How do you feel, my dear

Fabio Brambilla[978983]

Università Degli Studi di Milano

Abstract. Emotion can be expressed in many ways that can be seen such as facial expression and gestures, speech and by written text. Emotion Detection in text documents is essentially a content - based classification problem involving concepts from the domains of Natural Language Processing as well as Machine Learning.

The project aims at studying fictional scripts of several movies and TV series under the emotional profile focussing on 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.

For the purposes of the project it has been decided to use a categorical representation of emotions.

Keywords: Emotion · Outcome prediction · Information retrieval

1 Data Sources

To obtain the data it was necessary to rely on two datasets containing:

- written texts with the related emotion represented categorically
- movie scripts with related information

1.0.1 Emotion Dataset As a dataset for emotions, the 'Emotion Detection from Text' dataset on kaggle.com was used, which contains a collection of 40000 English-language tweets annotated with the relevant emotion. The emotions are categorized into 13 classes and are:

- Anger
- Empty
- Boredom
- Enthusiasm
- Fun
- Happiness
- Hate
- Love
- Relief
- Sadness
- Surprise
- Neutral
- Worry

1.0.2 Script Dataset For the script dataset, the 'Cornell Movie–Dialogs Corpus' was used, containing an extensive metadata-rich collection of fiction conversations extracted from raw film scripts.

2 Emotion Prediction Approach

A machine learning approach with a Bi-LSTM model was adopted to predict emotion from text. In addition to this, five other models from the Sklearn library were tested, including Logistic Regression, MLP, Decision Tree, Random Forest, K-Nearest Neighbor.

2.1 Preprocessing

In order to train the model, the emotion dataset needs to be split into three different Dataframe, one for training the model, one for valuating the model in the training phase, and one for testing the trained model.

For text cleaning, a basic pre-processing library called text_hammer was used to remove:

- punctuation
- stopwords
- emails, HTML tags, website and unnecessary links
- contraction of words
- normalizations of words

Punctuation removal was necessary as those characters do not offer much semantic meaning to the sentence. As the dataset comprises of tweets, some special characters present, such as '@','#' needed to be removed. This whole process was essential so as to avoid training the model on irrelevant data and adversely affecting its performance.

In order to consider only the most relevant and frequent words within the dataset, a Tokenizer was used to count word frequency and keep only the 10,000 most frequent words within the corpus.

Since in the datasets, different sentences have different lengths, it means the number sequence made by texts_to_sequence will also have different lengths. in order to pass them in our model, we must make all of them of the same length.

pad_sequences is used to ensure that all sequences in a list have the same length. By default, this is done by padding 0 at the beginning/end (pre, post) of each sequence until each sequence has the same length as the longest sequence. If in case the sequence length is greater than maxlen, it also trims from the end.

After that, it was necessary to create a text embedding as textual data cannot be interpreted mathematically by machine learning algorithms. It was decided to use a pre-trained embedding model since the use case is generic and not specific to a certain type of corpus or task (e.g. news corpus). In an emotion detection task text generally express general human feelings or emotions hence it was decided to move ahead with pre trained models for vectorization of the corpus.

2.2 Word Embedding

The pre-trained model used for the purpose is 'glove-wiki-gigaword-100' contained in the gensim library. GloVe is an unsupervised learning algorithm for producing vector representations for words. It is trained on global word to word statistics from a large text corpora, and therefore the resulting vectors show very interesting linear relations of the vector. space.

For each training example, that is the tweets or sentences, is created a matrix consisting of GloVe vectors for every word (after cleaning). Finally, the embedding matrix had the shape as [10000, 100].

2.3 Label Encoding

The tweets were categorized into 13 emotions initially. Four emotions **anger**, **empty**, **boredom** and **enthusiasm** were present in small quantities which led to a high class imbalance. For this reason, it was decided to remove all rows that presented that emotion in the sentiment column.

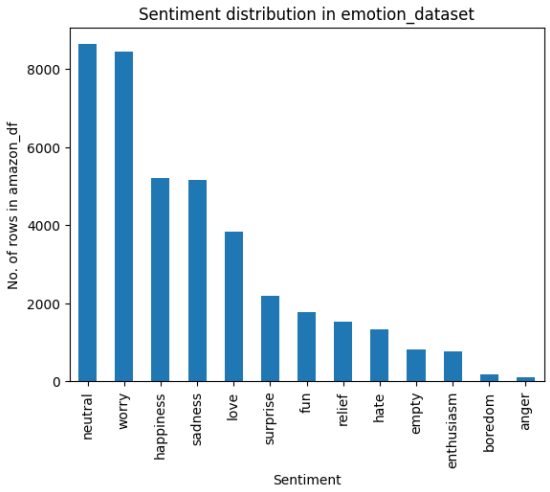

Fig. 1. Graph depicting the distribution of emotions in the dataset

Even the sentiment category in the emotion datasets need to be converted into some numbers in order to pass into the model. To covert text emotion into number it is use a dictionary that maps all emotions into an integer. After this, in order to be able to use the categorical_crossentropy loss function, all the labels are converted into categorical data using the "to_categorical" function provided by keras library.

2.4 Bi-LSTM Model

The last step was to make a Bi-LSTM based sentence model. A Bidirectional LSTM, or biLSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. BiLSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm (e.g. knowing what words immediately follow and precede a word in a sentence).

2.5 Training

The model was trained using 25 epochs with a batch size of 120 and an early stopping patience of 5. Results after training showed that the model started overfitting around the fifth epoch, stopping at the eighth epoch with a training accuracy = 43% and a validation accuracy = 37%.

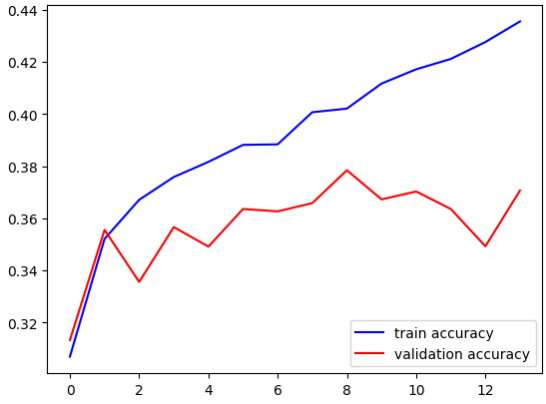

Fig. 2. Difference between training and validation accuracy throughout the epochs

After training the Bi-LSTM model, the other five models in the sklearn library were also trained.

2.6 Evaluation

The evaluation of all six models showed that the most accurate prediction was achieved by the Bi-LSTM model with an accuracy of 31% for 9 classes on the test dataset.

Model	Accuracy	Avg Precision (macro)	Avg Recall (macro)	Avg F1-score (macro)	Avg Precision (weighted)	Avg Recall (weighted)	Avg F1-score (weighted)
Logistic Regression	0.248691	0.078243	0.123559	0.077245	0.139153	0.248691	0.153440
MLP	0.248429	0.177542	0.125621	0.082807	0.218749	0.248429	0.158875
Decision Tree	0.176440	0.130238	0.130976	0.130188	0.177393	0.176440	0.176527
Random Forest	0.254450	0.183160	0.141888	0.122322	0.217356	0.254450	0.201172
K-Nearest Neighbors	0.176440	0.119603	0.118493	0.114719	0.166937	0.176440	0.168505
RNN LSTM	0.319372	0.366610	0.220660	0.196440	0.338476	0.319372	0.257470

Fig. 3. Accuracy of models on test data

	Precision
Logistic Regression	0.239529
MLP	0.156806
Decision Tree	0.180366
Random Forest	0.239005
K-Nearest Neighbors	0.181152
RNN LSTM	0.319372

Fig. 4. Accuracy of models on test data using cross validation

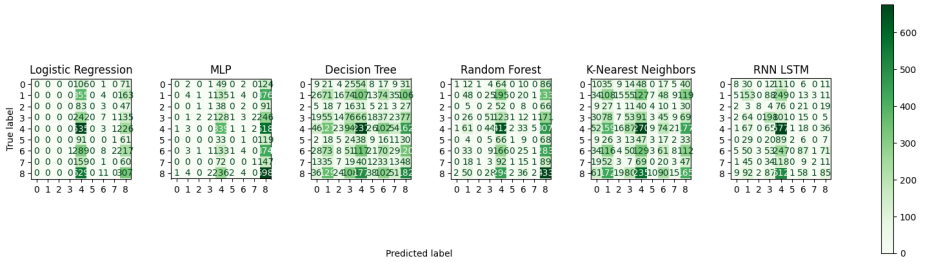

Fig. 5. Confusion matrix of the different models

3 Script Dataset

The corpus of films downloaded from the convokit library contains several dataframes containing different information on the film, script and actors.

The library provides three different dataframes, one with utterances, one for speakers, and one for conversations. In order to obtain a dataframe containing all the necessary information, a merge was performed between the three different dataframes and 8 attributes were stored for each record:

- Id
- conversation_id
- meta.movie_idx
- speaker
- meta.character_name
- meta.gender
- reply_to
- text

Next, the lines of all actors were cleaned using the text_hammer library as for the emotion dataset texts, and then a new column was added to the dataframe containing the vectorised text using the tokenizer previously used for the emotion dataset and the pad_sequence function provided by the keras.utils.data_utils library.

3.1 Script emotion prediction

The prediction of emotion was performed on all texts in the dataframe with all previously trained models. For each model, a column was added to the dataframe whose value corresponds to the emotion predicted by that model on the text.

4 Emotional Profile

Since the objective of the project is to evaluate the emotional profile of a main character of a film, it is necessary to extract all scripts of a character from a specific film.

First of all, since the script dataset is not sorted, the dataset is sorted by 'id' and 'conversation_id'. In this way, the entire script of a film is sorted by time sequence and it is easier to extract the rows of the time sequence of a character's script.

Subsequently, the speaker id is used to extract all the rows of that specific character and create a new dataframe with all his rows.

4.1 Emotions visualisation

As far as the visualisation of the actor's emotional state is concerned, it was chosen to observe:

- The actor's general emotional state as a percentage based on his lines
- The actor's emotional state during the course of the film
- Which emotions prevail during the exchange of lines with another character
- What are the prevailing emotions according to the gender of the character he/she is interacting with