OleNet at SemEval-2019 Task 9: BERT based Multi-Perspective Models for Suggestion Mining

Jiaxiang Liu, Shuohuan Wang, and Yu Sun

Baidu Inc., Beijing, China,
{liujiaxiang, wangshuohuan, sunyu02}@baidu.com

Abstract

This paper describes our system participated in Task 9 of SemEval-2019: the task is focused on suggestion mining and it aims to classify given sentences into suggestion and non-suggestion classes in domain specific and cross domain training setting respectively. We propose a multi-perspective architecture for learning representations by using different classical models including Convolutional Neural Networks (CNN), Gated Recurrent Units (GRU), Feed Forward Attention (FFA), etc. To leverage the semantics distributed in large amount of unsupervised data, we also have adopted the pre-trained Bidirectional Encoder Representations from Transformers (BERT) model as an encoder to produce sentence and word representations. The proposed architecture is applied for both sub-tasks, and achieved f1-score of 0.7812 for subtask A, and 0.8579 for subtask B. We won the first and second place for the two tasks respectively in the final competition.

1 Introduction

Suggestion mining, which can be defined as the extraction of suggestions from unstructured text, where the term *suggestions* refers to the expressions of tips, advice, recommendations etc. (Negi et al., 2018). For example, *I would recommend doing the upgrade to be sure you have the best chance at trouble free operation.* and *Be sure to specify a room at the back of the hotel.* should be a suggestion for electronics and hotel separately. Collecting suggestions is an integral step of any decision making process. A suggestion mining system could extract exact suggestion sentences from a retrieved document, which would enable the user to collect suggestions from a much larger number of pages than they could manually read over a short span of time.

Suggestion mining remains a relatively young area. So far, it has usually been defined as a problem of classifying sentences of a given text into suggestion and non-suggestion classes. Mostly rule-based systems have so far been developed, and very few statistical classifiers have been proposed (Negi and Buitelaar, 2017) (Negi et al., 2016) (Negi and Buitelaar, 2015) (Brun and Hagège, 2013). A related field to suggestion mining is sentiment classification which given a sentence or a document, it should infer the sentiment polarity e.g. positive, negative, neutral. So, many classical sentiment classification systems can be used in suggestion mining like the widely used CNN-based models (Kim, 2014) or RNN-based models (Kawakami, 2008). However, there are still many challenges in this suggestion mining task. First of all, both of the subtasks suffers from severely lack of data. Second, one of the subtasks requires the model should have transferability without seeing any of the target domain data. To tackle those problems, knowledge transfer or transfer learning between domains would be desirable. In recent years, transfer learning techniques have been widely applied to solve domain adaptation problem, e.g. (Ganin et al., 2016). And in our system, considering the simplicity for training a model, we turn to taking use the power of large amount of unsupervised data for knowledge representations for both same domain and cross domain tasks.

Recently researches have shown that pre-training unsupervised language model can be very effective for learning universal language representations by leveraging large amounts of unlabeled data, e.g. the pre-trained Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). It has shown that BERT can be fine-tuned to create state-of-the-art models for a range of NLU tasks, such as question answering

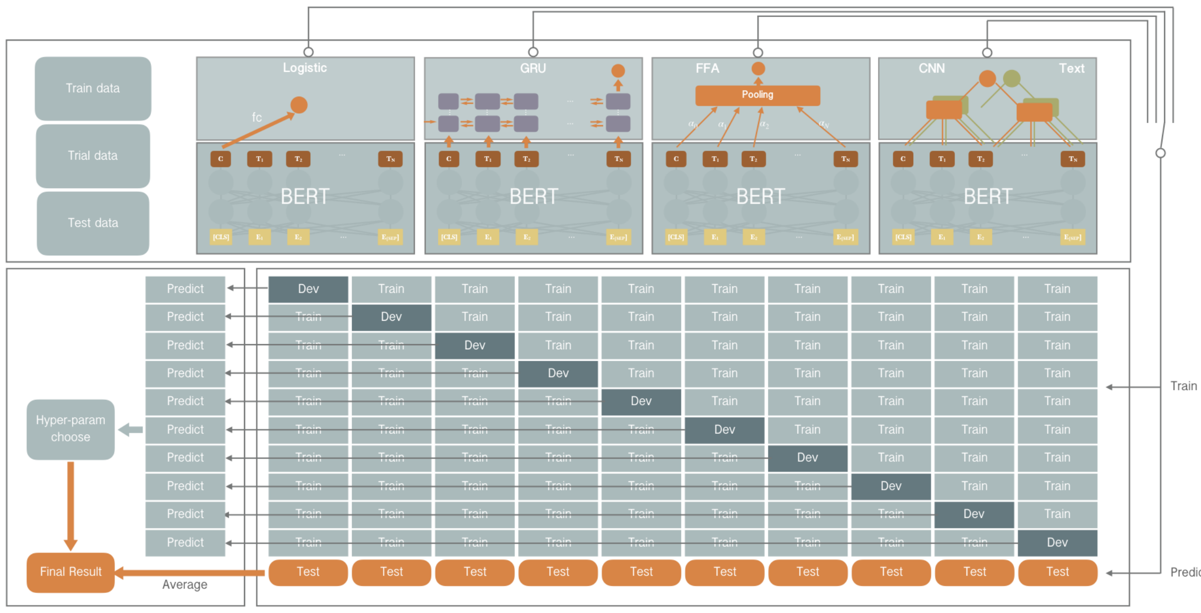


Figure 1: An overall framework and pipeline of our system for suggestion mining

and natural language inference. To further make use of the model in our task, various of different task specified layers are devised. The experiment on test datasets shows that with the devised task specified layers, a higher f1 scores can be got in both tasks, and moreover, benefiting from the large amount of unlabeled data, it is very easy to train cross domain models.

The paper is organized as follows: Section 2 describes the key models proposed for the SemEval 2019 Task 9 (Negi et al., 2019). Section 3 shows the experiment details including dataset preprocessing method, experiment configurations, threshold selection strategy and the alternatives we explored with respect to sublayers and their combination, and performances of different models. Finally, we conclude our analysis of the challenge, as well as some additional discussions of the future directions in Section 4.

2 System for Suggestion Mining

2.1 Multi-Perspective Architecture

As shown in Figure 1. our model architecture is constituted of two modules which includes a universal encoding module as either a sentence or a word encoder, and a task specified module used for suggestion classification. To fully explored the information generated by the encoder, we stack a serious of different task specified modules upon the encoder according to different perspective. Intuitively, we could use the sentence encoding directly to make a classification, to go further beyond that, as language is time-series information in essence, the time perspective based GRU cells can also be applied to model the sequence state to learn the structure for the suggestion mining task. Similarly, the spatial perspective based CNN can be used to mimic the n-gram model, as well. Moreover, we also introduce a convenient attention mechanism FFA (Raffel and Ellis, 2015) to automatically learns the combination of most important features. At last, we ensemble those models by a voting strategy as final prediction by this system. The different task specified modules will be described below.

2.2 Sentence Perspective Encoding

In the sentence encoder module, a special mark $[CLS]$ is added to the front of each sentence to help the encoder to encode all the input sentence. As a result, the output corresponds the first token is regarded as the sentence representation,

$$\begin{cases} c = E(w_t), & t = 0 \\ e_t = E(w_t), & t \in [1, T] \end{cases} \tag{1}$$

where E is the encoder module, which we user BERT in practice, c, e_t is sentence and word representation respectively, T is the total length of input sequences. We fed c into a logistic network to classify the suggestions.

2.3 Time Perspective Encoding

The Gated Recurrent Unit (GRU)(Cho et al., 2014) is famous for processing sequence data, e.g. sentences, with less parameters. In our task, we feed the e_t into GRU cells to get word representation from a time series perspective h_t,

$$\begin{aligned} z_t &= \sigma\big(e_t U^z + h_{t-1} W^z\big) \\ r_t &= \sigma\big(e_t U^r + h_{t-1} W^r\big) \\ \tilde{h}_t &= \tanh\big(e_t U^h + (r_t * h_{t-1}) W^h\big) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \\ u &= \frac{1}{T} \sum (h_t) \\ v &= \max_{1 \le t \le T} (h_t) \\ c &= [u; v] \end{aligned} \tag{2}$$

where h_t is hidden state of GRU of time step t, u is a mean pooling vector of h_t, and v is a max pooling vector of h_t. In practice, not only the h_t is used to feed into the classification layer, but the concatenated vector c is also used to train a binary classification logistic layer.

2.4 Spatial Perspective Encoding

To model the spatial connections of adjacent words, we use Convolutional Neural Network (CNN) (Kim, 2014), which is easy to implement and very fast for train. In our system, two CNN layers are stacked upon BERT model and the batch normalization (Ioffe and Szegedy, 2015) is applied in each layer. Also, the ReLu (Nair and Hinton, 2010) function is chosen as activation function. And we use max pooling to fuse the output of convolutional layers.

2.5 Attention Perspective Encoding

A recently proposed method for easier modeling of long-term dependencies is *attention* (Bahdanau et al., 2014). Attention mechanism allows for a more direct dependence between the state of the model at different points in time. Intuitively, the model with less parameters is easier to train based on small dataset, therefore, we try to use a more straight and simplified attention model, *Feed Forward Attention* (FFA) (Raffel and Ellis, 2015), which would allow it to be used to produce a single vector v from an entire sequence, the process could be formulated as follows:

$$\begin{aligned} s_t &= f(e_t) \\ \alpha_t &= \frac{exp(s_t)}{\sum_{k=1}^{T} exp(s_k)} \\ l &= \sum_{t=1}^{T} \alpha_t e_t \end{aligned} \tag{3}$$

where, f is a function mapping e_t to a unnormalized scaler s_t indicating the importance of word w_t. The l is used to make a classification to decide which input sentence is a suggestion.

But what should be noticed here is that, subtask B, whose trial data and test data are all from hotel review domain and no training data from same domain as test data is provided, is substantially a transfer learning problem. It can only learn from windows forum corpus provided in subtask A. Therefore, squeeze more cross-domain features and drop the noise is critical for subtask B. So we also introduce the hard attention mechanism (Shankar et al., 2018) :

$$l' = \sum_{\alpha \in TopK(\vec{\alpha})} \alpha h_t \tag{4}$$

we select top k important words by the attention weights α. At last the vector l and l' are used to train a binary classification logistic layer for the subtasks.

2.6 Ensemble

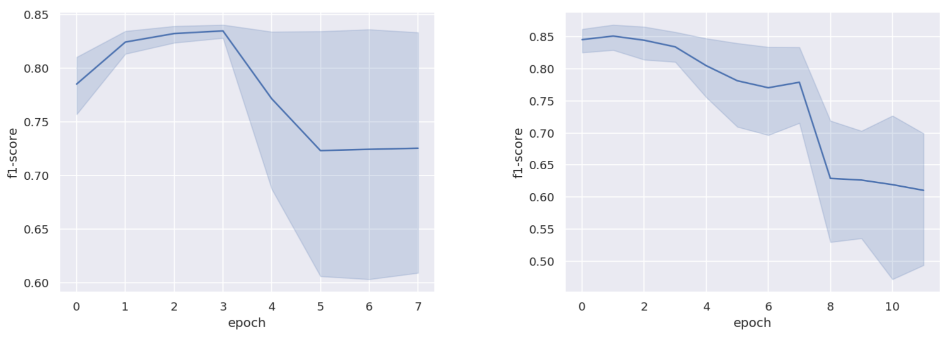
As shown in Figure 1., cross validation was adopted to ensure robustness for each model to the task 9 of SemEval-2019. In subtask A, after the 10 folds cross validation in training set has finished, the result for each fold is concatenated and used to select best classification threshold to decide a test sample from label 0 to 1. The model trained in each fold is also used to predict on test data, so the 10 test predictions is fused by mean pooling as a final prediction. Finally the simple voting method is used to fuse different model's result. In subtask B, we use the trial data as dev set to select best hyper parameters, so no cross validation is used.

3 Experiment

3.1 Dataset

The statistics of datasets provided by SemEval 2019 Task 9 are show in Table 1.

In both subtasks, no extra data are used for training models. As shown in Table 1, there are 8500



(a) Subtask A (b) Subtask B

Figure 2: Mean f1 score for trial set of different models for every epoch. In subtask A, f1 score incrementally increase and fall after the 3rd epoch, while in subtask B, f1 score of initial epoch is always surprisingly high and decreases thereafter.

Subtask A	Suggestion(%)	Non-Suggestion (%)
train	2085 (0.24)	6415 (0.75)
trial	296 (0.50)	296 (0.50)
test	87 (0.10)	746 (0.89)
Subtask B	Suggestion(%)	Non-Suggestion(%)
trial	404 (0.49)	405 (0.50)
test	348 (0.42)	476 (0.57)

Table 1: Dataset statistics for subtask A and subtask B

train examples, 592 labeled trial examples and 833 unlabeled test examples in subtask A. Different from subtask A, there are only 808 labeled trial examples, and 824 unlabeled test examples in subtask B, no training data from same domain as test data is provided. So, we use all labeled data in subtask A as the training data to do a transfer learning task to help learn subtask B. In both subtasks, the trial sets are used to help select the best model.

3.2 Details

Data Preprocessing: We use the same data cleaning method as (Cho et al., 2014), which removed the special marks. The sample is forced to *unk* if the cleaned sentence is empty.

Data augmentation method was also used in subtask A. During the error analysis procedure, we found that the model has strong tendency to learn specific terms for the task, which means the model is overfitting training data. To tackle this problem, not only dropout method is used, but also we introduce a auxiliary model to identify the importance terms according feature scores, e.g. the feature weights in a linear model. In our experiment, we use linear-kernel SVM as the auxiliary model. Specifically, we first run a linear-kernel SVM on training set. To get best performance of SVM, grid search is used to choose best hyper-parameters. When finished training SVM model, the coefficients of features in the model is collected. Then, according to the value of coefficient, the most J important word are selected as key features. Finally, we replicate training samples with random dropping those important words with drop rate α to force the model to not only rely on specific terms, but also learn sentence structure of this task. In our experiment, we take J as 100, and α 0.5.

In subtask B, besides cleaning data, we combine subtask A train set and subtask A trial set to form a bigger training set. But the drop important word strategy is not applied.

Threshold choosing: As suggestion mining is introduced as a binary classification problem, choosing appropriate threshold for the logit is vital to the performance. In subtask A, a 10-fold cross validation is executed and we obtain the best threshold by calculating f1-score between the concatenated 10 validation results and training set.

In subtask B, all the data of subtask A is used as training data, and the threshold is chosen by using the subtask B trial dataset.

Empirically, the representations from BERT is universal, so after task specified fine-tuning, the performance will increases as it is show in Figure 2a. But, for the subtask B, training dataset of subtask A have a different distribution from data of subtask B. However, we assume that they still share some underlying semantics. Therefore by

Models	CV f1-score	test score
BERT-Large-Logistic	0.8522 (±0.0213)	0.7697
BERT-Large-Conv	0.8520 (±0.0231)	0.7800
BERT-Large-FFA	0.8516 (±0.0307)	0.7722
BERT-Large-GRU	0.8503 (±0.0275)	0.7725
Ensemble	–	**0.7812**

Table 2: SubtaskA models performances. CV f1-score is used to record cross validation dev set scores, and the test score is generated by trained model predicting on released labeled test data.

Models	Subtask B Trial set score	Subtask B Test set score
BERT-Large	0.8695	0.8302
BERT-Large-Conv	0.9001	0.8425
BERT-Large-FFA	0.8795	0.8409
BERT-Large-GRU	0.8796	0.8486
Ensemble	–	**0.8579**

Table 3: Subtask B models performances. We use labeled data from subtask A as training set, subtask B trial data as dev set to select best hyperparameters, and test score is generated by trained model predicting on released labeled test data

training with subtask A data, the model should also works in the subtask B.

As shown in Figure 2a we noticed that, in subtask A, there is an obvious increasing tendency of f1 score until the 3rd epoch indicating the model have found the optimal parameters for fine-tuning. And for subtask B, which is shown in Figure 2b, best performance is always achieved in very early steps of initial epoch when fine-tuning the model and decrease all the way down, which proves that the model are learning common features cross the two different domains, but as the training process proceeds, more and more features about subtask A are learned, which cause the performance of subtask B decrease.

Learning rate tricks: Considering that the number of training dataset is too small to train a complex model, different learning rate are applied for different layers. Specifically, we apply a small learning rate for pre trained BERT layers, and a larger learning rate for new task specified layer.

3.3 Results

In the early stage of this competition, we have tried many non-BERT models, e.g. CNN (Kim, 2014), Transformer Encoder (Vaswani et al., 2017), Capsule Networks (Gong et al., 2018). However, none of the results from those models are competitive with the models based on BERT . The scores are summarized in Table 2 and 3. The result have shown that with the ensemble strategy of different models, scores of both tasks increases.

It should be noted here that, for every model of subtask A, we run 5-10 times training process repeatedly with different random seeds to ensure we can get a reliable evaluation result. For the final submission, we use the voting strategy to fuse all predictions of each model, and the ensemble results is 0.7812 and 0.8579 for subtask A and B respectively.

4 Conclusion

In this paper, we have introduced an empirical multi-perspective framework for the suggestion mining task of SemEval-2019. We propose an ensemble architecture for learning representations by using different classical models including CNN, GRU, FFA Network, etc. According to the obtained promising performances on both subtasks, we found that the pre-trained model by a large amount of unlabeled is critical for most nlp tasks, even for domain adaptation tasks without a specific neural architecture. In the future, in order to make more use of the dataset from different domains, adversarial gradient or common domain feature learning methods can be adopted along with pre-trained models to reach a better performance.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computer Science*.

Caroline Brun and Caroline Hagège. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Jingjing Gong, Xipeng Qiu, Shaojing Wang, and Xuanjing Huang. 2018. Information aggregation via dynamic routing for sequence encoding. *CoRR*, abs/1806.01501.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Kazuya Kawakami. 2008. *Supervised sequence labelling with recurrent neural networks*. Ph.D. thesis, Ph. D. thesis, Technical University of Munich.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. pages 170–178.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews.

Sapna Negi and Paul Buitelaar. 2017. Inducing distant supervision in suggestion mining through part-of-speech embeddings. *CoRR*, abs/1709.07403.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*.

Sapna Negi, Maarten De Rijke, and Paul Buitelaar. 2018. Open domain suggestion mining: Problem definition and datasets.

Colin Raffel and Daniel P. W. Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *CoRR*, abs/1512.08756.

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 640–645.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.