NL-FIIT at IEST-2018: Multiple neural approaches for emotion recognition

Samuel Pecar, Michal Farkas, Marian Simko, Peter Lacko, Maria Bielikova
Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
{samuel.pecar, michal.farkas, marian.simko, peter.lacko
maria.bielikova}@stuba.sk

Abstract

In this paper, we present neural models submitted to Shared Task on Implicit Emotion Recognition, organized as part of WASSA 2018. We propose a Bi-LSTM architecture with regularization through dropout and Gaussian noise. Our models use three different embedding layers: GloVe word embeddings trained on Twitter dataset, ELMo embeddings and also sentence embeddings. We see preprocessing as one of the most important parts of the task. We focused on handling emojis, emoticons, hashtags, and also various shortened word forms. In some cases, we proposed to remove some parts of the text, as they do not affect emotion of the original sentence. We also experimented with other modifications like category weights for learning and stacking multiple layers.

1 Introduction

Both text reconstruction and sentiment analysis are well studied and highly practical areas of research in the natural language processing area. Recently, there have been significant advances and improvements (Buechel and Hahn, 2017), at least partly due to the wider adoption of neural networks (Köper et al., 2017).

As it is, Implicit Emotion Recognition, as proposed by organizers of WASSA 2018 workshop (Klinger et al., 2018), can be seen both as a text reconstruction and as a sentiment analysis task. This is possible because, in this task, sentiment of a sentence should be equal to the missing word. In practice, the difference is marginal, nevertheless for both these tasks bi-directional LSTMs are widely used.

In recent years, there have been many competitions, papers and shared tasks dealing with emotion recognition and classification (Mohammad et al., 2018; Mohammad and Bravo-Marquez, 2017). Dealing with noisy and ungrammatical user-generated text can be also challenging in other high-level NLP tasks like summarization (Pecar, 2018).

In this paper, we present a neural network architecture with special focus on the preprocessing phase. We believe preprocessing can have significant impact on accuracy of each system in natural language processing. We explored many setups and also different types of regularization as dropout, Gaussian noise, kernel and activity regularization – L1 and L2, and also recurrent dropout within LSTM cells. We also experimented with three different types of embedding layers – GloVe, ELMo and various sentence representations. Finally, we explored impact of different setups on model accuracy.

2 Preprocessing

We are aware that preprocessing of input is one of the most important phases in natural language processing. This need is also highlighted when using user generated content which is more difficult to process. We can distinguish our preprocessing in a few stages displayed in Figure 1. We also evaluate different setups of our preprocessing in the results section.

Word-level cleaning Word-level cleaning consists of several rules to handle various forms of words in language. Especially, we focused on handling short forms of auxiliaries and also its negative variations. We split negative auxiliaries into its full form (e.g. *don't* as *do not*, *isn't* as *is not*). We also handled non negative auxiliaries and expanded them into their full form (e.g. *'ll* as *will*). In analysis of original dataset we decided to also omit some of the words which do not affect classification (e.g. @username, http://url.removed). The [NEWLINE] sign was replaced by sentence

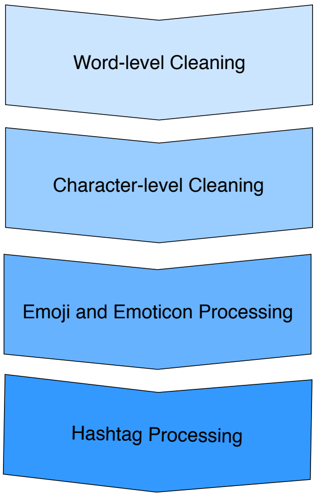


Figure 1: Preprocessing pipeline

endings followed by space.

Character-level cleaning Similarly to word-level cleaning, this phase consists of several rules operating on character-level preprocessing. We can describe this preprocessing in several categories, such as: currency handling, character escaping, replacing, and removing. In currency handling, we replaced signs for pounds, dollars, euros and yens with its word form. Other currency signs were replaced with the word 'currency'. Character replacing consists of unification of different forms used for apostrophe and as quotation marks. In character escaping, we surrounded characters like apostrophe, quotation mark, colon, dash, and caret with white-spaces to separate them from words and make tokenization easier. Finally, we decided to remove other unmentioned punctuation marks. We also considered removing all numbers as they often don't determine any sentiment.

Emoji and Emoticons Processing Handling emoticons and emojis is a more extensive part in our preprocessing phase. The first step consists of replacing emoticons (punctuation, numbers and characters used to create pictorial icons) with their emoji equivalent (only one unicode character). In phase of handling emojis, we removed all characters which modify original emoji with gender or skin color. We also tried to categorize emoji into categories, (Figure 2). This step helped us to reduce amount of emojis used in text. We replaced emojis symbolizing sport, moon, earth, animal, fruit, food, lag, music, flower, plant, drink, dress, money with their category word surrounded by colon. Another categories were produced by unification of different emojis with similar sense.

☺️☻😊😌🙂	🙂
😀😁😆😄😃😸😺	😀
☹😞😔🙁	🙁
♥❤♡💟💝💜💛💚💙🖤💘💗💖💕💓💞💌	💜
😗😙😚😍😽😻😘	😘
😮😯😲🙀	😮
😰😟😦	😦
😏	😏
😜😝😛	😛
🤣😹😂	😂
😿😢😭😥😪😢	😢
😠😾😤👿😡	😡
👬👭👫	👫
✔	✅
🌞	☀
🎊🎉🎈🎂🎆🎇	🎉

Figure 2: Emoji categorization

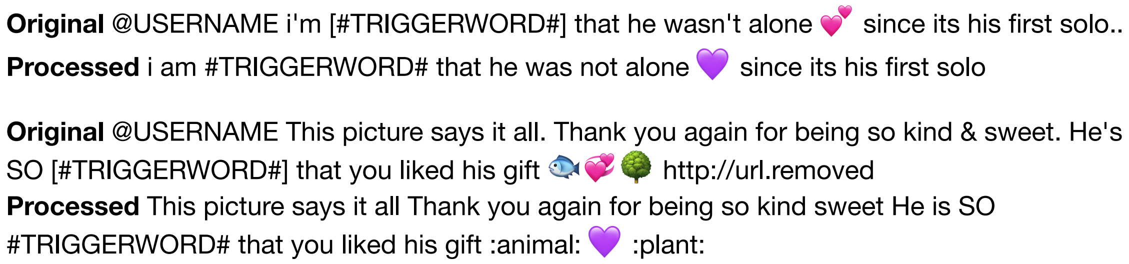
Finally, we surround all emojis with white-spaces. This step can significantly help in tokenization, as emojis were sometimes recognized as part of words and also group of emojis were recognized as one token.

Hashtags Processing In the[1] phase of handling hashtags, we only replaced those hashtags which can be found in word embeddings in their form without hash. We suppose removing other, unknown hashtags should be also considered as one of the step in hashtag handling. We also considered splitting hashtags into words but some of the separated words can bring different sentiment as the original hashtag and we decided to omit this step.

Examples of preprocessed texts are displayed in Figure 3. We can see examples of each preprocessing step described in this chapter.

3 Model

In this shared task, we experimented with many different setups, based on a different embedding layer and also different neural layers on top of an embedding layer.



Original @USERNAME i'm [#TRIGGERWORD#] that he wasn't alone 💕 since its his first solo..
Processed i am #TRIGGERWORD# that he was not alone 💜 since its his first solo

Original @USERNAME This picture says it all. Thank you again for being so kind & sweet. He's SO [#TRIGGERWORD#] that you liked his gift 🐟💞🌳 http://url.removed
Processed This picture says it all Thank you again for being so kind sweet He is SO #TRIGGERWORD# that you liked his gift :animal: 💜 :plant:

Figure 3: Examples of preprocessed sentences

3.1 Embedding layer

We experimented some of the commonly used embedding layers like Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) or ELMo (Peters et al., 2018) but also sentence embeddings like Universal Sentence Encoder (Cer et al., 2018) and InferSent (Conneau et al., 2017). For encoding input words, we used various pre-trained word embeddings available online like GloVe embeddings [1]. With GloVe embeddings, we experimented with domain specific embeddings trained on Twitter data but also with embeddings trained on data extracted by CommonCrawl. We have also experimented with recent ELMo embeddings, however these experiments were done additionally, after submission deadline.

Both GloVe and ELMo embeddings were included in model in such way that they can be further fine-tuned. However, available implementation of InferSent is done in Torch and integrating that model into Keras model proved to be problematic, this is mainly due to their use of custom layer. Hence, we encoded sentences into their vectors outside of the model and stored it in a separate file.

3.2 Hidden layers

Embedding layer is followed by several Bi-LSTM layers, when sentence embeddings were not used. We have experimented with up to two stacked layers. In case of sentence embedding models, we have experimented with several architectures with varying number of layers, however in the end we have settled on simple feed-forward network with one hidden layer, which uses parametric rectified linear unit.

We experimented with a different size of each layer and also a different number of stacked layers.

While using 1024 units in each of hidden layers was the best option, we also tested a much bigger model with sentence embeddings containing more units within each layer and also more stacked layers.

3.3 Activation

We experimented with several different activation functions, but their impact was either insignificant or obvious. Hence, all our result use the same configuration as far as activation functions are concerned.

In LSTM cells, we use typical activation functions – hard sigmoid and hyperbolic tangent. In case of models that utilize sentence embeddings, we are using parametric rectified linear activation, although its contribution is uncertain.

Since we are using categorical crossentropy as our loss function, choice of softmax activation function for the last layer is natural.

3.4 Regularization

In training stage, we discovered a problem with over-fitting on training dataset and some regularization was needed. We experimented with different types of regularization like dropout, Gaussian noise and also applying L1 and L2 norm to different types of regularization, such as: bias, kernel, recurrent or activation. We found out that using combined L1 and L2 regularization often causes learning to stop. Although, we fine-tuned L1 and L2 weight parameters, it failed to achieve better results than a model without this kind of regularization.

Unsurprisingly, we have utilized early stopping technique to halt training when validation accu-

[1] https://nlp.stanford.edu/projects/glove/

racy has not improved at least by 0.001% two times in a row. Although we experimented with different configurations, such as monitoring validation loss and tweaking patience, we did not observe any improvement and in most cases we even observed detrimental effect.

While L1 and L2 regularization had no positive effect on accuracy of the model, application of dropout and Gaussian noise significantly improved accuracy of tested models. We experimented with different setups and the most accurate combination was with the use of dropout with rate of 0.3 after each layer or replacing dropout after embedding layer with dropout with rate 0.2 and Gaussian noise on the embeddings with standard deviation 0.2. That being said, we have found out that applying these regularizations on sentence embeddings proved to be more challenging and the same settings often were too much for the neural network to handle.

4 Evaluation

In this section, we briefly summarize evaluation metrics for this task and also basic information about used dataset and embeddings. Later, we describe different setups of our model.

For evaluation, standard measures like precision, recall and f-score were used. Then micro and macro measures were counted. As final official result macro F1 was taken.

4.1 Dataset

Dataset for emotion recognition shared task consists of tweets where emotion word was removed. Dataset contains six different categories: anger, disgust, fear, joy, sadness, and surprise. Train dataset contains approximately 150 thousands of tweets and test contains more than 30 thousands of tweets. Detailed information can be found in main paper of shared task (Klinger et al., 2018).

4.2 Results

Our experiments show that the effect of LSTM size is apparent up to 1024 units after that it has negligible or even detrimental effect. Similarly, our experiments with two-layer Bi-LSTM achieved worse or same results as single-layer only. Our results are shown in Table 1.

Various variants of GloVe, even those trained on Twitter data, did not show much variance. On the other hand, ELMo embeddings did slightly improve our results as was expected.

Model	**P**	**R**	**F1**
GloVe			
Bi-LSTM-256	0.6	0.598	0.599
Bi-LSTM-2x256	0.601	0.599	0.6
Bi-LSTM-1024	0.657	0.655	0.655
Bi-LSTM-2x1024	0.643	0.64	0.638
ELMo			
Bi-LSTM-1024	0.665	0.666	0.665
Bi-LSTM-2048	0.661	0.661	0.661
Bi-LSTM-2x1024	0.666	0.665	0.664
Sentence embeddings			
InferSent	0.564	0.536	0.537
USE-small	0.504	0.501	0.5
USE-large	0.544	0.544	0.542

Table 1: Comparison between different models.

There are several possible reasons why sentence embedding methods failed. First, the provided dataset is actually quite substantial and does not need such methods. Secondly these methods cannot work well on Twitter data. Finally, the way we have included both InferSent and USE into our models does not enable fine tuning. Of course we cannot rule out a bug in our code as well.

Preprocessing had far greater impact on results than fine tuning our model. Details shown in Table 2 clearly demonstrate that text cleaning with emoji processing can improve classification of emotion. In the setup *Text cleaning*, only word-level and character-level cleaning were used. In the *Emoji processing* setup, we used previous features along with emoji processing and in *Hashtag processing* we used only text cleaning with hashtag processing. Finally, in the last setup, all previous setups were combined.

Setup	**P**	**R**	**F1**
No preprocessing	0.630	0.626	0.626
Text cleaning	0.645	0.639	0.641
Emoji processing	0.657	0.655	0.655
Hashtags processing	0.648	0.647	0.647
Combined	0.657	0.655	0.655

Table 2: Comparison of different preprocessing setups.

To see how our best model performed on different classes, we can take a look at Table 3. It is apparent that we have achieved best results on 'joy' class and worst results on 'anger' class. Precision metric of class 'surprise' is particularly noteworthy, due to it being considerably lower than other classes. This suggests that our model often classified other labels as 'surprise' class.

Confusion matrix, shown in Table 4, depicts accuracy of our official results (columns represent predicted classes while rows represent true labels). Quite surprisingly, false positives are more or less balanced across all categories. Nevertheless, we can see that our model rarely misclassified 'joy' as 'disgust', 'fear' and 'disgust', and vice versa. We can glean more findings from this confusion matrix, but they may be just a noise.

5 Conclusion

In this paper, we discussed different neural models for emotion recognition based on word and sentence embeddings followed by stacked Bi-LSTM layers and dense layers, respectively. We also discussed need of preprocessing that can significantly improve accuracy.

We observed that false negative and also false positive examples were equally distributed between classes. We tried also set sample weights for classes with the best and worst F1, but no combination brought any overall improvement.

In our preprocessing, we also removed all numbers as they do not contain any sentiment. After this preprocessing some of the sentences can be recognized as the same. Interesting point of research can be deduplication of these examples and examination of overall impact of these duplicated examples.

Acknowledgments

References

Sven Buechel and Udo Hahn. 2017. Emobank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 578–585. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Roman Klinger, Orphée de Clercq, Saif M. Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Brussels, Belgium. Association for Computational Linguistics.

Maximilian Köper, Evgeny Kim, and Roman Klinger. 2017. IMS at EmoInt-2017: Emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Copenhagen, Denmark. Workshop at Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 34–49. Association for Computational Linguistics.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Samuel Pecar. 2018. Towards opinion summarization of customer reviews. In *Proceedings of ACL 2018, Student Research Workshop*, pages 1–8. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Label	**TP**	**FP**	**FN**	**P**	**R**	**F**
anger	2717	1713	2077	0.613	0.567	0.589
disgust	3013	1368	1781	0.688	0.628	0.657
sad	2793	1722	1546	0.619	0.644	0.631
joy	3893	1193	1353	0.765	0.742	0.754
surprise	3124	2297	1668	0.576	0.652	0.612
fear	3345	1578	1446	0.679	0.698	0.689
MicAvg	18885	9871	9871	0.657	0.657	0.657
MacAvg				0.657	0.655	0.655

Table 3: Official results over classes.

Class	anger	disgust	fear	joy	sadness	surprise
anger	2717	345	437	310	397	588
disgust	352	3013	196	146	457	630
fear	314	169	3345	223	275	465
joy	303	119	266	3893	337	328
sadness	388	364	234	274	2793	286
surprise	356	371	445	240	256	3124

Table 4: Confusion matrix of official results.