NL-FIIT at SemEval-2019 Task 3: Emotion Detection From Conversational Triplets Using Hierarchical Encoders

Michal Farkas, Peter Lacko
Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovicova 2, 842 16 Bratislava, Slovakia
michal.farkas@stuba.sk, peter.lacko@stuba.sk

Abstract

In this paper, we present our system submission for the EmoContext, the third task of the SemEval 2019 workshop. Our solution is a hierarchical recurrent neural network with ELMo embeddings and regularization through dropout and Gaussian noise. We have mainly experimented with two main model architectures: simple and hierarchical LSTM network. We have also examined ensembling of the models and various variants of an ensemble. We have achieved microF1 score of 0.7481, which is significantly higher than baseline and currently the 19th best submission.

1 Introduction

Sentiment analysis has a long and successful history in the context of natural language processing. As with the majority of the problems in this domain, we have seen a gradual shift towards solutions based on neural models. Nowadays, such models can be readily used as a part of a larger solution, for example to analyse communication on social networks.

Then, perhaps unsurprisingly, the EmoContext task (Chatterjee et al., 2019b) with its format is highly evocative of the social networks. That is, it consists of conversational triplets, where the task is to correctly guess the emotion category of the last conversational turn.

Over the years, there was a number of different sentiment analysis and recognition competitions, workshops and shared tasks (Klinger et al., 2018; Rosenthal et al., 2017), however the conversational nature of the data is not common.

Our system is based on the recurrent neural networks, both simple and hierarchical architectures. We have experimented with various techniques and hyper-parameters, such as regularization, class weights, embeddings, strength of dropout and added noise. Finally, we have improved our results by creating an ensemble, with various voting methods and sample reweighing.

2 Approach

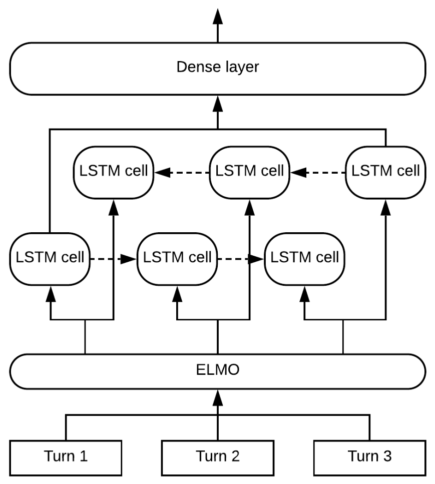
The first step in any natural language processing system is to preprocess the input data so it can be easily understood by the system. Since preprocessing was the major part of our previous work (Pecar et al., 2018), we have decided to prioritize work on the model instead. Nevertheless, we need to do some kind of preprocessing, hence we used the readily available Ekphrasis (Baziotis et al., 2017) tool.

We have experimented with both standard GloVe embeddings (Pennington et al., 2014) and more sophisticated ELMo embeddings (Peters et al., 2018). The improvements contextual embeddings, like ELMo, can bring are already well known and there is little need for additional comparisons, however the model that uses standard embeddings is considerably faster and hence more useful for quick experiments. Since training ELMo can be quite time consuming and resource intense, we have opted for pretrained models that are part of the AllenNLP library (Gardner et al., 2017).

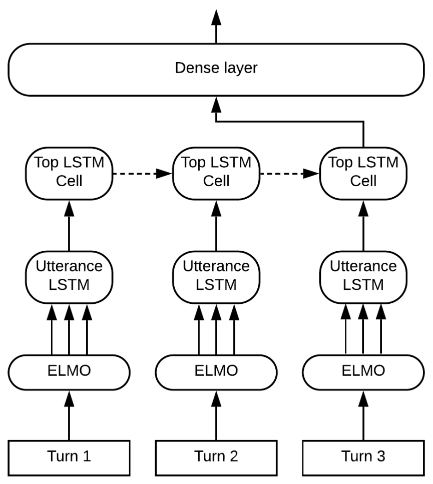
It should be noted that the character sensitive nature of the ELMo embeddings should help with typos, which were not fixed by preprocessing, and other similar errors.

2.1 Model

We have experimented with two main model variants. First model variant uses a simple bi-directional LSTM encoder (Hochreiter and Schmidhuber, 1997), second variant uses a hierarchical encoder, both can be seen in the figure 1. In both cases, the encoders are followed by dense



a) Simple encoder model.



b) Hierarchical encoder model.

Figure 1: Architecture of the main model variants.

layer which outputs probability distribution of labels.

Simple encoder works on concatenated utterances, utterances are part of a single string separated by semicolon. During the development we have also ran several runs only on the last utterance.

Hierachical model consists of two different encoders, at the utterance level and on the dialog level. This is fairly common practice in the dialog system domain (Serban et al., 2016). The utterance encoder is a bi-directional LSTM which encodes utterances into their representations. This utterance representation is then sent to the dialog encoder, which is a uni-directional LSTM.

Much of our model was dictated by a lack of GPU memory [1], we had to prioritize what to include in the model. Fully-connected layer as a dialog encoder, separate encoder for each turn, dense layer as a top encoder and attention mechanisms, did not achieve significantly better results and in some cases resulted in insufficient memory.

In the end, we had better results with the simpler, larger model rather than with the more complex, smaller model.

2.2 Training

We have opted to use Adam optimizer due to its far better performance on the hierarchical model where stochastic gradient descent did not perform up to our expectations.

To properly regularize the model we have used the dropout (Srivastava et al., 2014) combined with the gaussian noise applied to word embeddings. Since we did not use multi-layer LSTMs we did not apply dropout in any other place in the model.

In the case of ensembles, we scheduled 90% reduction of the learning rate at the third epoch. This was done in the hopes of achieving less variance in the performance of various runs. Since a single model is trained considerably faster than an entire ensemble, we did not schedule learning rate change, as we were able to pick the best model or average from a variety of runs.

The class imbalance in the dataset, both for the train, validation and test set, was an issue we had to deal with. We have opted for class reweighing instead of a weighted sampling, due to ease of implementation. We have experimented with various weight setups.

2.3 Ensemble

To improve our results and help with significant variance in performance of different experiment runs, we have used an ensemble of multiple models. It should be noted that they are the exact same model, trained several times.

We have experimented both with voting through summation of the probabilities and with stacked classifier on top of the ensemble results. Stacked classifier is a single dense layer with the rectified linear activation and softmax. As input it takes a tensor of shape *(batch_size, ensem-*

[1]We used a single RTX 2070 with 8GB of memory

Model	Size	Input data	Embeddings	MicroF1
Hierarchical	3076/1024	all	ELMo	**0.733**
Hierarchical	2048/1024	all	ELMo	**0.7213**
Hierarchical	1024/512	all	ELMo	**0.7172**
Simple	2048	concatenated	ELMo	**0.7123**
Simple	4096	concatenated	ELMo	**0.7079**
Simple	2048	last only	ELMo	**0.7076**
Hierarchical	2048/1024	all	GloVe	**0.6394**
Simple	4096	concatenated	GloVe	**0.639**
Simple	2048	concatenated	GloVe	**0.6312**
Hierarchical	1024/512	all	GloVe	**0.6294**
Hierarchical	3076/1024	all	GloVe	**0.6291**
Simple	2048	last only	GloVe	**0.5984**

Table 1: Comparison of various model variants.

Size	Voting	Reweighing	Max	Average	Combined
3	sum	No	0.7371	0.7265	**0.7481**(+0.0110/+0.0216)
5	sum	No	0.7426	0.7295	**0.7454**(+0.0028/+0.0159)
5	sum	Yes	0.7369	0.7272	**0.7430**(+0.0061/+0.0158)
3	sum	Yes	0.7306	0.7144	**0.7381**(+0.0075/+0.0237)
5	sum	Yes*	0.7281	0.7191	**0.7373**(+0.0093/+0.018)
3	stack	No	0.7248	0.7103	**0.7326**(+0.0078/+0.0223)
5	stack	Yes	0.7338	0.7270	**0.7310**(-0.0028/+0.004)
3	stack	Yes	0.7336	0.7257	**0.7289**(-0.0047/+0.0032)

Table 2: Comparison of various ensemble setups.

Distribution	Weights	MicroF1
train set	1.0/1.0/1.0/1.0	0.6924
test set	0.25/0.25/0.25/1.7	0.733
balanced_1	1.56/1.56/1.56/0.5	0.6888
balanced_2	1.56/1.56/1.56/0.3	0.6651
test w/o others	1.33/1.33/1.33/0.	0.2429

Table 3: Effect of different class weight setups.

*ble_size*num_labels)* and outputs a tensor of shape *(batch_size, num_labels)*.

To ensure that the single models in the ensemble will specialize on different samples, we have included the option for sample weight rebalance, based on their performance on the already trained models. However, error in the code caused that the rebalance calculation took into account only the last model and that the sample weights were gradually rising for the latter models in an ensemble. This was fixed only after the submissions were closed.

3 Evaluation

In this section, we cover metrics used, our experiments and analysis of our results. All results in this sections are achieved on the test set.

For evaluation we have modified code that is the part of the starter kit (Chatterjee et al., 2019a). Out of all metrics this function calculates we have primarily used microF1 score, which is the score reported in all our tables.

3.1 Results

In our experiments, we have explored a variety of different models, setups, ensembling approaches and effects of class weights. If not specified otherwise, models are using categorical crossentropy and following parameters:

- batch size: 32
- gaussian noise after embedding layer: 0.5 for simple, 3 for the hierarchical model
- dropout after embedding layer: 0.5 for simple, 0.6 for the hierarchical model

Class	Precision	Recall	F1
Angry	0.6948	0.8020	0.7445
Happy	0.7303	0.6866	0.7078
Sad	0.7687	0.8240	0.7954
Micro Average	0.7281	0.7692	0.7481

Table 4: Summary of the best submission.

Our first set of experiments are targeted at the model architecture and the effect of the used embeddings, results can be seen in table 1. In this table, all results are an average of three different runs. Unsurprisingly, the most significant effect is from the type of embeddings used. The effect of the rest of the factors seems to be in this order: model/input and size. At least for the ELMo embeddings, hierarchical models universally outperformed simple models. For the GloVe embeddings there is no clear separation, however the differences are rather small and if we averaged from more experiment runs a distinction could appear. The model that takes only the last turn into account was last when compared with the same embeddings, however we expected a more pronounced difference.

The next batch of experiments examines setup of our ensembles as seen in the table 2. In these experiments each row represent a single run of the entire ensemble. The combined score is the score of the ensemble after voting, in the parentheses we see change in respect to the max and average of the constituent models. Since the flawed reweighing does not seem to have a significant effect we have decided to left these experiments in. The experiment with asterisk, done after the submissions were closed, was run with the rebalancing fixed and while it shows second best improvement it is hard to tell if this is just a noise or a real effect. The most significant finding of this batch is that the stacked classifier performed rather poorly compared to the summation. For the stacked classifier, in two cases out of three, the combined score is actually worse than the best model in an ensemble.

Lastly, we have taken a look at the effect of different class weights, which can be seen in table 3, where the first column signifies distribution resulting from the given reweighing. We have experimented with ignoring others class due to the way the evaluation is done. The effect of such rebalancing is that all of the samples belonging to the 'others' class is classified as one of the other classes, which causes extremely high recall. Since the test set distribution is closer to the distribution of the train set than to the balanced dataset, trying to reweigh the data to obtain a balanced dataset[2] is worse than doing nothing. The best results are obtained when the reweighed distribution is the same as the test set, even though score is not averaged over the 'others' class.

Detailed summary of our best submission can be seen in the table 4.

4 Conclusion

In this paper, we have presented our models and experiments for emotion detection in conversational triples. We have also discussed results, various setups and model variants.

The bulk of our work was focused on simple vs. hierarchical models. We observed that the hierarchical model outperformed simple model. Additionally, simple model with only the last turn of conversation was only slightly worse than the model with the entire context.

We managed to improve our results by using an ensemble of multiple instances of the same model. During our experiments summation method of result combination proved to be superior to using stacked classifier. Interestingly, the stacked classifier could be perhaps a way to adapt model to different class distribution.

Our efforts were hindered by insufficient amount of memory on our GPU, thus we could not include every feature we wanted into our model. Perhaps, using GPU with more memory available, we could achieve slightly better results.

The source code of our system is available at GitHub [3].

Acknowledgements

This work was partially supported by the project Development of research infrastructure STU, project no. 003STU-2-3/2016 by the Ministry of Education, Science, Research and Sport of the Slovak Republic, Slovak Research and Development Agency under the contract No. APVV-16-0213 and APVV-16-0213 and by financial contribution from the STU Grant scheme for Support of Young Researchers.

[2]balanced_2 slightly supresses the others class

[3]`https://github.com/michalfarkas/nl-fiit_emocontext`

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317.

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

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Roman Klinger, Orphee De Clercq, Saif 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*, pages 31–42. Association for Computational Linguistics.

Samuel Pecar, Michal Farkaš, Marian Simko, Peter Lacko, and Maria Bielikova. 2018. Nl-fiit at iest-2018: Emotion recognition utilizing neural networks and multi-level preprocessing. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 217–223. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

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.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada. Association for Computational Linguistics.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.