GenSMT at SemEval-2019 Task 3: Contextual Emotion Detection in tweets using multi task generic approach

Dumitru C. Bogdan
Faculty of Mathematics and Computer Science, University of Bucharest
Human Language Technologies Research Center, University of Bucharest
bogdan27182@gmail.com

Abstract

In this paper, we describe our participation in SemEval-2019 Task 3: EmoContext - A Shared Task on Contextual Emotion Detection in Text. We propose a three layer model with a generic, multi-purpose approach that without any task specific optimizations achieve competitive results (f1 score of 0.7096) in the EmoContext task. We describe our development strategy in detail along with an exposition of our results.

1 Introduction

In recent years, emotion detection in text has become more popular due to its vast potential applications in marketing, artificial intelligence, education, politics, psychology, human-computer interaction, etc. Social platforms like Twitter and Facebook, enabled access to huge amount of textual data facilitating both theoretical and experimental research.

Consequently, sentiment analysis and emotion detection have gained the interest of researchers in natural language processing and other fields. While sentiment analysis refers to classifying a subjective text as positive, neutral, or negative; emotion detection identify specific types of feelings such as anger, joy, fear, or sadness.

SemEval is the International Workshop on Semantic Evaluation that has evolved from SensEval. The purpose of this workshop is to evaluate semantic analysis systems. Task 3 (Chatterjee et al., 2019) in this workshop, presents the task of detecting contextual emotion in a given three turn conversation where every turn is a short phrase (or tweet).

Our proposed model factors a *generic* approach to sentiment and emotion detection in tweets thus being suitable for several tasks in 2019 SemEval competition like task 3, task 5 or task 6. By not exploiting to the fullest specificity of task 3 (being a three turn conversation) and by not including transfer learning from any similar task we assumed a lower accuracy, being confident that in a longer term this approach will be useful for a large class of emotion detection tasks in text.

We leverage the latest developments in natural language processing, like *deep contextualized word representation* and a specially build submodel (encoder) as we will see in section 4.

2 Previous work

Sentiment classification, the task of detecting whether a text is positive or negative, has a long history of research. From the moment when (Pak and Paroubek, 2010) constructed a corpus of tweets for this task research in this area, on this type of corpus, increased considerably. Classifying tweets according to sentiment has many applications in political science, social sciences, market research, and many others (Martínez-Cámara et al., 2012).

It was only natural to extend research in this area to include a more fine grained classification of text, more than positive or negative. As result emotion classification in tweets started to gain interest.

Emotion detection is part of the broader area of Affective Computing that has the goal to help machines detect and simulate emotions (Picard, 1995). Psychology offers us a number of theories about how to represent emotions while two are the most important and the most often used in existing approaches in sentiment analysis: emotional categories and emotional dimensions.

Emotional categories approaches are concentrated on model emotions based on distinct emotion classes. The categorical model assumes that emotion categories clearly separable, are discrete

. Ekman presented a basic emotion model that fits categorical model approach. (Ekman, 2005) concluded that the six basic emotions are anger, disgust, fear, happiness, sadness and surprise.

Anger, sadness and happiness are the subject of the current task with one important addition: the absence of context which can add ambiguity making the task classification task much more complex, without access to facial expression and speech. Important amount of work has been done regarding to speech and facial emotion recognition however text based emotion detection systems still needs attraction of researchers (Sebea et al., 2005). Various methods where developed in the past like: *Keyword Spotting Technique*, *Lexical Affinity Method* or *Learning-based Methods*.

A similar research area is where emotions or more precisely emotion combinations plays a significant role is detection of optimism/pessimism in texts (Caragea et al., 2018).

Hopefully participants to this task will provide models and tools that will add value and will contribute to this field in a similar manner as previous SemEval tasks did (Mohammad et al., 2018).

3 Data preparation

Being a multi-task system, data preparation was also constrained to basic text preparation and tweets specific procedures. In effect we have treated all three sentences as a paragraph. We have concatenated turn1, turn2, turn3 separated by dot thus transforming the dataset in a list of pairs (label, text).

Tweets specific prepossessing involved sanitisation (links were replaced with *_url_* tag and user mentions were replaced with *_entity_* tag) and most important, we have translated emoji to text (e.g an emoji representing a face with tears of joy was expanded to ”face with tears of joy” text). Emoji conversion is very important and based on intermediary tests it can greatly influence accuracy of models on datasets containing tweets.

To run our experiments, we used the dataset provided by the task organizers (Chatterjee et al., 2019) as follows. In pre-evaluation period we have trained our models on *training-set* and evaluated our model versions on *dev-set*. Dataset provided by the organizers contained 30160 entries and we have separated them into 24999 entries for *training-set* and remaining 5161 for *dev-set*. Table 1 provides more information about categories split in each set.

Data	Angry	Sad	Happy	Others	Total
All	5506	5463	4243	14948	30160
Train	4560	4504	3544	12391	24999
Dev	946	959	699	2557	5161

Table 1: Data split by sets and categories

4 Approach

Our selected model has a classical layered approach as outlined in figure 1.

The first layer consists of few embedders, that receives the input data as a text sequence and converts it into a specific representation.

Resulted transformation is next feed to the middle layer where few encoders individually transforms input vectors to a final representation of input texts.

Next, final representations are concatenated into a large vector and transmitted to the final layer which is a dense layer those output is transformed into a probability distribution consisting of k probabilities, where k is the number of classes depending on the selected SemEval task: $k = 4$ for task 3, $k = 2$ for task 5 and task 6.

Along with layered approach, a parallel text transformation was considered, meaning that the input text was transformed into three different internal representations using three different encoders.

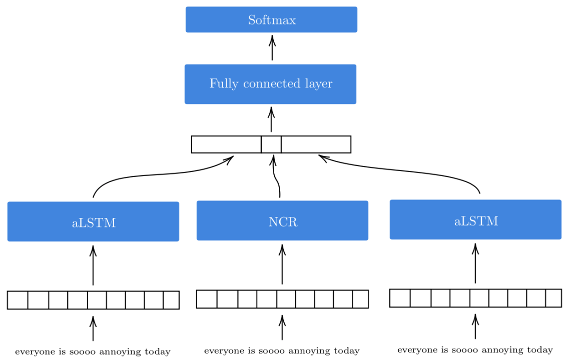
Softmax
Fully connected layer
aLSTM
NCR
aLSTM
everyone is soooo annoying today
everyone is soooo annoying today
everyone is soooo annoying today

Figure 1: GenSMT model structure

First encoder is ELMo (Peters et al., 2018), a deep contextualised word representation that models both complex characteristics of words syntax and semantics, and variations across linguistic contexts. Previous research and test showed significant improvements in various canonical NLP problems. A short description of this encoder is: ELMo word representations consists of functions of the entire input sentence computed on top of two-layer biLMs using character convolutions, as a linear function of the internal model states.

We can formalize this encoder as follows:

$$ELMo(x^{E_1}_{1:n}) = \gamma \sum_{j=0}^{l} s_j h_j^{LM}$$

where γ is a scalar parameter, l is the number of representation, s are softmax-normalized weights and h represents LSTM outputs.

Second encoder is NCR and works as follows: based on a list of words proposed by (Mohammad, 2018) it selects all occurrences of those words and their synonyms presented in the input text. Next it simply counts occurrences for each category (sad, anger, happy) and concatenates the results into a fixed size output vector.

We can formalize this encoder as follows:

$$NCR(x^{E_2}_{1:n}) = [\underset{cat \in \{anger,\dots\}}{count}(< cat >, x_{1:n})]$$

where $< cat >$ represents category and $count$ is the counting function.

Other NCR variations where proposed where we do not use a count function but we add, sub or dot product word representation and next we perform an operation of averaging, minimum or maximum.

Third encoder, RHN is an extended LSTM with recurrent dropout and possibility to use highway connections between internal layers.

We can formalize this encoder as follows:

$$RHN(x^{E_3}_{1:n}) = h \cdot t + x \cdot c$$

where h, t and c are internal compositions of nonlinear and linear functions.

Input data of the above encoders is formalized as follows:

$$\begin{aligned} x^{E_1}_{1:n} &= embedder_1(x_{1:n}) \\ x^{E_2}_{1:n} &= embedder_2(x_{1:n}) \\ x^{E_3}_{1:n} &= embedder_3(x_{1:n}) \end{aligned}$$

where $embedder_1$ produce character level representation of input text, $embedder_2$ and $embedder_3$ transform input words into their word embedding representation.

Now, we can formalize the full model as follows:

$$\begin{aligned} GenSMT(x_{1:n}) =& softmax(MLP(\Theta)); \\ \Theta =& [ELMo(x^{E_1}_{1:n}), \\ & NCR(x^{E_2}_{1:n}), \\ & RHN(x^{E_3}_{1:n})] \end{aligned}$$

As word representation we have use used various pretrained models in various configuration of GenSMT: Glove, Glove Twitter (Pennington et al., 2014), FastText (Joulin et al., 2016) and Wikipedia2vec (Yamada et al., 2018).

For development and testing we have used Python with ML related library like pytorch, numpy, pandas and others. Initial prototyping was done using Flair framework (Akbik et al., 2018) and development was done with AllenNLP (Gardner et al., 2017).

Training, validation and testing were performed on a single GPU machine. Training times were below one hour for each model configuration thus making this model suitable for hyper-parameters optimisation using grid search.

5 Experiments and results

As described in section 3 dataset was split in train and dev sets after being lightly pre-processed.

For training we use the model structure described in section 4 and we only changed model parameters and word embeddings of the embedders. We did not use any hyperparameters optimisation methods. Various model configurations were selected based on our previous experience with this kind of models.

Table 2 shows accuracy on dev tests of few selected configurations of GenSMT.

Model configuration	Accuracy
GenSMT$_1$	0.90
GenSMT$_2$	0.87
GenSMT$_3$	0.88
GenSMT$_4$	0.89

Table 2: Accuracy of various model configurations

$GenSMT_1$ configuration has the following general characteristics:

- $embeder_1$ performs character encoding
- $embeder_2$ uses 100d glove embeddings
- $embeder_2$ uses 500d wikipedia2vec embeddings
- ELMo large model
- MLP has three layers with ReLU activation
- batch size of 16
- Adam optimiser with a cosine learning rate scheduler
- 20 epochs with a patience of 5

Table 3 shows $GenSMT_1$ configuration where only default word embeddings of $embedder_3$ was changed. Results indicate that choosing word embeddings may influence accuracy.

Model configuration	Accuracy
GenSMT$_1(500d - wikipedia2vec)$	0.90
GenSMT$_1(300d - glove)$	0.89
GenSMT$_1(300d - fasttext)$	0.89

Table 3: Word-embeddings variations for a given configuration

We were interested to measure how each encoder contributes to model accuracy and for that ablation tests were performed. Table 4 shows $GenSMT_1$ configuration with one encoder removed.

Model configuration	Accuracy
GenSMT$_1(no\, ELMo)$	0.88
GenSMT$_2(no\, NCR)$	0.88
GenSMT$_3(no\, RHN)$	0.89

Table 4: Ablation tests on best performing configuration

While ELMo is powerful encoder, the gap between his absence and NCR absence, a simple encoder, is almost unnoticeable which is a surprise. This mean that a simpler model, with similar results, can be considered in scenarios where training or prediction speeds are of great importance (e.g mobile or IoT devices with low computing resources or low power resources).

For final, competition test we have used few $GenSMT_1$ configurations and the best performing one had an f1 score of 0.7096 placing this model in the upper half of the competition board.

6 Conclusions and future work

In this paper we have presented GenSMT model build as a generic system for SemEval 2019 task3, task 5 and task 6. It is a three layer model with three parallel embedders and encoders those outputs are concatenated and feed to dense layer which is next transformed into a probability distribution that produce text classification.

GenSMT was build with two constraint in mid. First we wanted to eliminate task specific dependencies as result we did not included any task specificity into the model. Second we did not use task specific transfer learning (e.g pretraining our model on a twitter sentiment dataset). This two constrains gave us the advantage to use the same model more than one task however reducing our accuracy capacity.

We have showed that choosing specific pre-trained word embeddings can slightly improve the results, however greater gains are obtained by altering model hyperparameters.

By performing ablation tests we have showed that using a powerful encoder like ELMo increase the accuracy but not with an impressive score thus giving the option to use a simple and lower computational model to attain similar results much faster for both training and prediction.

Future work should explore two hypothesis. First as an upgrade of the model, it will be interesting to study how the model will perform if we remove the initial constraints and pretrain our model with a large twitter sentiment dataset followed by task specific data manipulation (e.g remove turn1 from dataset or inverse turns order). Second, as an update of the model, we can replace current encoder and embedders with new, state of the art ones.

Another aspect, specific for micro-blogging text is emoji presence and importance. While also present in larger texts, in short micro-blogging texts, emoji have more meaning, they carry a high quantity and quality of information that can be used for emotion detection thru data preprocessing or maybe using a model/encoder especially for them.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Cornelia Caragea, Liviu P. Dinu, and Bogdan Dumitru. 2018. Exploring optimism and pessimism in twitter using deep learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 652–658. Association for Computational Linguistics.

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.

Paul Ekman. 2005. Basic emotions. In *Handbook of Cognition and Emotion*, pages 45–60. John Wiley & Sons, Ltd.

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.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Eugenio Martínez-Cámara, M. Teresa Martín-Valdivia, and L. Alfonso Ureña-López. 2012. Sentiment analysis in twitter. *Natural Language Engineering*, 20(01):1–28.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17. Association for Computational Linguistics.

Saif M. Mohammad. 2018. Word affect intensities. In *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*, Miyazaki, Japan.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA).

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 E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

R. W. Picard. 1995. Affective computing.

Nicu Sebea, Ira Cohenb, and The Netherl. 2005. Multimodal approaches for emotion recognition: A survey.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2018. Wikipedia2vec: An optimized tool for learning embeddings of words and entities from wikipedia. *arXiv preprint 1812.06280*.