CX-ST-RNM at SemEval-2019 Task 3: Fusion of Recurrent Neural Networks Based on Contextualized and Static Word Representations for Contextual Emotion Detection

Michał Perełkiewicz
National Information Processing Institute
al. Niepodległości 188b, Warsaw, Poland
michal.perelkiewicz@opi.org.pl

Abstract

In this paper, I describe a fusion model combining contextualized and static word representations for approaching the EmoContext task in the SemEval 2019 competition. The model is based on two Recurrent Neural Networks, the first one is fed with a state-of-the-art ELMo deep contextualized word representation and the second one is fed with a static Word2Vec embedding augmented with 10-dimensional affective word feature vector. The proposed model is compared with two baseline models based on a static word representation and a contextualized word representation, separately. My approach achieved officially 0.7278 microaveraged F1 score on the test dataset, ranking 47th out of 165 participants.

1 Introduction

The EmoContext task in the Semantic Evaluation 2019 (SemEval 2019) competition focuses on the classification of textual dialogues i.e. a user short conversation with a bot, into 'happy', 'sad', 'angry' and 'others' sentiment classes. Understanding emotions in textual conversations is a challenging task mainly because of an absence of others expression channels such as voice modulations and facial expressions usually accompanying a people conversation. In textual conversation determining the emotion of a given statement is very dependent on the context of previous statements.

A sentiment detection field has been thoroughly analysed. The first attempts to manage this problem included mainly extracting hand-crafted features and knowledge-based systems (Balahur et al., 2011; Chaumartin, 2007; Joulin et al., 2016).

Neural machine learning approaches, especially Recurrent Neural Networks (RNN) like LSTM and GRU and Convolutional Neural Networks (CNN), are effective tools for detecting sentiment from text and were widely used in this area (Tang et al., 2015; Liu et al., 2016; dos Santos and Gatti, 2014). Except that, one of the approaches employs Hierarchical Attention Networks (HAN) to determining emotions from textual dialogues data (Saxena et al., 2018).

In this work, I propose a deep neural fusion model that combines two Bidirectional LSTM Recurrent Neural Networks for detecting sentiments in textual dialogues. I use static Word2Vev word representation and a contextualized ELMo word representation to create a unified model. This architecture of the model is inspired by the work presented in (Gupta et al., 2017).

The rest of the paper is structured as follows. Section 2 describes the proposed approach and word representations. The experiments and results are presented and discussed in Section 3. Finally, in the last section, the conclusions are presented.

2 Approach

The following section provides details on preprocessing I used to normalize textual data provided by the task organizers, the method to manage unbalanced datasets problem and describes the model architecture and used word embeddings.

2.1 Preprocessing

To clean and normalize textual data, I adapt the *ekphrasis* text processing library[1] with some changes. It was designed with a focus on text from social networks, such as Twitter or Facebook. It provides tools to process text, such as tokenization, word normalization, word segmentation and spell correction, using word statistics from 2 big corpora (English-language Wikipedia and Twitter)(Baziotis et al., 2017).

The *ekphrasis* preprocessing techniques I used includes: Twitter-specific tokenization, omitting

[1] https://github.com/cbaziotis/ekphrasis

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 180–184
Minneapolis, Minnesota, USA, June 6–7, 2019.

special words and phrases (like emails, phone numbers, nicknames, dates and time, URLs), spell correction, words annotations (for uppercased, repeated, hashtagged and elongated words), reducing emoticons variations by replacing emoticons expressing similar emotions to the same form (e.g ':)' and ':-)' emoticons are both mapped to the ⟨happy⟩ mark).

To better normalize texts and suit the tool to EmoContext datasets, I did following modifications in the text processing library:

- adding a new dictionary to expand English contractions (e.g. can't → can not, couldn't've → could not have), normalize slang words (e.g. plz → please) and correct typos (whhat → what).
- extending the emoticons dictionary with emoticons found in the training and the testing corpora to reduce them to the basic form.
- adding new map for emoticons expressing strong emotions by adding the *very* word before them (e.g ':)))))))' → very ⟨happy⟩)
- changing emoticons mapping by adding the prefix 'emo_' to emotion marks[2], like ':)': '⟨emo_happy⟩'.

2.2 Altering the Training Balance

According to information pointed out by the task organizers on the EmoContext web page[3], provided datasets are unbalanced. Training data consists of about 5000 (about 17%) samples each from 'angry', 'sad', 'happy' class, and about 15000 (about 50%) samples from 'others' class, whereas, both the development dataset and the test dataset sets have a real-life distribution, which is about 4% each of 'angry', 'sad', 'happy' class and the rest is 'others' class.

To deal with unbalanced datasets and avoid to bias model towards the 'other' class, I created two derivative training datasets: the binary one, by mapping 'happy', 'sad', 'angry' labels to the one 'sentiment' label and left the 'others' label unchanged. This binary set contains about 50% of dialogues covey some sentiment and about 50% dialogues conveys no sentiment (the original 'other' label). The second derivative dataset contains dialogues labelled one of the following labels: 'happy', 'sad', 'angry'. So, this dataset contains examples originally conveys some sentiment. Classes distributions of these two derivative datasets are more balanced. These derivative datasets are used to learn a two-stage model based on two Recurrent Neural Networks as described in Subsection 2.4.

Furthermore, I extend the training dataset by carrying 1753 examples from the development dataset and left 1000 examples to valid my model.

2.3 Word Embeddings

Word embeddings are representations of words as n-dimensional vectors, previously learned on large text corpus. The proposed model is fed both a static word embedding and a contextualized (dynamic) word embedding.

Static Word Embedding Static word embeddings map the same word to the same vector, independently of the word context. The advantages of static word embeddings are easy interpretability and capturing semantic properties of words (embedding vectors for words semantically similar are similar as well). However, they suffer from some problems, for example a meaning conflation deficiency – the inability to discriminate among different meanings of a word.

To embed textual data to a static representation I adapt pretrained 300-dimensional Word2Vec word embedding vector augmented with a 10-dimensional vector of word affective features proposed in (Baziotis et al., 2018). It was trained on the collection of 550 million Twitter messages preprocessed by the *ekphrasis* text processing library. Such very similar preprocessing stage can better suit EmoContext textual data to this pretrained embedding.

Therefore, the static word embedding I use maps every sentence in EmoContext datasets to a 310 dimensional n-length list where n is a number of words in a sentence.

Contextualized Word Embedding As opposed to static embeddings, contextualized word embeddings generate words representation vectors dynamically, depending on the context in which a given word appears. They need the whole sentence to generate words embeddings because of a need to know the context of each word in a sentence.

[2] only used by the contextualized embedding

[3] https://competitions.codalab.org/competitions/19790#learn_the_details-data-set-format

I employee the Embeddings from Language Models (ELMo) word embedding, where word vector representations are learned functions of the internal states of a deep bidirectional language model (biLM) (Peters et al., 2018). I use official available, pretrained ELMo original model[4] which was learned on the dataset of 5.5 billion tokens consisting of Wikipedia (1.9 billion) and all of the monolingual news crawl data from WMT 2008-2012 (3.6 billion). To vectorize words, I get the state of the last biLM layer built on 1024 neurons, therefore the embedding vectors are the length of 1024 elements. To better suit this embedding to the EmoContext datasets, I fine-tuned the ELMo model by learning pretrained biLM. For this purpose, I used all utterances from the datasets provided by the organizers.

To generate more context-aware representations of words, for each dialogue in the datasets I merged the previous one or two utterances with the second or the third utterance in a dialogue, respectively. Such a context extending allows generating vector representations for two last utterances taking into account the context of the previous utterances in whole dialogue. For a first utterance, I use only the first utterance without any extension.

2.4 Model Architecture

Next, I present in detail the submitted model. My final model is based on the fusion of two deep, two-layer Bidirectional LSTM Neural Networks with Attention Mechanism. First one consumes 310-dimensional vectors and produces a 250-dimensional encoding as a averaged Bidirectional LSTM network states over time. The second one consumes 1024-dimensional vectors and produces a 300-dimensional encoding vector, as in the previous case, as a averaged Bidirectional LSTM network states over time. Then these two encodings are merged and are consumed by a Fully Connected layer with 120 neurons. The activation function of this layer is a softmax function to get probabilities of output classes. The architecture is depicted in Figure 1.

To avoid overfitting during learning the model, I use the following regularization techniques:

– Dropout inputs to BiLSMT networks, for each layer.

– Dropout input to Fully Connected Layer.

[4] https://allennlp.org/elmo

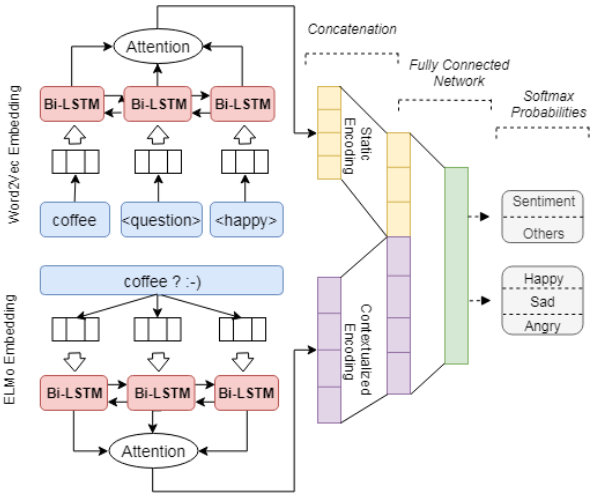

Figure 1: The proposed model architecture.

– Recurrent Dropout to dropout connections between the recurrent units in Recurrent Networks for each layer.

I learnt two models based on described architecture separately using datasets describing in Section 2.2. These two networks are stacked and the prediction process runs as follows:

– the general model (called sent-others network) predicts if a dialogue conveys some sentiment or not. This model predicts one of two labels: 'sentiment' or 'others' to input data.

– if the general model has predicted the 'sentiment' label, the input data is carried forward to the second model (called happy-sad-angry network). The responsibility of this model is to determine the sentiment of an input data and put one of three labels: 'angry', 'sad', 'happy'.

After this two-stage classification, the model labels input data as 'happy', 'sad', 'angry', 'others' example.

Furthermore, because of different class distributions in the datasets, I added an parameter T to the sent-others network to fit the model to development/test datasets classes distribution. This parameter specifies the minimum value that the softmax probability for 'sentiment' class has to achieve to label input data as 'sentiment'. For predicting on the test dataset, I set the parameter T to 0.75, which was the value that achieved the best result on the validation dataset.

Parameter	Value		Tested Values
	sent-others	happy-sad-angry	
LAYERS_NUMBER	2	2	2
NEURONS_NUMBER	300, 300	250, 250	200, 200; 250, 250; 300, 300
BI-LSTM_INPUT_DROPOUT	0.3, 0.5	0.3, 0.5	0.2, 0.5; 0.3, 0.5
RECURRENT_DROPOUT	0.3, 0.5	0.3, 0.5	0.2, 0.5; 0.3, 0.5
FULL_CONNECTED_LAYER_DROPOUT	0.3	0.3	0.3, 0.4, 0.5
FULL_CONNECTED_LAYER_SIZE	120	120	100, 120
BATCH_SIZE	32	32	32
THRESHOLD_T	0.75	-	0.25, 0.55, 0.75

Table 1: The parameters of the proposed model.

Model	Validation(1000)			Test		
	sent-others	happy-sad-angry	2-stage	sent-others	happy-sad-angry	2-stage
ELMo	0.9456	**0.9700**	0.7511	0.9517	**0.9447**	0.7192
Word2Vec-310	0.9534	0.9641	0.7500	**0.9568**	0.9351	0.7180
Word2Vec-310 + ELMo	**0.9542**	0.9672	**0.7558**	0.9537	0.9422	**0.7278**

Table 2: Microaveraged F1 score of baselines models and the fusion model on the validation and the test datasets.

Table 1 presents tested model parameters and the best parameters set for the validation dataset.

3 Results

Table 2 presents microaveraged F1 score achieved on the test and the validation datasets for the proposed model containing also F1 scores for the first (sent-onthers network) and the second (happy-sad-angry network) classification stages. I compared the proposed fusion model against 2 baselines based on the static and contextualized models used in the proposed method, separately. Therefore the first baseline model uses 2-layer Bi-LSTM Neural Network which is learned on static, *Word2Vec* with affective features word representation, and the second one is a baseline model 2-layer BI-LSTM Neural Network as well but is learned on contextualized ELMo word representation embedding. The results of such baseline models allow better insight into the performance of the proposed fusion model.

The best results for 2-stage classification for the test dataset achieved the fusion model (0.7558 for validation and 0.7278 for the test dataset) despite the worse results in the sent-others and the happy-sad-angry classification stages. For the validation dataset, the proposed fusion model achieved the best score as well. The best result for 2-stage classification was better by about 1 percent from baselines results.

4 Conclusion

In this paper, we have presented the fusion model, a sentiment classifier that combines the features of static and contextualized word embedding. This approach achieved officially 0.7278 F1-score, ranking 47$^{\text{th}}$ out of 165 participants.

My results show that combining word embeddings can improve sentiment detection models based only on one, static or contextualized, embedding. The two-stage classification model can better insight to classification process and parameterized each stage separately.

References

Alexandra Balahur, Jesús M. Hermida, and Andrés Montoyo. 2011. Detecting implicit expressions of sentiment in text based on commonsense knowledge. In *Proceedings of the 2Nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA '11, pages 53–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christos Baziotis, Nikos Athanasiou, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. NTUA-SLP at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning. *CoRR*, abs/1804.06658.

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, 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.

François-Régis Chaumartin. 2007. Upar7: A knowledge-based system for headline sentiment tagging. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 422–425, Prague, Czech Republic. Association for Computational Linguistics.

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.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2873–2879. AAAI Press.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Rohit Saxena, Savita Bhat, and Niranjan Pedanekar. 2018. Emotionx-area66: Predicting emotions in dialogues using hierarchical attention network with sequence labeling. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 50–55, Melbourne, Australia. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal. Association for Computational Linguistics.