MILAB at SemEval-2019 Task 3: Multi-View Turn-by-Turn Model for Context-Aware Sentiment Analysis

Yoonhyung Lee, Yanghoon Kim, and Kyomin Jung
Seoul National University, Seoul, Korea
{cpi1234, ad26kr, kjung}@snu.ac.kr

Abstract

This paper describes our system for SemEval-2019 Task 3: EmoContext, which aims to predict the emotion of the third utterance considering two preceding utterances in a dialogue. To address this challenge of predicting the emotion considering its context, we propose a Multi-View Turn-by-Turn (MVTT) model. Firstly, MVTT model generates vectors from each utterance using two encoders: word-level Bi-GRU encoder (WLE) and character-level CNN encoder (CLE). Then, MVTT grasps contextual information by combining the vectors and predict the emotion with the contextual information. We conduct experiments on the effect of vector encoding and vector combination. Our final MVTT model achieved 0.7634 microaveraged F1 score.

1 Introduction

Sentiment analysis is a task of identifying emotional information from text materials and has been studied by various research fields since it can be applied to many applications such as public survey and market analysis. However, most studies in sentiment analysis have only focused on a single sentence (Socher et al., 2011) or a single document (Pang and Lee, 2005). It is still hard to predict the emotion of a sentence with extra contextual information because the emotion can be understood differently depending on its context. SemEval-2019 Task 3: EmoContext (Chatterjee et al., 2019) provides a dataset of dialogues which consist of three utterances between two users (Figure 1). Participants are required to predict the emotion of the third utterance among 'Happy', 'Sad', 'Angry', and 'Others', considering its context of two preceding utterances.

In this paper, we propose a Multi-View Turn-by-Turn (MVTT) model which encodes each utterance separately and combines the encoded vectors to get the contextual information. MVTT model first generates vectors from each utterance with two encoders: word-level Bi-GRU encoder (WLE) and character-level CNN encoder (CLE). The two-encoder strategy makes MVTT model more robust to the noisy texts which have a lot of typos and abbreviations. Then, MVTT extracts contextual information by combining the vectors and makes a prediction. We compare the MVTT model with some variants focusing on utterance vector encoding and utterance vector combination methods with microaveraged F1 score (F1) which is the main evaluation metric.

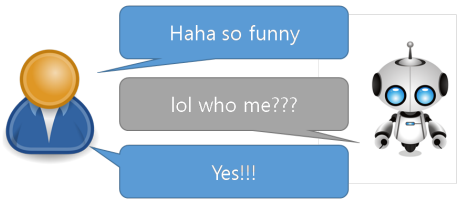

Figure 1: An example of a dialogue between two users.

This paper is organized as follows: Section 2 describes MVTT model architecture in detail. Section 3 describes dataset and various methods that we use to reflect the dataset's characteristics to our training. Section 4 compares our results of MVTT model and other variants, and Section 5 outlines our conclusions.

2 System Description

This section describes our Multi-View Turn-by-Turn (MVTT) model which consists of two encoders: word-level Bi-GRU encoder (WLE) and character-level CNN encoder (CLE), which make our model more robust to noisy data. First, MVTT generates utterance vectors from each utterance using the encoders. Then, to understand the contextual information, MVTT combines the vectors into context-aware dialogue vectors and makes a

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 256–260
Minneapolis, Minnesota, USA, June 6–7, 2019.

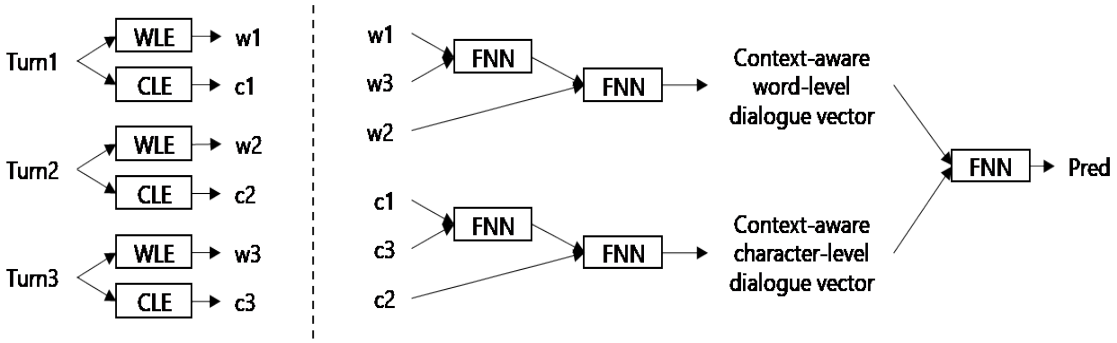

Figure 2: Overview of MVTT model: how each encoder generates utterance vectors from each utterance and how MVTT model combines the vectors to make a prediction.

prediction using the context-aware dialogue vectors.

In this section, we first describe how each encoder generates the utterance vectors from each utterance and how MVTT combines the vectors into the context-aware dialogue vectors in detail.

2.1 Word-level Bi-GRU encoder

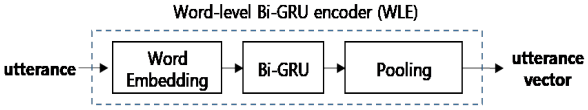

Figure 3: Word-level Bi-GRU encoder.

Word-level Bi-GRU encoder (WLE) takes an utterance as a sequence of word tokens $\{w_1, w_2, ..., w_N\}$. The tokens are fed into an embedding layer which is initialized with Glove word-embedding (Pennington et al., 2014) trained with a large twitter corpus. Then the embedded tokens are fed into the Bi-GRU encoder to get an utterance vector by max-pooling its hidden states over time.

WLE allows the model to benefit from pre-trained Glove word-embedding which have abundant information in syntactic and semantic word relationships. Also, MVTT benefits a lot from Bi-GRU encoder (Cho et al., 2014): (a) MVTT model can understand an text contextually with previous word information using Bi-GRU's gating mechanism; (b) by reading an text in two opposite ways, MVTT model can extract contextually more informative information. This is especially beneficial for MVTT to understand contextual meaning of a dialogue from each utterance.

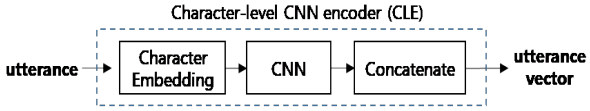

Figure 4: Character-level CNN encoder.

2.2 Character-level CNN encoder

Character-level CNN encoder (CLE) takes an input as a sequence of character tokens $\{c_1, c_2, ..., c_N\}$. The tokens are fed into a randomly initialized embedding layer, and then the embedded tokens are fed into the CNN encoder (Kim, 2014) to get an utterance vector.

Since it is impossible to consider all words in a word-level encoding, vocabulary consisting of 15k words is pre-defined based on word frequency and the other words are considered as out of vocabulary (OOV) tokens. Therefore, if the model only depends on WLE to extract features from a sequence of word tokens, a significant proportion of words will be tokenized as the OOV tokens when the texts are noisy. As a result, the model can't sufficiently utilize the benefit from pre-trained word-embedding. However, since the CNN encoder helps extract the local features from a sequence of character tokens, CLE enables our model to speculate the meaning of the text has some typos.

2.3 Multi-View Turn-by-Turn encoding

Multi-View Turn-by-Turn encoding is a method of generating utterance vectors from each utterance first and then combining the vectors into context-aware dialogue vectors. MVTT makes a prediction by encoding each utterance and combining them using concatenation and feed-forward neural network (FNN) as follows:

$$w13 = FNN([w1; w3])$$

$$w123 = FNN([w13; w2])$$

$$c13 = FNN([c1; c3])$$

$$c123 = FNN([c13; c2])$$

$$pred = FNN([w123; c123])$$

where $w1, w2, w3$ are the utterance vectors from each utterance generated by WLE and $c1, c2, c3$ are the utterance vectors from each utterance generated by CLE.

Since the first and third utterances are written by the same user, they are more informative in predicting the emotion of the third utterance. Therefore, by processing each utterance separately and combining them as above, the model can understand the context while maintaining the important emotional information.

2.4 Binary relevance classification

Binary relevance classification is a classification scheme to independently train binary classifiers for each label. It has usually been used for multi-label classification tasks and we apply the method to our task. In our system, we build three identical classifiers for 'Happy', 'Sad', 'Angry' classes and independently train them to output probabilities of each class. Then, we take the emotion with the highest probability as a class prediction, otherwise, if all the probabilities don't exceed 50%, take 'Others' as a predicted class.

3 Experiments

SemEval-2019 Task 3: EmoContext provided dialogue dataset consisting of three utterances written by two users and each sample is labeled among 'Happy', 'Sad', 'Angry' and 'Others'. In this section, we describe the dataset and some implementation details.

3.1 Datasets

The provided dialogue dataset is split into training, validation and test sets. Table 1 shows the label distribution of each data split. As Table 1 indicates, there are large differences in class label distributions among data splits and it is important to consider the differences in configuring our system.

Data split	Happy	Sad	Angry	Others
training	4243	5463	5506	14948
validation	142	125	150	2338
test	284	250	290	4677

Table 1: The statistics for the number of labels of each split.

3.2 Implementation details

We optimize our model using Adam optimizer (Kingma and Ba, 2014) and learning rate is set to 0.0015. We use Bi-GRU with 256 hidden units and CNN filters with window sizes of [3, 5, 9], 64 feature maps each. All FNN have 256 hidden units with $tanh$ activation function except for the last FNN classifier with $sigmoid$ function.

3.3 Pre-processing

In this task, we pre-process the utterances as described in Figure 5. We first lowercase all texts and replace abbreviations with their original forms as many as possible to make the best use of Glove word-embedding. Next, we unify emojis that have similar meanings into one specific emoji to help our model to learn emoji embeddings.

Figure 5: Text pre-processing.

3.4 Label smoothing

Label smoothing is a method to relax our confidence on the labels by using lower target values like 0.7 instead of 1. In the test set, almost all samples belong to the "Others" class with only a small percentage of examples belonging to the "Happy", "Sad", or "Angry" classes. Therefore, if we train each classifier for each emotion with label smoothing, we can prevent the model from predicting a emotion with excessive confidence and make the model be more likely to predict the emotions as 'Others'.

4 Results

In this section, we compare the performance of our MVTT model with some variants of our model. Since our model mainly consists of WLE and CLE, we try to investigate how our model benefits

from both encoders. Further, we found that different combination methods of utterance vectors make a great difference in model evaluation. All of the results below are experimental results on the test set. MVTT outperforms all other variants and achieved 0.7634 microaveraged f1 score.

4.1 Ablation test on sentence embeddings

Our MVTT model utilizes the features of WLE and CLE. As is shown in Table 2, the model takes more advantage from WLE than CLE since the WLE utilizes the pre-trained word-embedding vectors trained on large twitter corpus which have abundant information in syntactic and semantic word relationships in the corpus. However, when we use both encoders, our MVTT model outperforms both models that use only one encoder. This results from the fact that CLE gives robustness to our model because it takes an input as a sequence of character tokens and extracts the local features from it.

Model	F1(H)	F1(S)	F1(A)	F1μ
WLE	0.7227	0.7680	0.7638	0.7515
CLE	0.6975	**0.7860**	0.7089	0.7270
MVTT	**0.7273**	0.7853	**0.7767**	**0.7634**

Table 2: Performance comparison among WLE, CLE and MVTT.

4.2 Impact of Turn-by-Turn encoding

Considering the characteristic of the given task, we find that the way to combine features from each utterance of a dialog is crucial. We tried several different combination methods, especially in the order of combination, to find out which setting has the most explainable structure with the best performance. We here list some variants with comparably better performance:

- C123: We simply concatenate $w1\,(c1)$, $w2\,(c2)$, $w3\,(c3)$ and feed it into a FNN to generate context-aware dialogue vectors.
- C12_3: Firstly concatenate $w1\,(c1)$, $w2\,(c2)$ and feed it into a FNN, and then concatenate the output and $w3\,(c3)$ and feed it into another FNN to generate context-aware dialogue vectors.
- C13_2 (MVTT): Firstly concatenate $w1\,(c1)$, $w3\,(c3)$ and feed it into a FNN, and then concatenate the output and $w2\,(c2)$ and feed it into another FNN to generate context-aware dialogue vectors.
- Submission: Ensemble of C123, C12_3, C13_2 models with various hyper parameters

Model	F1(H)	F1(S)	F1(A)	F1μ
C123	0.7119	0.7798	0.7602	0.7502
C12_3	0.7029	0.7674	0.7524	0.7406
Submission	0.7236	**0.7860**	0.7656	0.7581
MVTT	**0.7273**	0.7853	**0.7767**	**0.7634**

Table 3: The effect of vector combination on performance.

Table 3 shows the results of MVTT and some variants which combine the utterance vectors in other ways. As is shown in Table 3, our utterance vector combination method enables our model to understand both the emotional and contextual information. Since it is likely that a person's emotion is maintained through a 3-turn dialogue, combining the utterance vectors by user first and then making a prediction is beneficial to understand the context while maintaining emotional information.

5 Conclusion

In this paper, we propose a Multi-View Turn-by-Turn model (MVTT) for SemEval-2019 Task 3: EmoContext. Our goal was to predict the emotion of the third utterance in a dialogue consisting of three utterances. Firstly, MVTT model generates utterance vectors from each utterance using two encoders: word-level Bi-GRU encoder and character-level CNN encoder. The encoders make MVTT model more robust to the noisy texts. Then, MVTT combines the vectors to understand both the emotional and contextual meanings. We evaluated our MVTT model and its variants, focusing on utterance vector encoding and utterance vector combination. Our final MVTT model achieved 0.7634 microaveraged f1 score.

References

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.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger

Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. 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.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.