UAIC at SemEval-2019 Task 3: Extracting Much from Little

Cristian Simionescu, Ingrid Stoleru, Diana Lucaci, Gheorghe Balan, Iulian Bute, Adrian Iftene

Faculty of Computer Science, "Alexandru Ioan Cuza" University of Iasi, Romania
cristian@nexusmedia.ro, ingridstoleru@gmail.com,
{diana.lucaci22, balangheorghe1997}@gmail.com,
iulian.bute@gmail.com, adiftene@info.uaic.ro

Abstract

In this paper, we present a system description for implementing a sentiment analysis agent capable of interpreting the state of an interlocutor engaged in short three message conversations. We present the results and observations of our work and which parts could be further improved in the future.

1 Introduction

It is hard to understand emotions in textual conversations in the absence of voice modulations and facial expressions (Gupta et al., 2017). In sentiment analysis task researchers work on different levels of sentiment analysis: document (when are considered single topics documents), sentence (when single sentences are classified as positive, negative or neutral), entity or aspect (which deal with finding the aspects in the text and then classifying in respective aspect) (Liu, 2012).

Similar to sentiment analysis at sentence level, in last years tweets from Twitter were analyzed and classified (Zhang and Liu, 2017), (Kumar and Sebastian, 2012), (Mukherjee and Bhattacharyya, 2013) and (Singh and Husain, 2014). In the beginning, a binary classification was used, which linked opinions or opinions only to two classes: positive or negative. In (Pak and Paroubek, 2010) the authors proposed a model for classifying tweets in goals, positive and negative feelings using a classifier based on multinomial Naive Bayes to use features such as N-grams and tags POS (part- of-Speech). In (Parikh and Movassate, 2009) the authors have implemented two models, one based on the Naive Bayes bigrams model and one using Maximum Entropy to classify tweets. (Go et al., 2009) proposed a solution for analysing feelings on Twitter using distant supervision, the training data were tweets with emoticons, which are regarded as noise data. They built several well performing models using Naive Bayes, MaxEnt, and SVM. (Barbosa and Feng, 2010) have modeled an automated method to classify tweets using space features including retweets, hashtags, links, punctuation mark amazement in combination with words and POS features polarity. (Luo et al., 2013) have brought to light the difficulties that they encounter when they want to classify tweets. Spam and a variety of languages on Twitter make the task of identifying opinions very difficult.

In SemEval 2019, in Task 3, EmoContext: Contextual Emotion Detection in Text (Chatterjee et al., 2019), the organizers ask participants to classify users messages in one of four classes: *Happy*, *Sad*, *Angry* or *Others*. These are given in the context of another two previous messages. The textual dialogue is composed of short messages that appear to be from a chat conversation. In such a context, the users express their thoughts and ideas in a compact way. In this paper, we describe how we created one classifier to detect the sentiment of short messages such as tweets.

2 Related Work

2.1 Word Embeddings

In order to incorporate the meaning of the words in a software system that processes natural language, distributed representations of words in a vector space are used to achieve better results by grouping similar words.

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 355–359
Minneapolis, Minnesota, USA, June 6–7, 2019.

Previous work (Mikolov et al., 2013) introduces two architectures *CBOW* (Continuous Bag-of-Words) and *Skip-gram model* for learning word representations using neural networks. The later is more efficient for small training data, generating better representations for the infrequent words (Naili et al., 2017).

When choosing the best representations for a certain training dataset, one can either use pre-trained word embeddings that were built using large general corpora or train their own embeddings on a specific corpus which is similar to the type of data the model will be working with. The advantage of the first approach is that the representations only need to add without any additional computational cost, meanwhile, the second one requires a large enough corpus that can lead to meaningful representations that can capture both syntactic and semantic regularities. While vectors like Word2Vec, GloVe (Global Vectors for Word Representation) or fastText capture the most frequent semantic meaning of the words, training new representation on social media data can bring a number of advantages such as embedding the specific informal language that is used on these platforms and comprising numerous words that might not be very frequent in general corpora (Rotari et al., 2017) and (Flescan-Lovin-Arseni et al., 2017).

3 System Architecture

In this section we will present the systems developed for the EmoContext task.

3.1 Data Pipeline

Starting off, a critical characteristic in our architecture was the ease of configuration of different parameters of our system. We want our system to require little additional work and troubleshooting when changing, adding or removing pre-processing, feature extractions or post-processing techniques.

As seen in Figure 1, the system can take any configuration and order of pre-processing, feature extraction and post-process methods as well as a model to be fed the data.

Since a lot of small changes would sometimes occur on the later stages of the pipeline, we implemented an auto-save feature in all components of the system which will simply use the cached processed data up to the point of the last modified step in the pipeline.

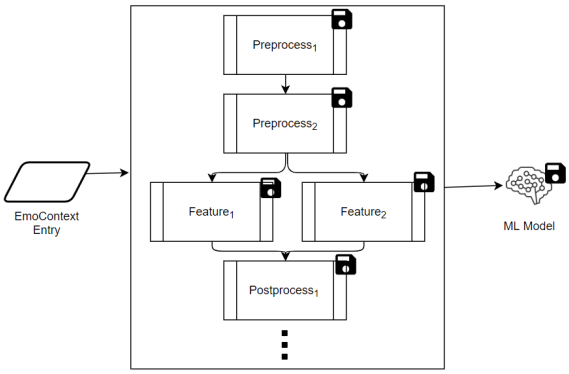

Figure 1: Pipeline structure

3.2 Data Processing

The dataset in from EmoContext task presented some clear challenges. Since we had to learn the expected sentiment of one of the interlocutors from a relatively small amount of text it was of utmost importance to remove noise from the data with minimal loss of potentially useful information. With such small amount of data ($\approx$ 4 words per message) in each column, we decided to concatenate all three messages in every entry in order to be able to infer more information from it.

3.3 Pre-processing Stage

In the pre-processing stage, we implemented a number of steps progressively remove noise such as non utf-8 encoded symbols or random punctuation or characters, from which no important information could be extracted, using regex rewriting rules. After which we transform all misspelled words to the closest correct English word (closest in terms of Hamming distance) while some words would be transformed wrongly, the system performed better when using the "corrected" dataset.

We considered emoticons to be important in determining the sentiment state of the communicating parties, as such we identified as many standard use emoticons. With these emoticons, we analyzed the distribution of where they appear. For example: ":)" appeared predominantly in entries labeled "happy". Using these we replaced each emoticon with the keywords: "happyemoticon", "angryemoticon", "angryemoticon" and "otheremoticon" respectively.

In terms of the actual noise reducing rewriting rules:

- Eliminating any elongated series of characters greater than two, to a size of two;
- Reducing any repetition of English symbols or punctuation marks to just one, since it would not lead to any loss of information but it will remove noise;
- Removal of spaces between punctuation marks;
- Removal of any number with more than one decimal;
- Rewriting Unicode characters into utf-8 equivalents or complete removals when not possible;
- Isolated characters get deleted;
- Deletion of ASCII emoticons.

We have observed that this combination of preprocesses leads to the best results, without eliminating too much or leaving too much noise. Extensive empirical experimentation was done to assert the performance of various combinations of preprocesses and parameters.

We have to keep in mind, that before applying these modifications the average length of the concatenated messages is around 12 words, afterward, it became around 10.

3.4 Feature Selection

For the actual features we wanted to use in our system, we have attempted a number of syntactic features we thought of extracting.

All of these features proved to be either not helpful in the aid of the model performance or detrimental in the sense that it left any model we attempted prone to overfitting, such as getting stuck in the local maxima of classifying all instances as ”others”.

As such, we chose to use embeddings as our only form of feature selection. We have tried to utilize pre-trained embeddings offered online such as GloVe, FastText and Word2vec.

Sadly, all of the embeddings we have attempted to incorporate into the system produced weaker results compared to training the embedding from scratch on the data.

For this, we tokenized the data and padded it to have 200 elements per list. Even though these vectors were trained on a relatively small corpus, due to the high usage of jargon, rare abbreviations and bad grammar which made our dataset very much different compared to the corpus used by any of the above mentioned pre-trained word embeddings this was most likely the cause of improved performance when our own embedding even though the corpus is extremely small compared to what would be required to create a good embedding.

3.5 Model

In constructing our machine learning model, we chose to use artificial neural networks with the use of the ”Keras” python library[1].

For the actual model of the system, we have made use of very simple and small architectures since any attempt of creating a deeper or wider artificial neural network models resulted in drastic overfitting. Even with other overfitting alleviating techniques such as regularization, dropout and batch normalization we had to stick to a shallow architecture. We suspect this is due to the fact that we trained our embedding on such a small dataset, perhaps if more similar data can be collected and a more general word vector is created, overfitting would also be reduced. As such, the model we are presenting does not suffer from overfitting but it is relatively shallow.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 159, 256)          4352000
_________________________________________________________________
bidirectional_2 (Bidirection (None, 256)               394240
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 1028
=================================================================
Total params: 4,747,268
Trainable params: 4,747,268
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Model

As seen in Figure 2, we used a trainable embedding layer of size 256 as input to fit on our training data. Next we used a single hidden layer of 128 Bidirectional LSTM cells with a 30% dropout and a tanh activation function. Finally outputting the result in a 4 neuron layer using the softmax function to learn the correct expected labels.

The model was trained using the RMSprop optimization algorithm.

	Metrics				
	micro F1	Accuracy	Precision	Recall	Sensitivity
Others	**0.92459089**	0.87620258	**0.95740783**	**0.89394911**	0.77644231
Angry	0.61855670	0.94626974	0.50209205	0.80536913	0.95432738
Sad	0.63508772	**0.96224360**	0.56562500	0.72400000	0.97356912
Happy	0.56877323	0.95788709	0.60236220	0.53873239	**0.98066986**
Average	*0.68675210*	*0.93565070*	*0.65687170*	*0.74051260*	*0.92125210*

Table 1: Submission metrics

4 Results

Using that simple model and an extensive meta-parameter tuning we were able to reach an average micro F1 score of 0.6895, this being the last submission we were able to upload during the workshop. See Table 1 for complete metrics of this submission, calculated by training the model 5 times, to factor in for randomness of shuffled data and weight initialization (all runs had comparably similar results).

We noticed that the greatest difficulty our system faces is correctly classifying instances belonging to the ”happy” class. As such, we should look into what data pre-processing we could use in order to decrease the high number of false-negatives.

Another potential improvement would be to add weights to the loss function based on the proficiency we observe the system to exhibit on each type of entry.

Applying the pre-processing we described previously we managed to boost that result to average micro F1 of 0.7362.

Both of these models were trained using a K-fold cross-validation with four splits and a batch size of 64.

What we observed time and time again, the main issue we faced was overfitting of the training data, as we can see when looking at the progression on the validation data, see Figure 3.

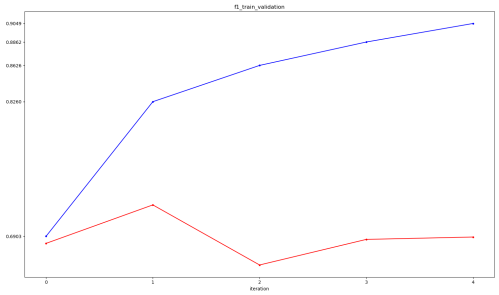

Figure 3: Training / Validation F1 - red validation data set, blue train data set

5 Conclusions

This paper presents the system developed by our group for the EmoContext task. The architecture of the system includes data processing, feature selection, and machine learning model. The results are promising, but they also expose the need for more experiments that should be done in this field in the next period.

For the future, we believe a CNN approach could prove fruitful. As well as a different or deeper network and configuration while using a similar pre-processing process which we believe is the main contributor to our relatively successful result.

Another direction worth investigating would be a Mixture of Experts approach, using various variations of the system even if they prove sub-optimal individually, such as *The currently proposed system*; *A system using a pre-trained embedding*; *Three sub-systems each trained to only classify one of the classes*; *A system with three input layers, one for each message reply, removing the concatenation of the text pre-processing*; *A system which only looks at the emoticons present in the text*.

Acknowledgments

This work is partially supported by POC-A1-A1.2.3-G-2015 program, as part of the PrivateSky project (P_40_371/13/01.09.2016).

References

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy

[1] https://keras.io/

data. volume 2, pages 36–44.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*, Minneapolis, Minnesota.

Iuliana Alexandra Flescan-Lovin-Arseni, Ramona Andreea Turcu, Cristina Sirbu, Larisa Alexa, Sandra Maria Amarandei, Nichita Herciu, Constantin Scutaru, Diana Trandabat, and Adrian Iftene. 2017. warteam at semeval-2017 task 6: Using neural networks for discovering humorous tweets. *SemEval@ACL 2017*, pages 407–410.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6.

Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal. 2017. A sentiment-and-semantics-based approach for emotion detection in textual conversations. *CoRR*, abs/1707.06996.

Akshi Kumar and Teeja Mary Sebastian. 2012. Sentiment analysis on twitter. *IJCSI International Journal of Computer Science Issues*, 9.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan Claypool Publishers.

Tiejian Luo, Su Chen, Guandong Xu, and Jia Zhou. 2013. *Sentiment Analysis*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR Workshop*.

Subhabrata Mukherjee and Pushpak Bhattacharyya. 2013. Sentiment analysis : A literature survey.

Marwa Naili, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala. 2017. Comparative study of word embedding methods in topic segmentation. *International Conference on Knowledge Based and Intelligent Information and Engineering Systems*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*.

Ravi Parikh and Matin Movassate. 2009. Sentiment analysis of user-generated twitter updates using various classication techniques.

Razvan-Gabriel Rotari, Ionut Hulub, Stefan Oprea, Mihaela Plamad-Onofrei, Alina Beatrice Lorent, Raluca Preisler, Adrian Iftene, and Diana Trandabat. 2017. Wild devs at semeval-2017 task 2: Using neural networks to discover word similarity. *SemEval@ACL 2017*, pages 267–270.

Pravesh Kumar Singh and Mohd Shahid Husain. 2014. Methodological study of opinion minng and sentiment analysis techniques. *IJSC International Journal of Soft Computing*, 5.

Lei Zhang and Bing Liu. 2017. *Sentiment Analysis and Opinion Mining*. Springer US, Boston, MA.