SemEval 2019 Task3 EmoContext - A Shared Task on Contextual Emotion Detection in Text

Hussem Ben Belgacem, Thibaud Chominot, Sirine Kéfi, Carine Zhang, Guillaume Drapala-Bizouarn

Abstract
In this paper we present our results of emotion detection in a short conversation. The emotions that we aim to detect are "happy", "sad", "angry" and "others". The small dataset provided and the lack of ground breaking state-of-the-art method left us with multiple options. We chose to use neural networks to implement multiple models and then test a soft voting system and a binary classification system. Our best F1 score is 0.6858 and is achieved using a voting system on top of a binary classification and a GRU-CNN model.

Keywords
Sentiment analysis — SemEval2019 — NLP

Introduction

Use of social networks have increased text-based interactions. With such interactions the urge to detect emotions is more and more important to create tools that can use all the information in the conversation. The development of new machine learning techniques and the huge improvement of computer power in the last decades made some solutions impossible before more possible now.

The goal of this project was to detect the main emotion in a short conversation between 2 people. The structure of a conversation is :

- Person A
- Person B
- Person A

The goal is to classify a given conversation into one of the following four classes: "happy", "sad", "angry" and "others". However, one of the constraint to such project is the very limited amount of data available. Indeed, the less data we have, the less our model can train and it may even overfit. To overcome this constraint we can manually label more data but, the amount of data needed being huge, the extension of the dataset can be very time consuming. Here, we tried to overcome this by using a voting system and a binary classification.

In this paper we present our preprocessing, the models that we trained and the results of those models then, we present the experiments that we have done with the voting system and the binary classification.

1. Preprocessing

1.1 Text Preprocessing

The training and testing datasets contain respectively 30160 labeled and 2755 unlabeled dialogue. In order to train properly our model we need a proper text preprocessing including: tokenization, lemmatization, normalization, contraction removal and emoji handling. For this purpose we created two distinct preprocessing and we tested them on all of our models. Then, we kept the best preprocessing for a given model.

Our first preprocessing uses *nltk* for the lemmatization and normalization and it uses a *tweet tokenizer* for the tokenization. Also, for our normalization we remove all stopwords except the negations "not" and "no" as it can change the meaning of a sentence. For this preprocessing, we also tried to manage properly the emojis by converting them to unicode.With this preprocessing we obtain the following result:

I'm a dog person	youre so rude	Whaaaat why
dog person	rude	whaaaat

Our second preprocessing uses *ekphrasis* and its tweeter tools. In addition to the previous preprocessing, *ekphrasis* adds a spell checker and the gestion of the elongated and repeated words. With this preprocessing we obtain this result:

I'm a dog person	youre so rude	Whaaaat why
dog person	youre rude	

1.2 Word Embedding

Word embedding main idea is to map words into vectors of real numbers so we can use these vectors to add semantics information to the words. It aims to quantify and find semantic similarities between words by analyzing the context of each word, how they are used and their distribution properties in large samples of text. Most new word embedding techniques use neural networks to create the vectors containing the information about a word. You can then calculate the closest words to a word by using the cosine distance. However, some of these techniques are not without drawbacks. One example of these cons in the unsupervised learning of word embedding technique is the fact that the words "bad" and "good" are neighbours because their vectors are close based on the cosine similarity. The cause of this is that these two words are often used in the same way, in the same context in the large corpora of text used for training.

The two types of words embedding that we considered are Word2Vec and GloVe. The former is a neural network that use a single hidden layer and try to capture co-occurrence one window at a time. The main idea is to train a model on the context on each word. Whereas with GloVe you try to capture the counts of overall statistics of a word. The main idea behind it is to construct a co-occurrence matrix that count the number of time a word appears in a context then to factorize it because it takes a lot of place.
After some research we chose GloVe. This article[1] present the differences and results between multiples words embedding techniques such as Word2Vec, SSWE and 2 GloVe models trained with different sets of data (Twitter and Crawl). The article measured the accuracy and the macro-F1 results of each of these models using a training corpus of more than 12000 labels extracted from the SemEval Task 10 challenge. The results show that the GloVe models performs better than the SSWE and the Word2Vec models by 2 points in the accuracy score and by more than 4 points in the macro-F1 score.

In all of our models we added a trainable embedding layer containing the vectorization of the words that we used during training. When a word is unknown (ie not present in GloVe) we simply generate a random vector of 300 dimensions. Moreover, this embedding layer is of size 64x300 where 64 correspond to the maximum sequence length and 300 correspond to the size of the embedding.

1.3 Uterances fusion

We started by using models that handle the three dialogue phases separately like suggested in those papers[2][3] but because of the lake of computing power at our disposal the training was too long to complete. So, we decided to fusion the three dialogue phases into only one creating a simple string that we preprocess before giving it to our models. By doing so, the training time for one epoch reduced drastically from 30min to 2-3min.

2. Models and Results

2.1 GRU Model

This model uses the simple preprocessing created with *nltk*.

Table 1. GRU Model Architecture

Layers	Output Shape	Nb Params
Embedding	(None, 64, 300)	4248900
Dropout (0.2)	(None, 64, 300)	0
GRU	(None, 128)	164736
Dropout (0.2)	(None, 128)	0
Dense	(None, 32)	4128
Dropout (0.2)	(None, 128)	0
Dense	(None, 32)	132

Table 2. Table of Results for GRU Model

Labels	Precision	Recall	F1 score
Angry	0.5301	0.8800	0.6617
Happy	0.4956	0.7958	0.6108
Sad	0.5206	0.8080	0.6332
Micro-average	0.5156	0.8297	0.6360

2.2 LSTM-CNN Model

This model uses the simple preprocessing created with *nltk*.

Table 3. LSTM-CNN Model Architecture

Layers	Output Shape	Nb Params
Embedding	(None, 64, 300)	4248900
LSTM	(None, 64, 300)	721200
Conv1D	(None, 64, 32)	28832
Max Pooling 1D (2)	(None, 32, 32)	0
Flatten	(None, 1024)	0
Dropout (0.2)	(None, 1024)	0
Dense	(None, 4)	4100

Table 4. Table of Results for LSTM-CNN Model

Labels	Precision	Recall	F1 score
Angry	0.5721	0.8200	0.6740
Happy	0.5093	0.7746	0.6145
Sad	0.5479	0.8240	0.6581
Micro-average	0.5428	0.8058	0.6486

2.3 CNN-LSTM Model

This model uses the preprocessing created with *ekphrasis*.

Table 5. CNN-LSTM Model Architecture

Layers	Output Shape	Nb Params
Embedding	(None, 64, 300)	4248900
Conv1D	(None, 64, 32)	28832
Max Pooling 1D (2)	(None, 32, 32)	0
LSTM	(None, 300)	399600
Dense	(None, 4)	1204

Table 6. Table of Results for CNN-LSTM Model

Labels	Precision	Recall	F1 score
Angry	0.5378	0.8533	0.6598
Happy	0.5091	0.7887	0.6188
Sad	0.6144	0.7520	0.6763
Micro-average	0.5466	0.8010	0.6498

2.4 CNN-GRU Model

This model uses the preprocessing created with *ekphrasis*.

Table 7. CNN-GRU Model Architecture

Layers	Output Shape	Nb Params
Embedding	(None, 64, 300)	4248900
Conv1D	(None, 64, 32)	28832
Max Pooling 1D (2)	(None, 32, 32)	0
GRU	(None, 300)	299700
Dense	(None, 4)	1204

Table 8. Table of Results for CNN-GRU Model

Labels	Precision	Recall	F1 score
Angry	0.5747	0.8467	0.6846
Happy	0.5503	0.7324	0.6284
Sad	0.6992	0.6880	**0.6935**
Micro-average	0.5947	0.7602	0.6674

2.5 GRU-CNN Model

This model uses the preprocessing created with *ekphrasis*.

Table 9. GRU-CNN Model Architecture

Layers	Output Shape	Nb Params
Embedding	(None, 64, 300)	4248900
GRU	(None, 64, 300)	540900
Conv1D	(None, 64, 32)	28832
Max Pooling 1D	(None, 32, 32)	0
Flatten	(None, 1024)	0
Dropout (0.4)	(None, 1024)	0
Dense	(None, 4)	4100

Table 10. Table of Results for GRU-CNN Model

Labels	Precision	Recall	F1 score
Angry	0.5972	0.8600	**0.7049**
Happy	0.6084	0.7113	**0.6558**
Sad	0.6218	0.7760	0.6904
Micro-average	0.6078	0.7842	**0.6848**

3. Experiments

3.1 Binary Classification

For the binary classification we used three GRU-CNN. Each of those models were trained on a training set composed of only 2 labels: "sad" or "angry" or "happy" and "others". For the first model we labeled all the data that are note "sad" as "others". By doing so we obtained a binary dataset which we used to train this model. We then did the same thing for the two other models and the labels "happy" and "angry" and by applying a inversed voting system that will take the most uncommon response among these 3 models. We obtain the following results:

Table 11. Table of Results for binary classification

Labels	Precision	Recall	F1 score
Angry	0.6243	0.7867	0.6962
Happy	0.6667	0.6761	**0.6713**
Sad	0.5486	0.7680	0.6400
Micro-average	0.6102	0.7434	0.6703

Here we can observe that the F1 score for the label "happy" has increased significantly when the F1 scores for the "angry" label has slighty decreased. However, the main drawback is for the "sad" label that saw its F1 score decrease tremendously. In order to solve this lose in performance, we decided to try the voting system by taking the binary classification and the GRU-CNN Model as part of the vote.

3.2 Voting System

For the voting system we simply implemented a method that takes the most common response among all of our models. The results below are obtain without the binary classification:

Table 12. Table of Results for voting system (with all models)

Labels	Precision	Recall	F1 score
Angry	0.5752	0.8667	0.6915
Happy	0.5388	0.7817	0.6379
Sad	0.6178	0.7760	0.6879
Micro-average	0.5739	0.8106	0.6720

We then tried the same voting system but only with the binary classification model and the GRU-CNN model:

Table 13. Table of Results for voting system (with only the best models)

Labels	Precision	Recall	F1 score
Angry	0.6098	0.8333	0.7042
Happy	0.6131	0.7254	0.6645
Sad	0.6309	0.7520	0.6861
Micro-average	0.6169	0.7722	**0.6858**

References

[1] Dario Stojanovski & al. Twitter sentiment analysis using deep convolutional neural network. 2015.

[2] Soujanya Poria & al. Context-dependent sentiment analysis in user-generated videos. 2017.

[3] Amr El-Desoky Mousa & Bjorn Schuller. Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. 2017.