual data that manage to yield the best classification results. Next we compared different types of feature engineering. We tested 5 methods of machine learning and even combined the top performers in a voting algorithm and saw that it gives results that are similar to the best algorithm when dealing with textual data in our case. And finally we tested multiple combinations of different types of neural networks. We experienced hands-on the power and the limits of each one of them.

References

[1] "Machine learning," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning

[2] "Machine learning," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning

[3] "Sentiment analysis," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Sentiment_analysis

[4] "Twitter. it's what's happening." [Online]. Available: https://twitter.com/

[5] "What is natural language processing and generation (nlp/nlg)?" Feb 2018. [Online]. Available: https://bdtechtalks.com/2018/02/20/ai-machine-learning-nlg-nlp/

[6] "emoji." [Online]. Available: https://www.yourdictionary.com/emoji

[7] "What is stemming? - definition from whatis.com." [Online]. Available: https://searchenterpriseai.techtarget.com/definition/stemming

[8] "Stemming and lemmatization in python." [Online]. Available: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python

[9] C. Baziotis, N. Pelekis, and C. Doulkeridis, "Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, August 2017, pp. 747–754.

[10] "Stop words," Oct 2018. [Online]. Available: https://en.wikipedia.org/wiki/Stop_words

[11] "N-gram," Nov 2018. [Online]. Available: https://en.wikipedia.org/wiki/N-gram

[12] "Tfidf," Oct 2018. [Online]. Available: https://en.wikipedia.org/wiki/Tfidf

[13] Russell, "More nlp with sklearn's countvectorizer russell medium," Aug 2017. [Online]. Available: https://medium.com/@rnbrown/more-nlp-with-sklearns-countvectorizer-add577a0b8c8

[14] "Support vector machine," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine

[15] "K-nearest neighbors algorithm," Nov 2018. [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[16] "Random forest," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Random_forest

[17] J. Le, "A gentle introduction to neural networks for machine learning." [Online]. Available: https://www.codementor.io/james_aka_yale/a-gentle-introduction-to-neural-networks-for-machine-learning-hkijvz7lp

[18] "Convolutional neural network," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network

[19] "Long short-term memory," Nov 2018. [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory

[20] "Recurrent neural network," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network

[21] "Understanding gru networks," Dec 2018. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be