How do you feel, my dear (P7)

Vincenzo Conversano

Information Retrieval Project
University of Milan, Milan, Italy

Abstract. Recently, emotion detection in text has received attention in the literature on sentiment analysis. Detecting emotions is important for studying human communication in different domains, including fictional scripts for TV series and movies. The project aims at studying fictional scripts of several movies and TV series under the emotional profile. In particular, the task of the project is threefold:

– Create a model to predict emotions in text using available datasets as EmoBank or WASSA-2017 or Emotion Detection from Text as training sets (see below);
– Emotions may be represented either as categorical classes or in a continuous space such as Valence-Arousal-Dominance (see for example [10]);
– Exploit the model to study an emotional profile of the main characters in one of the movies included in the Cornell Movie–Dialogs Corpus;
– Study how this emotional profile changes in time along the evolution of the movie story and how it is affected by the various relations among the different characters.

Keywords: Natural Language Processing · Emotion Detection · Sentiment Analysis.

1 INTRODUCTION

This report belongs to the Information Retrieval field and presents a solution for predicting the emotion of a tweet. Emotion recognition is the process of identifying human emotion. People vary widely in their accuracy at recognizing the emotions of others. The use of technology to help people with emotion recognition is a relatively nascent research area. Generally, the technology works best if it uses multiple modalities in context.

In this work we will focus on textual data and employ Natural Language Processing (NLP) techniques. Specifically, we will use two twitter based datasets.

In order to reduce the complexity and balance the merged dataset, we discarded out tweets that were not labeled with a primary emotion according to Plutchik theory [8].

Motivated by the above, this work 1. addresses the problem of classifying tweets in five primary emotions, 2. considers state of art methods such as BERT, 3. is based exclusively on reliable and publicly available data ensuring reproducibility.

2 MACHINE LEARNING MODELS

This section describes the considered models used for the classification task. Aiming at experimenting both sequence learning techniques and classical ones we decided to use three different models.

2.1 Recurrent Neural Network based on static embeddings

Recurrent Neural Networks (RNNs) are special type of neural architecture characterized by recurrence: networks are endowed with feedback loops allowing to exhibit temporal dynamic behavior. They have successfully been applied to create sequence representations in NLP [1] [2] [4].

Recurrence enables RNNs to "remember the past". They try to mimic the human process of reading text in which a sequence is interpreted word by word, while keeping a memory of previously seen words.

Recurrent Neural Networks receive input in a sequence of time steps. The output of each time step is computed by combining the current input and the output from the previous time step. This output (vector) will be passed as an additional input to the next time step and this recurrence repeats until the end of the sequence is reached.

The layers that we used in our Recurrent Neural Network are:

- TextVectorization Layer [1]: a preprocessing layer which maps text features to integer sequences. The tensors of indices are then 0-padded to the longest sequence in the batch.
- Embedding Layer [2]: it turns positive integers (indexes) into dense vectors of fixed size.
- LSTM Layer [3]: a common LSTM unit [6] 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.
- Dense Layer [4]: it is composed of n hidden neurons, each one performs a weighted summation of its inputs and forward the output to neurons in the successive layer.
- Dropout Layer [5]: during the training phase, this layer randomly shut down some outputs of the previous layer in order to cut down the network complexity. It is considered one of the most basic techniques to avoid overfitting in neural networks.

[1] https://www.tensorflow.org/api_docs/python/tf/keras/layers/TextVectorization
[2] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Embedding
[3] https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
[4] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
[5] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

2.2 Multilayer Perceptron based on BERT encoder

A Multilayer Perceptron network employs multiple perceptrons distributed in a multi-layered scheme where the technique of Backpropagation is applied to update the weights and learn the characteristics of the training set [9]. The layers that we used in our MLP are:

– Input Layer [6]: this layer is composed of as many input neurons as the number of features. The goal of this layer is to bring the inputs into the system.
– KerasLayer [7]: this layer wraps a SavedModel (or a legacy TF1 Hub format) as a Keras Layer. It is used to wrap in our network a preprocessing layer and BERT encoder.
– Dense Layer [8]: see above.
– Dropout Layer [9]: see above.
– Activation Layer [10]: this layer applies a specified activation function to the output of the previous layer. In this work, we considered: linear function for the hidden layers and Softmax function for the output layer.

BERT [3] is a transformer-based neural language model designed to find contextualised bidirectional representations for text. It understands the context of each word in a sequence using the transformer multi-head attention mechanism. Simply put, we feed a sentence as input to the transformer's encoder and get the contextual embedding of each word in the sentence as output. These embeddings are then fed to the MLP to make predictions. In this work we used BERT tiny variant and fine tuned it.

2.3 Random Forest based on TF-IDF vectorizer

The specific classifier creates multiple weak-learners (that demonstrate high variance but low bias) and combine them in order to robustify the achieved predictions [5]. The Random Forest algorithm fits a plethora of Decision Trees (DT), typically on different bootstrap samples, while each tree elaborates on a random subset of features.

As the name suggests, TF-IDF vectorizer [11] computes for each word in a tweet its TF-IDF score. Then these vectors are fed to the Random Forest.

3 EXPERIMENTAL SETUP

This section describes 1. the considered tasks; 2. how is the dataset composed; 3. the data preprocessing step; 4. the creation of the holdouts; and 5. how results have been studied.

[6] https://www.tensorflow.org/api_docs/python/tf/keras/layers/InputLayer
[7] https://www.tensorflow.org/hub/api_docs/python/hub/KerasLayer
[8] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
[9] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout
[10] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Activation
[11] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

All the experiments were performed on a machine equiped with Intel Core i7-1065G7 CPU.

3.1 The considered tasks

In this work, we will develop models that, given a text are able to discriminate between five primary emotions: fear, joy, anger, surprise and sadness.

The task was tackled using all the Models described in Section 2. RNN working on features extracted using a static word embedding technique, MLP working on features extracted using BERT and RF working on features extracted using TF-IDF vectorizer. Please note that while the first two models are able to model the temporal dependencies of the sentences the last approach no.

Once the models have been learned they were employed to compute the emotion evolution of the main characters in the movie “Titanic”.

3.2 The dataset

The dataset has been created by aggregating data coming from two sources of information:

– WASSA-2017 [7]: the data is composed by tweets annotated for anger, fear, joy, and sadness intensities using a technique called best–worst scaling (BWS). Since all the tweets are labeled with a primary emotion we kept all the tweets.
– “Emotion Detection from Text” from Kaggle [12]: the data is basically a collection of tweets annotated with the emotions behind them. We have three columns tweet_id, sentiment, and content. In “content” we have the raw tweet. In “sentiment” we have the emotion behind the tweet. Since most of emotions in this dataset are not primary and “sadness” is already over-represented in WASSA we selected only tweets labeled with “surprise” and “anger”.

3.3 Text normalization

Text normalization is a fundamental step in a common NLP pipeline. In this work, we considered the following sub-steps.

Removing stop words and useless parts of the sentence Exploiting the neattext python package [13], the following parts of the sentences were removed: stop words, emails, twitter usernames, numbers.

Explaining emojis Emojis are very useful when we have to recognize an emotion in a text, thus, we can’t just get rid of them. Exploiting the neattext python package we explained the emojis i.e. replacing each one with a description.

[12] https://www.kaggle.com/datasets/pashupatigupta/emotion-detection-from-text
[13] https://jcharis.github.io/neattext/userguide/

Lowering case the text and removing multiple white spaces.

3.4 Models Parameterization

This section describes the parameterization process of the Models. Specifically, we followed two main approaches: Using Fixed Models and Using Meta Models.

Fixed Models Since the task at hand is not complex, we decided to employ very simple architectures for our Models. The architectures and hyper-parameters' settings are available on the github page [14].

Meta Models We tuned the RNN and Random Forest's hyper-parameters using a Random Search with 20 trials. MLP models with BERT were not optimized because of the limited computational resources. In order to tune the Models properly, we realized an External 10 fold Carlo Cross-Validation and tuned the hyper-parameters on an internal sub-split of the training set (more details in sub-section 3.5). The tools we used for optimization are: keras-tuner for RNN [15] and RandomizedSearchCV for Random Forest [16].

3.5 Cross Validation Setup

Fixed Models are evaluated through a 10 fold stratified cross-validation technique.

On the other hand, for Meta Models we realized an external 10 fold cross-validation and tuned the hyperparameters on a sub-split of the training set (validation set: 20% of training set). Thus, Meta Models are optimized on the validation set and then evaluated on the test set. In order to obtain statistically correct results and at the same time not depend on the specific training set, we should run a Nested Cross Validation procedure, but this is not considered in this work because it is too computationally heavy.

The Cross Validation procedures have been realized using StratifiedKFold class [17].

3.6 Figures of Merit

The metrics used to evaluate Models performances were chosen taking into account that we are in a multi-class setting and the classes are balanced. Specifically, we used:

– Accuracy: the ratio between the number of TP and the total number of examples;

[14] https://github.com/vincenzoconv99/how_do_you_feel_my_dear
[15] https://keras.io/keras_tuner/
[16] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
[17] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

Table 1. Performances achieved by the proposed approaches.

Method	Accuracy	F1-score macro	F1-score micro	F1-score weighted
RNN	0.77 ± 0.02	0.76 ± 0.02	0.77 ± 0.02	0.77 ± 0.02
MetaRNN	0.83 ± 0.01	0.83 ± 0.01	0.83 ± 0.01	0.83 ± 0.01
MLP (BERT)	0.81 ± 0.01	0.81 ± 0.01	0.82 ± 0.01	0.82 ± 0.01
RF (TF-iDF)	0.80 ± 0.01	0.80 ± 0.01	0.80 ± 0.01	0.80 ± 0.01
MetaRF (TF-iDF)	0.83 ± 0.01	0.83 ± 0.01	0.82 ± 0.01	0.83 ± 0.01

- F1-Score micro: calculate metrics globally by counting the total true positives, false negatives and false positives and then apply F1-score formula;
- F1-Score macro: the macro-averaged F1 score is computed using the arithmetic mean of all the per-class F1 scores;
- F1-Score weighted: the weighted-averaged F1 score is calculated by taking the mean of all per-class F1 scores while considering each class's support.

4 Results

In this section, we report the experimental results and compare the Models. In order to have a general idea of their performances, we showed in Table 1 how each one performed on average.

First, we observe that Meta Models perform better than the respective Fixed Models.

Second, we observe that even though we employed the tiny variant of BERT and not optimized the model, it can compete with other models. Thus, we expect that if we use a bigger variant and optimize hyper-parameters the performances will be boosted.

Lastly, we observe that the Random Forest approach based on TF-IDF vectorizer works well, meaning that the order of words appearance is not very important for the task at hand.

5 Titanic Case study

In this work, we studied how the emotional profiles of Rose and Jack change in time along the evolution of the well known movie "Titanic".

In Figure 1 and Figure 2 you can notice the predictions created by our three models for each of the five emotions.

Considering the script of the movie, the Model that adapts the best to the emotional evolution of Rose and Jack is the MLP based on BERT.

Specifically, we considered as sentences only the dialogues between Rose and Jack. During the initial and central part of the movie, the emotions in the interactions of the two characters are balanced: there are peaks of Joy, Anger and

Fear. In the last part of the movie, instead, the main emotions are Sadness, Surprise and Fear.

In general, the models are penalized by the fact that they were trained on a Twitter corpus, whose vocabulary is different from the one employed in relatively old movies, such as "Titanic". Indeed, the constant presence of Surprise peaks is unexplained.

Regarding the other two models, the Random Forest model is the only one not taking strong decisions, making it difficult for us to evaluate it. While the RNN predictions are strange because there is no Sadness in Jack emotions.

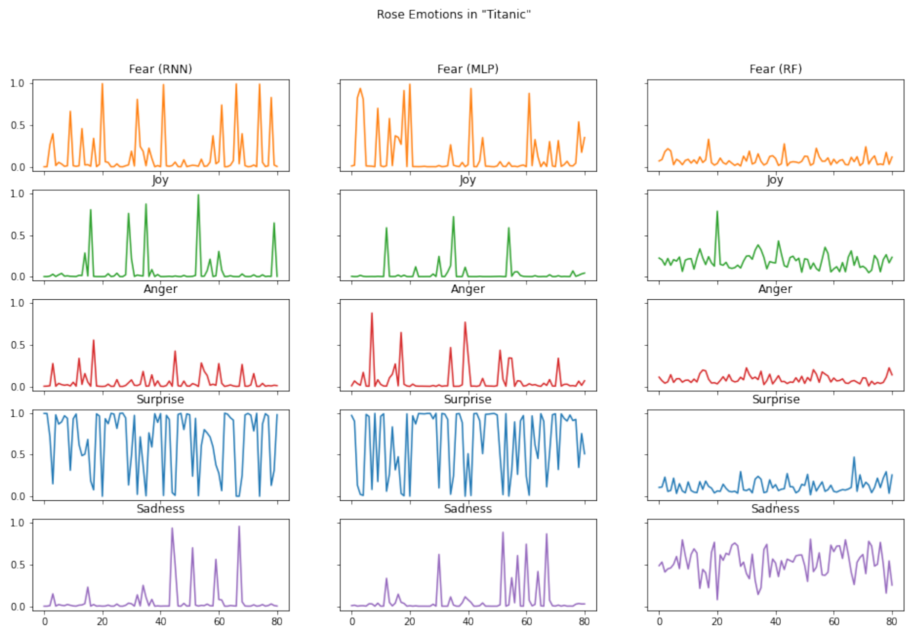

Fig. 1. Rose Emotion profile predicted by our three models.

6 Conclusions

This report described three different Machine Learning approaches for emotion detection from text: a RNN based on static embeddings; an MLP based on BERT embeddings; and a RF based on TF-IDF vectorizer. After extensive experiments, it was shown that all the models can tackle the task, and considering the encouraging results obtained by the MLP it is worth carrying out more experiments exploiting a bigger BERT variant and optimizing the network hyperparameters.

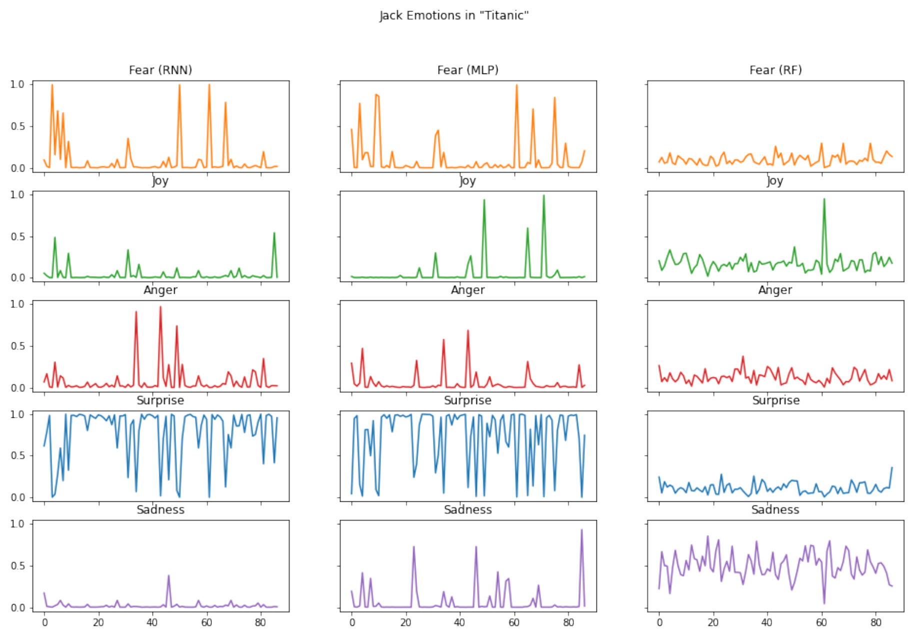

Fig. 2. Jack Emotion profile predicted by our three models.

It should be pointed out that the present experimental set-up is based on publicly available datasets, while the results are fully reproducible and the code available on github [18].

References

1. Cheng, J., Dong, L., Lapata, M.: Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733 (2016)
2. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
5. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on PAMI **20**(8), 832–844 (1998). https://doi.org/10.1109/34.709601, https://doi.org/10.1109/34.709601
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[18] https://github.com/vincenzoconv99/how_do_you_feel_my_dear

7. Mohammad, S.M., Bravo-Marquez, F.: Wassa-2017 shared task on emotion intensity. arXiv preprint arXiv:1708.03700 (2017)
8. Plutchik, R., Kellerman, H.: Emotion, theory, research, and experience, vol. 3. Academic press (1980)
9. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Representations by Back-propagating Errors. Nature **323**(6088), 533–536 (1986). https://doi.org/10.1038/323533a0, http://www.nature.com/articles/323533a0
10. Warriner, A.B., Kuperman, V., Brysbaert, M.: Norms of valence, arousal, and dominance for 13,915 english lemmas. Behavior research methods **45**(4), 1191–1207 (2013)