Sentiment analysis in tweets

1st David Silva
MIEIC - FEUP
Porto, Portugal
up201705373@fe.up.pt

2nd Luís Cunha
MIEIC - FEUP
Porto, Portugal
up201706736@fe.up.pt

3rd Manuel Coutinho
MIEIC - FEUP
Porto, Portugal
up201704211@fe.up.pt

***Abstract*—In this paper we present two different approaches for solving the subtask 5 of the SemEval-2018 Task 1 (English version). The basic model is a pipeline with a preprocessor and a classifier. We used Naïve-Bayes as our baseline and Logistic Regression as our main classifier. The other model made use of more advanced NLP techniques, such as word embeddings and long short-term memory (LSTM) neural networks. Despite the unbalanced and reduced dataset, we achieved satisfactory results breaking the 50% multilabel accuracy mark with our first model and 55% with the more complex one.**

***Index Terms*—Natural Language Processing, Sentiment analysis, Twitter, Tweet, Tweet preprocessing, Classification, Multilabel, SemEval-2018**

I. Introduction

Natural Language Processing is a subfield of many disciplines including **artificial intelligence**, **computer science** and **linguistics**. The goal of NLP is to program a computer to extract and process information from large amounts of **natural language data** [1].

A common classification task of NLP is **sentiment analysis** (also called opinion mining), which strives for the extraction of **sentiment**, i.e. the affective and emotional state of the speaker, from a textual source [2].

In the present project, done for the course of "Artificial Intelligence" we will dive into the topics of NLP and sentiment analysis by applying preprocessing and classification techniques on task E-c of *SemEval-2018* which regards sentiment analysis in **Tweets** [3].

II. Problem description

The goal of *SemEval-2018*'s **E-c task** is to classify a Tweet as '**neutral** or no emotion' or as **one, or more**, of **eleven** given emotions (anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise and trust) that best represent the mental state of the tweeter.

"I have the best girlfriend in the world #blessed" (1)

For example, the tweet 1 is classified as having present the sentiments of **love**, **joy** and **optimistm**.

Using Tweets as a textual source for Natural Language Processing is challenging due to their reduced size and noisy data. Also, Tweets, as is general in text extracted from social media, contain certain language constructs that are not present in common textual sources, such has **emoji**, **hashtags**, **slang** and **abbreviations**.

" Only I could be talking to a catfish on tinder ☺
glad I don't use it srsly ☺☺ " (2)

In the example tweet 2 we can see the use of **emojis** (even the case of repeating emojis), **slang** (the term catfish meaning "to lure someone into a relationship by means of a fictional online persona.") and **abbreviations** ("srsly" instead of "seriously").

III. Data set analysis

The used training data set contains **6838 tweets**, each having a vector of ones and zeros representing the presence of a given sentiment. Our initial analysis of this data set revealed that it is **unbalanced**, i.e. not all emotions are presence in equal amount among all tweets. The distribution of emotions among all tweets can be seen in Figure 12.

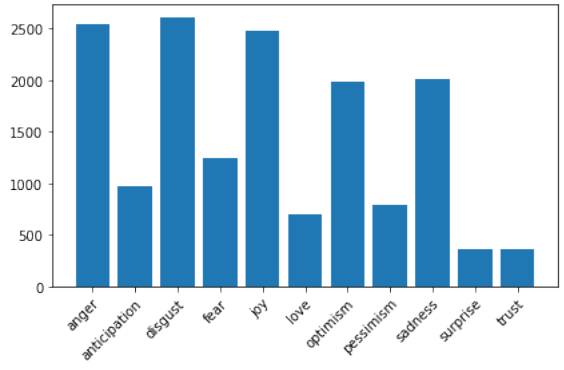

Fig. 1. Distribution of sentiment among all 6838 tweets of the training set.

We attribute this unbalance to the fact that some emotions have a **broader meaning** than others and, as such, can be applied to more tweets than some. Also, the meaning of some labels usually **overlap** with other, for example, joy is likely to be present in tweets that are also labeled as love, surprise or optimism.

IV. Approach

Solving the classification problem can be divided into two stages. The first is the task of representing the textual data in a structured manner, so that it can be interpreted by the classification algorithms. In this project we chose **two ways**

of representing the data. The first is to represent each tweet by a vector resultant of running a **TF-IDF** processing on the "processed" (see Subsection IV-A) tweets. The second way is to represent a tweet using **word embeddings**.

A. Preprocessing

Due to the reasons described in previous sections, **preprocessing** [14]–[20] takes on an important role in tweet sentiment analysis. By doing it, we can **reduce the amount of noise** existent in a tweet and **normalize the structure** of each tweet, preparing them for the classification algorithms.

The preprocessing step may be divided into three phases: **tokenization**, **inflection normalization** and **input structure building**. For the **tokenization** steps we used the **ekphrasis** library, that provides tokezination built specifically for social media text.
This tokenization included the following transformations:

- Normalizing mentions ('@POTUS' to '<user>')
- Normalizing language constructs (URL, email addresses, percentages, money designations, phone numbers, time descriptions, dates and numbers)
- Annotating all-caps words ('NICE' to 'nice <allcaps >')
- Annotating repeated tokens ('!!!' to '!<repeated>')
- Annotating elongated words ('happyyyy' to 'happy <elongated>')
- Annotating censored words ('f**k' to 'f**k <censored>')
- Annotating hashtags ('#BLESSED' to '<hashtag>blessed </hashtag>)
- Annotating emphasis ('I am *very* mad' to 'I am very <emphasis>mad')
- Unpack contractions ('can't' to 'can not')
- Spell correction
- Hashtag unpacking ('#MyGirl' to '<hashtag>my girl </hashtag>')
- Emoticon translation (':)' to '<happy>')
- Slang translation ('wtv' to 'whatever')
- Removal of stop-words

The **inflection normalization** step included either **lemmatization** or **stemming** and the processing of word/sentence negations. Although lemmatization is more complex than stemming, in some cases we achieved better results by using the latter.

For the last stage we decided on two ways of **representing a tweet**, based on the needed inputs of the algorithms. For the "base" algorithms (Naïve-Bayes, Support Vector Classification, Logistic Regression, ...) we represented each tweet by running a **TF-IDF based vectorization**, provided scikitlearn's TfidfVectorizer class. As such, we build a pipeline that started by applying the tokenization steps followed by a vectorizer. We also experimented with more modern, deep learning based, NLP techniques, representing each token with **word embeddings** [5], which are dense vector representations of words that capture semantic similarities between them. For this approach, we didn't perform any inflection normalization.

B. Classification

Our approach for solving the classification step can be separated into **two categories**. For the more **traditional approaches** we compared the performance of six algorithms ('Naïve-Bayes', 'Support Vector Classification', 'Logistic Regression', 'Perceptron', 'Decision Tree Classifier' and 'Random Forest Classifier') using Naïve-Bayes as our baseline. We used the algorithms provided in the Scikit-Learn library [4] which allowed for the customization of the algorithms' parameters. - We began by training all these algorithms with similar parameters then, after comparing the obtained results, we chose to optimize the algorithm that got the best value for the *Jaccard score*. We chose to optimize according to this metric because it was used in SemEval-2018 to rank the submitted solutions.

In the deep learning model, we used an architecture based on **long short-term memory (LSTM) neural network** [6], which is a common approach for many deep learning tasks. LSTM is a type of recurrent neural network [7], which are networks capable of processing sequential data, such as tweets. The model consists of a frozen embedding layer, initialized with pretrained **300 dimensional embeddings** trained using the **word2vec** algorithm on a large corpus of Twitter messages [8]. The embedded tweet representation is then encoded by a **bidirectional LSTM**, which processes the tweet both forwards and backwards. We concatenate the output of the last timestep from both passes and use it as a tweet representation, that we then feed to a **feedforward network without hidden layers**. From this last step we get a vector of size 11, we apply a sigmoid function to it (converting the real numbers to probabilities) and then consider that a **label exists when the probability is greater than or equal to 0.5**. We used the LSTM state vectors of size 256 and the final feedforward layer takes inputs of size 512. We used the **Binary Cross Entropy loss function** and the **ADAM optimizer** [9] with learning rate 0.001.

V. Experimental evaluation

A. "Base" Approach

In our preliminary tests we found out that setting the class weight as balanced improved our results considerably (from 0.375 to 0.480 with Logistic Regression). We also set the random state parameter to 0, in order to allow reproducible results when testing different solvers.

The comparison between all the aforementioned algorithms was made with these parameters set, when possible, and can be found in table I.

We focused our attention in optimizing the result for the **Logistic Regression** because, although the initial result was only marginally better than the Support Vector Classification, it was much more efficient allowing us to quickly iterate through different solutions.

Our objective for this research was to reach 0.500 with a simpler approach (bear in mind that the first place of this competition "only" reached 0.588 with a much more complex

	Jaccard	F1(macro)	F1(micro)	T. Time
Naïve-Bayes	0.203	0.173	0.318	1.177
Logistic Regression	0.480	0.531	0.609	2.439
Support Vector Class.	0.476	0.461	0.606	80.146
Perceptron	0.409	0.463	0.544	1.211
Decision Tree Class.	0.369	0.424	0.508	11.045
Random Forest Class.	0.369	0.348	0.503	46.396

TABLE I

EVALUATION OF ALGORITHMS USING STEMMING (USING NLTK'S SNOWBALL STEMMER), THE DEFAULT ALGORITHM PARAMETERS AND ALL OF THE TOKENIZATION TRANSFORMATIONS EXCEPT FOR THE SLANG DICTIONARY USING THE VALIDATION DATASET.

	Jaccard	F1(macro)	F1(micro)	T. Time
Naïve-Bayes	0.312	0.241	0.432	1.170
Logistic Regression	0.508	0.537	0.636	6.999
Embeddings + LSTM	0.551	0.453	0.663	19.708*
SemEval-2018 winner	0.588	0.528	0.701	-

TABLE II

EVALUATION OF OPTIMIZED ALGORITHMS ON THE COMPETITION'S TEST DATASET. THE LAST ROW REPRESENTS THE RESULTS OBTAINED BY THE BEST TEAM AT SEMEVAL-2018 (NTUA-SLP). * DEEP LEARNING MODEL TRAINED ON A NVIDIA GTX 1060 GPU.

algorithm using transfer learning and neural networks [8] and 0.457 using a pipeline with TF-IDF, a normalizer and a classifier).

Various different approaches were tried during this phase: using slang dictionary, word/sentence negation, custom stop-words, removing words with small absolute frequency, n-gram size variation, lemmatization vs stemming, instead of wrapping capitalize words with two tags, inserting one for each word (this worked because TF-IDF is order independent) and changing the classifier parameters.

Contrary to the expected, both the word negation (using the spacy library [13]) and the slang disambiguation didn't improve our results that much, the real improvement came from accepting bigrams (produced a worst outcome with the Naïve-Bayes) and changing the capitalize words tags. Changing the solver from L-BFGS [10] to SAG [11] and setting the C value (inverse of regularization strength) to 0.8 began by producing a slightly better score but we end up restoring the initial values because with the introduction of other optimizations these changes were only worsening the outcome. The difference found between the use of word lemmatizers and stemming (both from nltk's stem package [12]) was about 1 percentage point (0.469 vs 0.480), but the latter was used due to our customized stop-words being tailor-made to this kind of word normalization (and the training time was also reduced from 4.629 to 2.439 seconds).

In the end, the results were the ones present on table II and III, meaning that we reached our objective successfully.

As we can observe in the confusion matrix from table III (columns 1 through 4), the categories where the number of tweets were rarer, were the ones where the percentage of true negatives was higher. This can be explained by the fact that the predictions default to 0 meaning a higher rate of correctly predicting when the this sentiment is absent (0.9 in trust and 0.93 in surprise), but a small percentage of true positives (0.01 and 0.02 in the same categories - the ones with the least

	Confusion table				Measures			
	TN	FP	FN	TP	Prec.	Rec.	F	Sup.
Anger	0.55	0.10	0.11	0.24	0.71	0.69	0.70	315
Anticipation	0.72	0.14	0.08	0.06	0.29	0.40	0.33	124
Disgust	0.53	0.11	0.12	0.24	0.68	0.66	0.67	319
Fear	0.80	0.07	0.03	0.10	0.61	0.76	0.68	121
Joy	0.46	0.09	0.10	0.35	0.79	0.78	0.78	400
Love	0.75	0.10	0.05	0.10	0.50	0.65	0.56	132
Optimism	0.52	0.13	0.10	0.25	0.66	0.72	0.69	307
Pessimism	0.78	0.11	0.06	0.06	0.34	0.50	0.41	100
Sadness	0.59	0.11	0.12	0.18	0.62	0.60	0.61	265
Surprise	0.93	0.03	0.02	0.02	0.39	0.43	0.41	35
Trust	0.90	0.05	0.04	0.01	0.18	0.23	0.20	43

TABLE III

CONFUSION TABLE AND PERFORMANCE MEASURES FOR LOGISTIC REGRESSION USING THE BEST COMBINATION OF PARAMETERS AND PREPROCESSING TECHNIQUES IN THE VALIDATION DATASET. TN = TRUE-NEGATIVES; FP = FALSE-POSITIVES; FN = FALSE-NEGATIVES; TP= TRUE-POSITIVES; PREC. = PRECISION; REC. = RECALL; F = F-MEASURE; SUP = SUPPORT.

	Confusion table				Measures			
	TN	FP	FN	TP	Prec.	Rec.	F	Sup.
Anger	0.57	0.08	0.09	0.27	0.78	0.76	0.77	315
Anticipation	0.86	0.00	0.14	0.00	1.00	0.01	0.02	124
Disgust	0.54	0.10	0.10	0.26	0.72	0.72	0.72	319
Fear	0.84	0.02	0.05	0.09	0.79	0.65	0.71	121
Joy	0.50	0.05	0.10	0.35	0.88	0.77	0.82	400
Love	0.83	0.02	0.09	0.05	0.73	0.36	0.48	132
Optimism	0.56	0.09	0.09	0.25	0.73	0.73	0.73	307
Pessimism	0.88	0.00	0.10	0.01	0.70	0.07	0.13	100
Sadness	0.66	0.04	0.16	0.14	0.79	0.48	0.60	265
Surprise	0.96	0.00	0.04	0.00	1.00	0.06	0.11	35
Trust	0.95	0.00	0.05	0.00	0.00	0.00	0.00	43

TABLE IV

CONFUSION TABLE AND PERFORMANCE MEASURES FOR THE NEURAL NETWORK ON THE VALIDATION DATASET. TN = TRUE-NEGATIVES; FP = FALSE-POSITIVES; FN = FALSE-NEGATIVES; TP= TRUE-POSITIVES; PREC. = PRECISION; REC. = RECALL; F = F-MEASURE; SUP = SUPPORT.

examples).

The precision (column 5, table III) and recall (column 6 of the table III) (and, consequently the F-measure - column 7 of the same table), tell us the same story: the categories with higher frequency are the ones the model correctly predicts most often (0.78 for joy), while the ones with least examples is where the model is the least accurate and precise in (F-measure of 0.20 for trust).

There are, however, outliers. Fear has a surprisingly good F-score bearing in mind the reduced data set compared to other categories like disgust. This is due to the uniqueness of the words that appear under this label proved by the word cloud in the appendix.

B. Deep Learning model

The deep learning approach improved upon our best previous model (Logistic Regression) in the accuracy (Jaccard index) and micro averaged F1 scores, but had a worse macro averaged F1 score. The bad macro F1 score is due to the fact that this value is the result of averaging the F measure for each label without taking into account label imbalance, which greatly penalizes the score since there are labels (such as trust and anticipation) with very few examples in the dataset with an F measure of 0, as seen in table IV. A downside of this

method is that it needs more computational resources to train, taking a much longer time than Logistic Regression despite being trained on specialized hardware.

VI. Conclusion

In this paper we present a traditional and a deep neural network approach to sentiment analysis applied to a multilabel classification task.

We reached satisfactory results with both models taking into account the bigger picture. However, we would like to try methods of upsampling data in order to balance the dataset in future iterations of this work, as well as using weighted loss functions in the deep learning approach.

Another strategy would be the application of transfer learning from other similar datasets to our neural network, this way the unbalance wouldn't be as big of a problem as it is now.

Still, this kind of problem is not trivial, because the annotation of the examples can be very biased and some categories end up having some overlap. Finding tweets with similar categories classification isn't an easy task either, because their division ends up being a little arbitrary [3].

References

[1] Russell, S., Norvig, P. (2009). Artificial Intelligence: A Modern Approach, 3rd edition. PrenticeHall.

[2] Jurafsky D., Martin J. (2019). Speech and Language Processing, 3rd edition. Chapter 4: Naive Bayes Classification and Sentiment

[3] Semeval-2018 Task 1: Affect in Tweets. Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. In Proceedings of the International Workshop on Semantic Evaluation (SemEval-2018), New Orleans, LA, USA, June 2018.

[4] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jef-frey Dean. 2013a. Efficient Estimation of WordRepresentations in Vector Space. InICLR Work-shop Papers.

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural Computation, 9(8):1735–1780.

[7] Jeffrey L. Elman. 1991. Distributed representations, simple recurrent networks, and grammatical struc- ture. Mach. Learn., 7(2–3):195–225.

[8] Baziotis, C.; Athanasiou, N.; Chronopoulou, A.; Kolovou, A.; Paraskevopoulos, G.; Ellinas, N.; Narayanan, S.; Potamianos, A. NTUA-SLP at SemEval-2018 Task 1: Predicting Affective Content in Tweets with Deep Attentive RNNs and Transfer Learning. In Proceedings of the 12th International Workshop on Semantic Evaluation, New Orleans, LA, USA, 5–6 June 2018; pp. 245–255.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. InICLR, 2015.

[10] Zhu C., et al. (1997). L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization, ACM Transactions on Mathematical Software, Vol 23, Num. 4, pp. 550 - 560.

[11] Mark Schmidt, Nicolas Le Roux, Francis Bach (2016). Minimizing Finite Sums with the Stochastic Average Gradient. Mathematical Programming B, Springer, 2017, 162 (1-2), pp.83-112. ffhal-00860051v2f

[12] nltk.stem package documentation. Retrieved from: https://www.nltk.org/api/nltk.stem.html

[13] Spacy Industrial-Strength Natural Language Processing in Python. Retrieved from: https://spacy.io/usage/spacy-101

[14] Haddi E., et al. (2013). The Role of Text Pre-processing in Sentiment Analysis.

[0]All the results presented were obtained with the validation dataset, except the ones in table II which were obtained by submitting the results on the in the task websiteso that we could compare our values against the other teams. This allowed us to iterate quickly through solutions without the need of waiting for a new submission and also calculate more results like the F-measure, precision and recall.

[15] Prakash N., Amalanathan A. (2019). Data preprocessing analysis using Twitter data.

[16] Zin H., et al. (2017). The effects of pre-processing strategies in sentiment analysis of online movie reviews

[17] Singh, T., Kumari, M. (2016). Role of Text Pre-processing in Twitter Sentiment Analysis.

[18] Angiani G, et al. (2016). A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter

[19] Tutorial on text preprocessing using NLTK and Scikit-learn: Sentiment analysis of reviews: Text Pre-processing

[20] Social network text processing: Ekphrasis- Social Network tokenization

Appendix

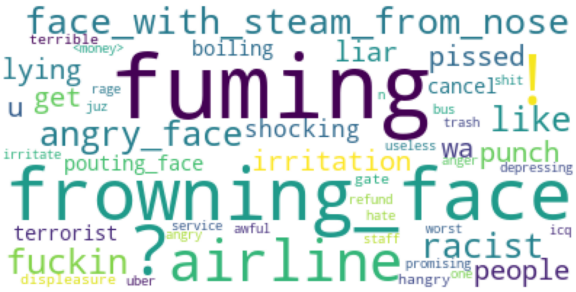

Fig. 2. Anger.

Fig. 3. Antecipation.

Fig. 4. Disgust.

Fig. 5. Joy.

Fig. 6. Fear.

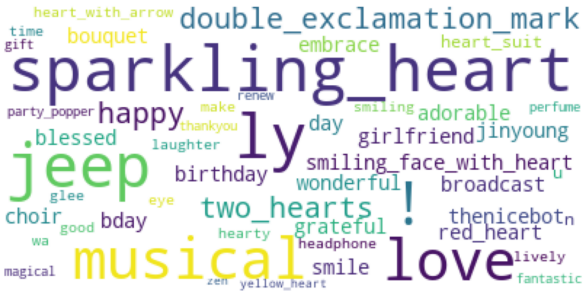

Fig. 7. Love.

Fig. 8. Optimism.

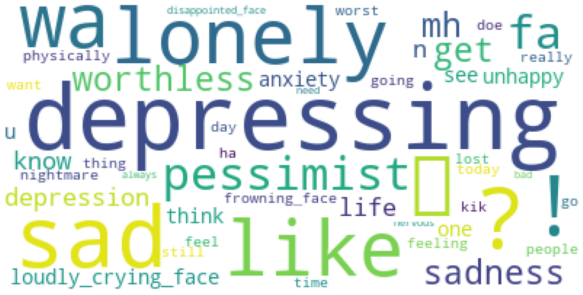

Fig. 9. Pessimism.

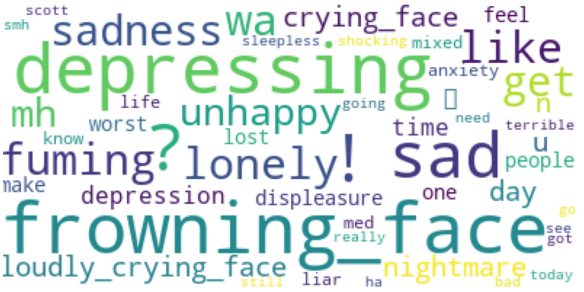

Fig. 10. Sadness.

Fig. 11. Surprise.

Fig. 12. Trust.