Course: **Text Mining and Sentiment Analysis**
Project 8: How do you feel my dear?

Andrea Pio Cutrera - 965591

May 8, 2022

Abstract

This project wants to develop a **deep-learning-based system** able to detect emotions from textual data. Some models, implemented from the standard libraries of Python, like Tensorflow and and Keras, are used to train and evaluate built-in neural networks architectures. The **best model**, selected through hyper-parameter tuning, is a recurrrent neural network with a pre-trained Embedding layer and Long-Short Term Memory layers which attains a pretty good external validity: **84.09% accuracy** on the test set. Once the model has been trained, it is used to predict emotions from the dialogues of the main characters in the movie ***"The Godfather" (1972) by Mario Puzo, Francis Ford Coppola***. Then it is evaluated the emotional status in time along the evolution of the movie story and how it is affected by the various relations among the different characters.

Keywords— Deep Learning, Text Mining, Emotion Detection

Contents

[1] **Link to the GitHub repo**: `https://github.com/andreacutrera/how_do_you_feel_my_dear`

[2] I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study

1 Introduction

Emotion detection is the field of sentiment analysis which makes the machines able to mimic one of the senses of human intelligence: the *human feelings*. This kind of artificial intelligence allows to extract and analyze the emotions from dialogues of real individuals. Machines have become pretty much capable of recognizing objects in images, videos and sounds, and even NLP (i.e. Natural Language Processing) helped in solving many tasks *"to aid computers understand and sometimes generate human languages"*[1].

In this little experiment I tried to find out how to build a **supervised classifier** able to distinguish in short dialogues *the emotions* that each individual is feeling while speaking. The model has been trained, validated and tested on the data of the **WASSA-2017** challenge[2] which provided on March 8 of 2017 raw corpus of tweets annotated with emotions and also an intensity value for that particular emotion. Three sets of *datasets* (train, development and test) are available in the website reachable from the url below reported. Once pre-processed we can get a final dataset of texts in which datapoints are annotated with the class of the emotion which would be our target variable to predict.

The **recurrent neural network** model used has few main features which allows us to make the learning from the sequences (of words) meaningful and memory preserving. The *embedding layer* maps each unique word in our sentences to a unique vector which *embeds the meaning* according to an arbitrary number of *features*. Then the *LSTM layer* (i.e. Long-Short Term Memory) can preserve important information from earlier parts of the sequence and carry it forward[3].

2 Data and Methodology

2.1 Data Description

WASSA-2017 Shared Task on Emotion Intensity (EmoInt) data available in the dedicated webpage[4] are essentially datasets just organized in **3 folders**:

- **Training** folder containing 4 datasets (one for each of the 4 emotions) with **textual data** from **real tweets** labelled with the corresponding emotion;
- **Validation** folder containing 4 datasets (one for each of the 4 emotions) with **textual data** from **real tweets** labelled with the corresponding emotion;
- **Test** folder containing 4 datasets (one for each of the 4 emotions) with **textual data** from **real tweets** labelled with the corresponding emotion.

Emotions in the WASSA-2017 data have the dual possibility of being treated as in a continuum or as in one of the 4 classes proposed. They are namely, and in alphabetical order: **Anger, Fear, Joy** and **Sadness**. We have decided to take into account the **discrete classes** indeed performing a *multi-class classification* of emotions.

2.2 Data Pre-processing

Once fetched the data they have been concatenated in a **unique dataframe**. This would allow to process all the datapoints all together in the same way and ultimately get the desired distribution

[1] Acheampong et al, 2020, "Text-based emotion detection: Advances, challenges, and opportunities": `https://onlinelibrary.wiley.com/doi/full/10.1002/eng2.12189`

[2] WASSA-2017 Emoint: `http://saifmohammad.com/WebPages/EmotionIntensity-SharedTask.html`

[3] LSTM working: `https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-\teach-a-network-to-remember-the-past-55e54c2ff22e`

[4] `http://saifmohammad.com/WebPages/EmotionIntensity-SharedTask.html`

between the train-test and validation. Below just a little snapshot of the data pre-processed which we got:

id	text	sentiment
11041	Nothing is more relentless than a dog begging ...	anger
40980	Ok it really just sunk in that I'm seeing the ...	sadness
41210	@MariamVeiszadeh #depressing it's so freaking ...	sadness
41365	I feel like a burden every day that I waste bu...	sadness
31085	@partydelightsUK it's 5679787. Cannot DM you a...	joy
30832	This tweet is dedicated to my back pain, which...	joy
20804	@StephenKing\n\nStephen King never once spoke ...	fear
30318	I found #marmite in Australia. `:)	joy
41289	doese reading help depression or is it another...	sadness
11340	#Scorpio's can withdraw to a quiet place after...	anger

2.3 Text Processing

In order to allow the functioning of a Natural Language Processing model we need to **clean and standardize the texts** of our tweets. *Emails, numbers, urls and usernames* have been **abruptly removed** since they are not useful for extracting any meaningful information for emotion detection. Instead **emoji** are really a powerful feature of the textual documents which could enable us to better predict emotions. So, they have been converted to a textual description and then handled through removing the separator in output (i.e. ":red heart:" to "red heart"). *Special characters, bad quotes, html tags punctuations, stopwords (english ones as default) and multiple spaces* are also removed in order to get a clean text. Thanks to the *remove special characters* function we have the possibility to manage hashtags as single words. As a last step texts have been stripped and lower-cased. Below reported the complete **cleaning pipeline**:

```
def clean_emoji_output(text):
    return re.sub(":", " ", text)

def strip_lowercase(text):
    return text.strip().lower()


def clean_text(data):
    data['clean_text'] = data['text'].apply(nfx.remove_emails)
    data['clean_text'] = data['clean_text'].apply(nfx.remove_numbers)
    data['clean_text'] = data['clean_text'].apply(nfx.remove_urls)
    data['clean_text'] = data['clean_text'].apply(nfx.remove_userhandles)
    data['clean_text'] = data['clean_text'].apply(demoji.replace_with_desc)
    data['clean_text'] = data['clean_text'].apply(clean_emoji_output)
    data['clean_text'] = data['clean_text'].apply(nfx.remove_special_characters)
    data['clean_text'] = data['clean_text'].apply(nfx.remove_bad_quotes)
    data['clean_text'] = data['clean_text'].apply(nfx.remove_html_tags)
    data['clean_text'] = data['clean_text'].apply(nfx.remove_punctuations)
    data['clean_text'] = data['clean_text'].apply(nfx.remove_stopwords)
    data['clean_text'] = data['clean_text'].apply(nfx.remove_multiple_spaces)
    data['clean_text'] = data['clean_text'].apply(strip_lowercase)
```

Once cleaned, our texts are **tokenized** into sequences of single words and then *further normalized* through **stemming**. **De-tokenizing** finally allows us to get a fully cleaned and normalized set of tweets. Below reported a snapshot of the dataset created insofar:

id	text	sentiment	clean_text	tokenize	tokenize_lemmatized	final_text
11041	Nothing is more relentless than a dog begging ...	anger	relentless dog begging food	[relentless, dog, begging, food]	[relentless, dog, begging, food]	relentless dog begging food
40980	Ok it really just sunk in that I'm seeing the ...	sadness	ok sunk im seeing goat hours wow face rolling ...	[ok, sunk, im, seeing, goat, hours, wow, face,...	[ok, sunk, im, seeing, goat, hour, wow, face, ...	ok sunk im seeing goat hour wow face rolling e...
41210	@MariamVeiszadeh #depressing it's so freaking ...	sadness	depressing freaking close	[depressing, freaking, close]	[depressing, freaking, close]	depressing freaking close
41365	I feel like a burden every day that I waste bu...	sadness	feel like burden day waste dont know bc discou...	[feel, like, burden, day, waste, dont, know, b...	[feel, like, burden, day, waste, dont, know, b...	feel like burden day waste dont know bc discou...
31085	@partydelightsUK it's 5679787. Cannot DM you a...	joy	dm dont follow delight party fail letdown	[dm, dont, follow, delight, party, fail, letdown]	[dm, dont, follow, delight, party, fail, letdown]	dm dont follow delight party fail letdown
30832	This tweet is dedicated to my back pain, which...	joy	tweet dedicated pain understand youthful spry ...	[tweet, dedicated, pain, understand, youthful,...	[tweet, dedicated, pain, understand, youthful,...	tweet dedicated pain understand youthful spry ...
20804	@StephenKing\n\nStephen King never once spoke ...	fear	king spoke left crushes freespeech publishing ...	[king, spoke, left, crushes, freespeech, publi...	[king, spoke, left, crush, freespeech, publish...	king spoke left crush freespeech publishing wo...
30318	I found #marmite in Australia. `:)	joy	found marmite australia	[found, marmite, australia]	[found, marmite, australia]	found marmite australia
41289	doese reading help depression or is it another...	sadness	doese reading help depression form escapism	[doese, reading, help, depression, form, escap...	[doese, reading, help, depression, form, escap...	doese reading help depression form escapism
11340	#Scorpio's can withdraw to a quiet place after...	anger	scorpios withdraw quiet place hurt think plan ...	[scorpios, withdraw, quiet, place, hurt, think...	[scorpio, withdraw, quiet, place, hurt, think,...	scorpio withdraw quiet place hurt think plan s...

2.4 Data Preparation

To set up our experiment we require the **random split** between **train and test set**. Part of training data have been retained as **validation set**, to get the best hyper-parameters. Test set would be used instead to measure the external validity of the trained and validated model (on never seen texts). Split has the following form:

- 5313 **training** *cleaned texts* with corresponding 5313 training *labels*;
- 865 **validation** *cleaned texts* with corresponding 865 validation *labels*;
- 924 **testing** *cleaned texts* with corresponding 924 testing *labels*.

Since our deep learning model would understand only numerically encoded information we need to manipulate a bit our texts and labels. Creating a **Tokenizer()** class with 10K words fitted on our train and validation texts allows us to get **padded numerical sequences** from texts which have been already split in an array form. Sequences have been padded with a maximum length of 50.

Labels instead have been **encoded numerically** first in the following alphabetical order:

- **Anger**: 0
- **Fear**: 1
- **Joy**: 2
- **Sadness**: 3

Once got the numerical mapping of the labels it has been performed a **one-hot encoding** which would enable us to implement a binary predictor for each of the 4 classes which we want to predict.

2.5 Model

The model we propose for *emotion detection* from texts is the **Recurrent Neural Network**. Instead of using the simple *Multi-layer Perceptron* which propagates the input signal to the output layer through n fully-connected hidden layers, it has been chosen to build the Neural Network Architecture with 3 main features:

- **Embedding**: a single layer in which we are able to get a vectorial representation of each word in the sequences;
- **CudNNLSTM**: 3 **bi-directional** *long-short term memory layers* which propagate the signal forward and backward trying not to lose the interdependence which exist among words in a sequence;
- **Dropout**: layers to regularize the learning phase.

Once texts have been cleaned and encoded numerically through the Tokenizer, we need a **clever** way to carry some meaningful information inside each vector which represents each word. An improvement to the simple Bag of Words (BOW) which can do that is the *keras embedding layer*. Inside the embedding we insert an **embedding matrix** which comes from a pre-trained model called **GloVe**. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space[5].

An LSTM or Long-Short-Term-Memory classifier is an artificial recurrent neural network which has both feedforward and feedback connections. In particular, once the GPU has been set in the runtime of Colab it is possible to execute the training phase in a **5x faster way**[6] than the CPU execution of LSTM.

3 Hyper-parameter tuning for Model Selection

Once the model architecture has been chosen, what we need is to understand which are the best hyper-parameters to be tuned. Performing a grid search over all possible combinations of values specified in each hyper-parameter we compare the validation accuracies across all the models. For simplicity I have set just 3 values for each of the 3 following hyper-parameters, computing at least 27 models:

- **lstm_1**: *cardinality of the first lstm layer* [50, 100, 150];
- **lstm_2**: *cardinality of the second lstm layer* [50, 100, 150];
- **lstm_3**: *cardinality of the third lstm layer* [50, 100, 150].

```
def build_model(hp):
    model = tf.keras.Sequential(
        [
            embedding_layer,
            Dropout(0.2),
            Bidirectional(CuDNNLSTM(hp.Choice('lstm_1', values=[50, 100, 150]),
                                    return_sequences=True)),
            Dropout(0.2),
            Bidirectional(CuDNNLSTM(hp.Choice('lstm_2', values=[50, 100, 150]),
                                    return_sequences=True)),
            Dropout(0.2),
            Bidirectional(CuDNNLSTM(hp.Choice('lstm_3', values=[50, 100, 150]),
                                    return_sequences=False)),
            Dense(class_num,
                  activation = 'softmax')
        ])


    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics='accuracy')

    return model
```

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Follow the link to get more information: `https://nlp.stanford.edu/projects/glove/`

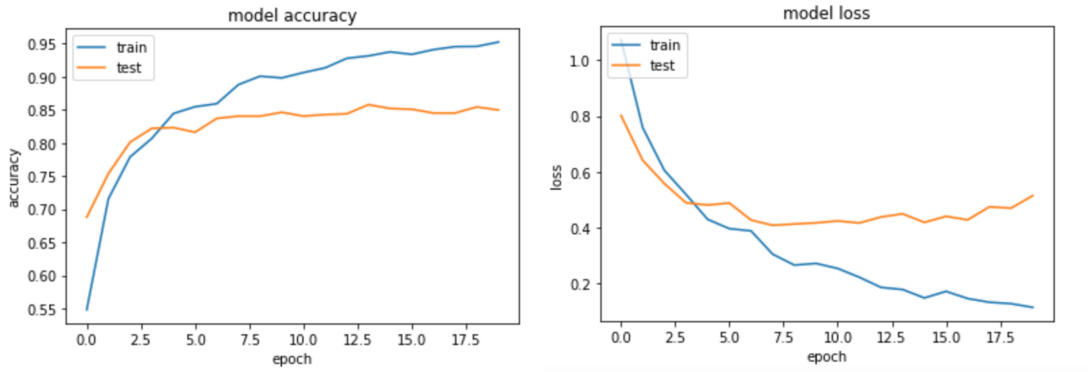
[6] Google Colab Notebook in which you can execute training phase faster on GPU: `https://colab.research.google.com/drive/16TOIflO9CPvh8WMYQRDCKT6dZ43PHoPM?usp=sharing`

3.1 Best Model

In the experiment it has been made, the model which have come out to be the best in the validation set is the one with the following combination of hyper-parameters:

- **lstm_1**: 50
- **lstm_2**: 150
- **lstm_3**: 50

Retraining the model with the configuration above specified, as we can see in the following graph the **best number of epochs is 14**, when validation accuracy reaches a stabilized value around **84%**.



In the last phase we just retrain the best model with the best number of epochs and evaluate it on the test set. The result is an **accuracy** of **85.39%**. Computing a 5-fold cross validation on train and validation sets concatenated we get an estimate of the accuracy around 80%.

Once the model has been trained, selected and at last evaluated we can save it into a *.h5* file which would retain the full structure and also the trained weights. In this way whenever we want to use our model we just need to load it from the file.

Another essential passage is to save the tokenizer fitted on the train and validation data. Saving it is essential to get a working embedding layer for subsequent model applications since the recurrent neural network classifier has been trained with a specific embedding layer.

4 *The Godfather* Emotions

As Wikipedia writes[7]: "***The Godfather*** is an American drama and **crime movie of 1972** directed by *Francis Ford Coppola*, who co-wrote the screenplay with *Mario Puzo*, based on Puzo's best-selling **1969 novel** of the same name. The film stars Marlon Brando, Al Pacino, James Caan, Richard Castellano, Robert Duvall, Sterling Hayden, John Marley, Richard Conte, and Diane Keaton. It is the first installment in The Godfather trilogy. The story, spanning from 1945 to 1955, chronicles the Corleone family under patriarch Vito Corleone (Brando), focusing on the transformation of his youngest son, Michael Corleone (Pacino), from reluctant family outsider to ruthless mafia boss.

In order to study the **emotional status** of main characters of the movie under discussion we relied on a large open database on movie dialogues called *Cornell Movie-Dialogs Corpus*. This corpus contains a large metadata-rich collection of fictional conversations extracted from raw movie scripts[8]:

[7]Look more here for the plot: `https://en.wikipedia.org/wiki/The_Godfather`

[8]Follow the link for the complete documentation: `https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html`

- 220,579 conversational exchanges between 10,292 pairs of movie characters;
- 9,035 characters from 617 movies;
- 304,713 total utterances

The structure of the database is pretty much articulated but for the purpose of our analysis we just need to process the **utterances table** filtering the movie we are going to deal with and also the dialogues of the 5 main characters chosen:

- **Don Vito Corleone**, the Godfather;
- **Tom Hagen**, the counselor (i.e. consigliere);
- **Michael Corleone**, the son of the Godfather approaching to become the heir of the family;
- **Peter Clemenza**, the main henchman of the family;
- **Sonny Corleone** the choleric son of the Godfather.

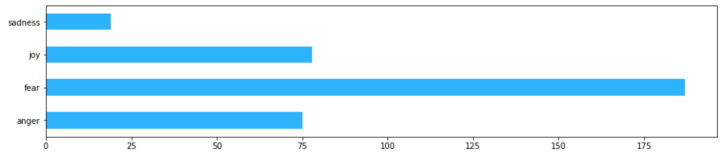
The dataset which contains the dialogues can be described as follow:

	id	text	character	conversation_id	reply_to
0	L593445	You owe the Don a service. He has no doubt th...	HAGEN	L593443	L593444
1	L593443	This is Tom Hagen; I'm calling for Don Corleon...	HAGEN	L593443	NaN
2	L593425	Yes.	HAGEN	L593422	L593424
3	L593423	Good. He never doubted you.	HAGEN	L593422	L593422
4	L593421	You owe the Don a service. In one hour, not b...	HAGEN	L593420	L593420

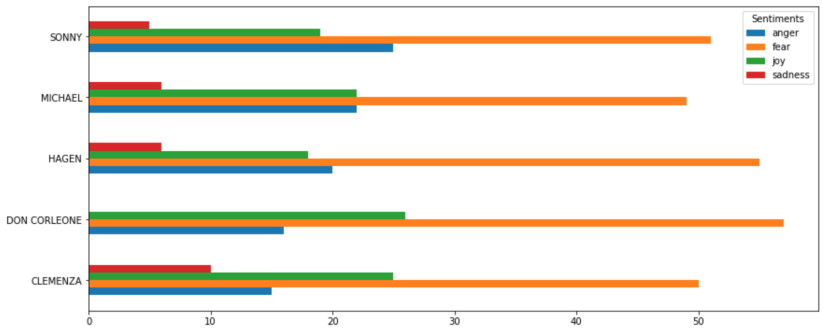
Once we fetched the portion of database of our interest we can apply our **cleaning pipeline** above used for the text processing. **Texts** are then converted to **padded sequences** through the Tokenizer

loaded. Loading the model we got trained in the first part of this experiment we can make **predictions of emotions** for each utterance of our character of the movie under study.

Below we can see how the predictions of the emotions from the textual documents are distributed. Since it is a drama, crime and thriller movie, the main emotion detected is indeed ***fear***.



We can also disaggregate the distribution among the selected characters, and we need to normalize counts in order to get comparable results. In this way values for the bars represent *shares* among all the emotions felt by each character.

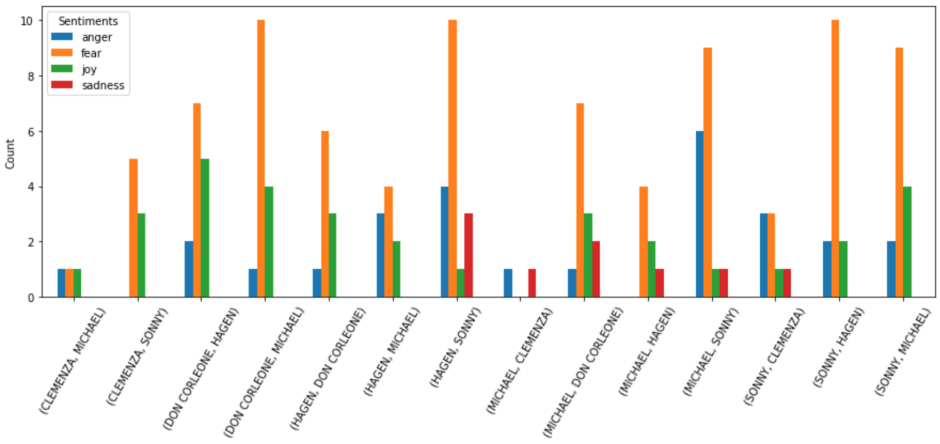


We can also check the emotions of characters in relation to the character they replied to. An interesting thing could be done by matching two equal temporary views of the dataframe used so far. Since the *id* of each line can be joined on the attribute *reply_to* we can get a new dataframe in which each row correspond to all the couples of speaker-respondent dialogues. In this way we can exploit the information to understand how the emotional profile changes in relation to who the responder or the speaker is.

Below a snapshot of our new dataframe:

	conversation_id	character_speaker	text_speaker	prediction_speaker	character_respondent	text_respondent	prediction_respondent
147	L592889	DON CORLEONE	Is it necessary?	fear	HAGEN	You understand him better than anyone.	fear
55	L592928	HAGEN	Hello Kay. Your father's inside, doing some b...	anger	MICHAEL	Thanks Tom.	fear
65	L592936	DON CORLEONE	I'm sure it's the most generous gift today.	joy	HAGEN	The Senator called--apologized for not coming ...	fear

From that, we can perform a grouping by the speaker getting the distributions of emotions of each possible combination of speaker-respondent.



To the other hand, sorting by the attribute *conversation_id* we can get the **evolution of the emotions** throughout the movie ongoing. A plot for each emotions is reported below. Aggregate counts of emotions are reported both for speaker and respondent. Separating the plots by emotions can help understanding hoe each emotion evolved in the movie story.