Yimmon at SemEval-2019 Task 9: Suggestion Mining with Hybrid Augmented Approaches

Yimeng Zhuang
Samsung Research China - Beijing (SRC-B)
ym.zhuang@samsung.com

Abstract

Suggestion mining task aims to extract tips, advice, and recommendations from unstructured text. The task includes many challenges, such as class imbalance, figurative expressions, context dependency, and long and complex sentences. This paper gives a detailed system description of our submission in SemEval 2019 Task 9 Subtask A. We transfer Self-Attention Network (SAN), a successful model in machine reading comprehension field, into this task. Our model concentrates on modeling long-term dependency which is indispensable to parse long and complex sentences. Besides, we also adopt techniques, such as contextualized embedding, back-translation, and auxiliary loss, to augment the system. Our model achieves a performance of F1=76.3, and rank 4th among 34 participating systems. Further ablation study shows that the techniques used in our system are beneficial to the performance.

1 Introduction

Suggestion mining is a trending research domain that focuses on the extraction of extract tips, advice, and recommendations from unstructured text. To better recognize suggestions, instead of only matching feature words, one must have the ability to understand long and complex sentences.

SemEval-2019 Task 9 provides the suggestion mining task (Negi et al., 2019). The task can be recognized as a text classification task, given a sentence collected from user feedback, participating systems are required to give a binary output by marking it as suggestion or non-suggestion.

To address this problem, we focus on solving long-term dependency on long and complex sentences. Consequently, we transfer Self-Attention Network (SAN), a successful model in machine reading comprehension field, in which long-term dependency is crucial, into this task. Furthermore, we also utilize multiple techniques to improve the suggestion mining system.

2 System description

In this paper, we consider suggestion mining as a text classification task. Figure 1 gives an overview of our model. First, the input text is converted into word embeddings with linguistic features. Then, we use several stacked semantic encoders to generate the hidden representations for each token. On top of that, a softmax output layer estimates the probability of the text being a suggestion.

2.1 Input encoding

The input encoding layer is in charge of encoding each token of the input text to singular vectors. Tokenization is completed during preprocessing. In our work, we adopt WordPiece embedding (Wu et al., 2016) and feed it into a pretrained language model (LM) to generate contextualized embeddings. Compared to context independent word vectors, such as widely used GloVe (Pennington et al., 2014), SGNS (Mikolov et al., 2013), contextualized vectors show significant advantages in disambiguation and sentence modeling. Besides, well-pretrained language model also transfers external knowledge to this task, full use of transfer learning is the key to the advance in modern neural natural language processing. On the choice of the pretrained language model, we compared two publicly available models ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), and we finally choose BERT for better performance. Due to the out-of-memory issues [1], we do not update the parameters of BERT during training, thus we only use it as a static feature extractor.

[1] https://github.com/google-research/bert#out-of-memory-issues

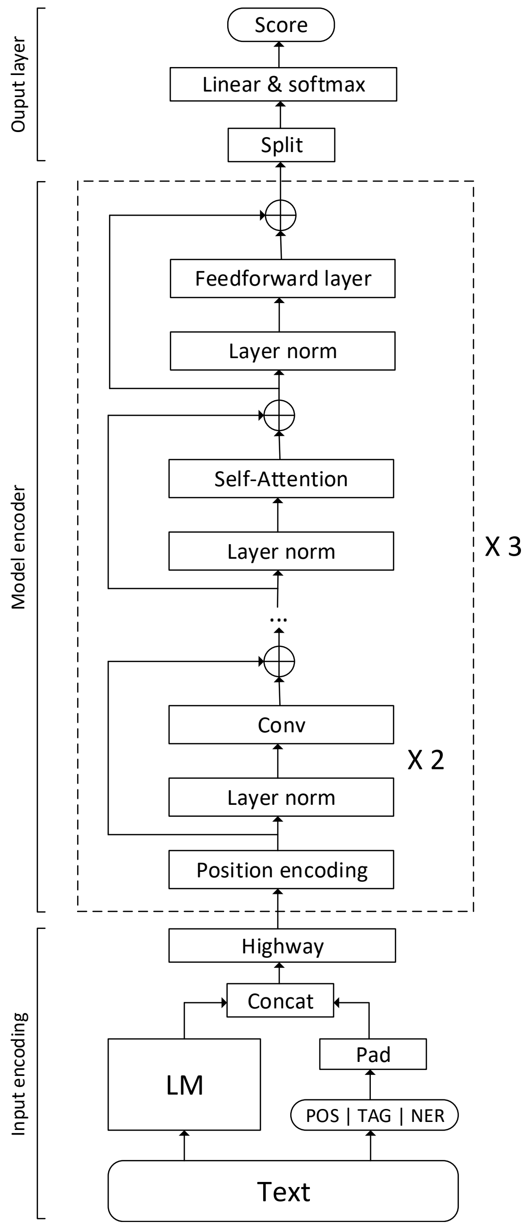

Figure 1: Overview of our system

Also, linguistic features are extracted to improve system performance further. In this work, we extract part-of-speech (POS) and named entities (NER) by spaCy [2]. Two kinds of part-of-speech granularity are used, primary POS and extended POS tag (TAG). In order to obtain linguistic feature sequences with the same length as the BERT outputs, we pad zero vectors at the start and end position of the linguistic feature sequences. Then, the contextualized vectors and linguistic feature vectors are concatenated. A two layers highway network (Srivastava et al., 2015) is also adopted on top of this representation. The vectors are projected to d dimensions immediately.

[2]https://spacy.io

2.2 Model encoder

The model encoder is the central part of our system, and it is in charge of modeling long-term dependency and extracting deep features. Because of the success of Self-Attention Network (SAN) (Shen et al., 2018) in various NLP tasks, we adopt a structure from QANet (Yu et al., 2018), which is a variant of Self-Attention Network, as our model encoder.

As is shown in the middle part of Figure 1, the model encoder is a combination of convolution, self-attention and feed-forward layer. This structure is repeated three times. The input vectors of this structure are firstly added by sinusoidal position embedding (Gehring et al., 2017) to encode a notion of the order in the sequence. The position embedding is calculated as follows:

$$\begin{aligned} PE_{i,2j} &= sin(i/10000^{2j/d}) \\ PE_{i,2j+1} &= cos(i/10000^{2j/d}) \end{aligned} \tag{1}$$

After that there are convolution blocks, following (Yu et al., 2018), depth-wise separable convolution layers are chosen for better generalization and memory efficiency. The model encoder is highly dependent on input layer normalization and residual connections, each block is in a uniform structure: *layer normalization / operation / residual connection*. In our system, we repeat convolution blocks two times for better and deeper local feature extraction.

In the self-attention layer, the scaled dot-product attention is computed:

$$A_i = softmax(\frac{QK^T}{\sqrt{d}}) \cdot V \tag{2}$$

Where Q, K, and V are query, key, and value respectively, they are the linear projection of each position in the input. As in (Vaswani et al., 2017), multi-head attention mechanism is adopted which integrates multiple scaled dot-product attentions.

$$A = [A_1; \cdots; A_i; \cdots; A_n] \cdot W^A \tag{3}$$

where A_i denotes the i-th head, $[\cdot]$ is concatenation operator, W^A is a trainable parameter.

At last, there is a fully connected block. In our method, it is a little different from the original work (Yu et al., 2018). We append a gate mechanism to refine tokens by their importance (Wang et al., 2017).

$$S = FFN^2(LayerNorm(H)) \odot G + H \quad (4)$$
$$G = G^*/max(G^*) \quad (5)$$
$$G^* = sigmoid(LayerNorm(H) \cdot W^G + b^G) \quad (6)$$

where we assume the input of this block is $H \in \mathbb{R}^{m \times d}$, m is the sequence length, FFN^2 denotes a 2-layer non-linear feed-forward network, $S \in \mathbb{R}^{m \times d}$ represents the output of this fully connected block. $G \in \mathbb{R}^{m \times 1}$ and $G^* \in \mathbb{R}^{m \times 1}$ are the output weight of the gate. $W^G \in \mathbb{R}^{m \times 1}$ and $b^G \in \mathbb{R}^1$ are trainable parameters. The maximum operation aims to select the maximum element in G^*, and the division operation normalizes these weights so that the maximum weight in G is always one.

2.3 Output layer

Given the output $S = [s_1, s_2, \cdots, s_m]$ of previous layers, this output layer converts these hidden representations into the final probability. Since we have adopted BERT in the input encoding layer, the first vector of the sequence is a special classification token [CLS], which can be used as the representation of the whole sentence. We split the matrix S and take the first vector $s_1 \in \mathbb{R}^d$. The probability of the input text being a suggestion text is estimated as follows.

$$p = softmax(W^p \cdot s_1 + b^p) \quad (7)$$

where $W^p \in \mathbb{R}^{2 \times d}$ and $b^p \in \mathbb{R}^2$ are trainable parameters, $p \in \mathbb{R}^2$ denotes the output probabilities including "yes" probability p^1 and "no" probability p^0.

2.4 Loss

We treat this task as a text classification problem and use log-loss as the loss function.

$$\mathcal{L}_0 = \frac{1}{N} \sum_{i=1}^{N} y_i log(p_i^1) + (1 - y_i) log(p_i^0) \quad (8)$$

where N is the number of examples, $y_i \in \{0, 1\}$ represents the label of i-th example, p_i^1 and p_i^0 are the predictions.

Besides, in order to better recognize important tokens and filter trivial tokens out, we add an auxiliary loss to discourage large weights in G in Equation 4. Thus only those tokens that have contributions to the classification have non-zero weights in G.

$$\mathcal{L}_1 = \beta \frac{1}{N} \sum_{i=1}^{N} \|G_i\|_1 \quad (9)$$

where $\|\cdot\|_1$ denote 1-norm, β is a hyper-parameter, we use $\beta = 10^{-3}$ in this work. The final loss is the sum of $\mathcal{L}_0$ and $\mathcal{L}_1$.

2.5 Class imbalance

As is pointed out in (Negi et al., 2019), suggestions appear sparsely in online reviews and forums, and this makes class imbalance a critical problem. For simplicity, we do not take measures during training. In inference, we slightly adjust the predicted probability and divide it by a *priori*, which is the rate of positive examples in training data.

$$\begin{aligned} p &= softmax((W^p \cdot s_1 + b^p) \times 10) \\ p^{1*} &= p^1/\mathit{priori} \quad (10) \\ p^{0*} &= p^0/(1 - \mathit{priori}) \end{aligned}$$

where p^{1*} and p^{0*} are the actual predictions for inference, the system outputs "yes" when p^{1*} is larger than p^{0*}. Other symbols are the same as in Equation 7.

2.6 Back-translation

Because the given training data set is not large, we also utilize a data augmentation technique to enrich the training data. The data augmentation technique we used is back-translation (Yu et al., 2018). Specifically, we first translate the given training data into Chinese by a neural machine translation system and then translate it back into English by another neural machine translation system. The two neural machine translation systems are trained on a subset of the WMT18 data sets [3]. Both original training data and the augmented data are applied to train our text classification system, but the augmented data is given a small weight (=0.2) when calculating the loss.

3 Experiments

3.1 Setup

SemEval 2019 Task 9 Subtask A [4] provides a suggestion mining data set collected from feedback

[3]http://statmt.org/wmt18

[4]https://competitions.codalab.org/competitions/19955

posts on Universal Windows Platform. The data set is split into an 8980-example training set, a 592-example validation set, and an 833-example test set. Since the organizer does not limit the usage of the validation set, we merge it into training data and train our model through the k-fold cross-validation method. Specifically, we split all training data into eight subsets, and guarantee the rate between the number of positive examples and the number of negative examples is about 1:3 in each subset. The preprocessing process is implemented as the description in section 2.1.

The kernel size of convolution layers is 7, and the hidden size d is 256, the number of heads is 8 in multi-head self-attention layers. Adam optimizer (Kingma and Ba, 2014) with learning rate 0.0008 is used for tuning the model parameters. The mini-batch size is 32. For regularization, the dropout rate is set to 0.1. The submission predictions are obtained by integrating eight runs through voting.

Approach	Test F1	
Baseline	26.8	
1st ranked system	78.1	
2nd ranked system	77.8	
3rd ranked system	77.6	
Our system	76.3	
5th ranked system	74.9	
Ablation - single model	F1	Δ
Our full system	73.3	-
- Contextualized embedding	67.6	-5.7
- Back-translation	71.4	-1.9
- Priori	72.5	-0.8
- Linguistic features	72.6	-0.7
- Auxiliary loss	72.7	-0.6

Table 1: Performance of the Top 5 systems on the leaderboard of subtask A, and our ablation experiments.

3.2 Results

Table 1 shows the main results on the test set. Compared with the rule baseline, participants improve their performance with substantial gains. Our system achieved F1=76.3 on the test set and ranked 4th among all 34 teams.

In order to evaluate the individual contribution of each feature, we run an ablation study. Contextualized embedding is most critical to the performance, and it concludes that transferring common sense by learning large corpus is vital for this task. Back-translation accounts for about 2% of the performance degradation, which clearly shows the effectiveness of data augmentation. Besides, linguistic features, auxiliary loss, and *priori* are also beneficial to the system.

Base		Large	
F1	F1	F1	F1
72.9	71.8	75.4	75.9
72.9	70.6	73.9	76.6
74.3	74.8	74.0	76.2
75.3	73.5	76.3	75.1
Ensemble			
#	F1	#	F1
2	72.9	8	74.5
4	74.7	12	75.5
16	76.3	Oracle	79.6

Table 2: Performance of every single model and ensemble models.

3.3 Ensemble models

Table 2 reports the effect of ensemble. To obtain the submission predictions, we trained eight models on the eight subsets mentioned in section 3.1. Four of these models are based on a 110M parameters base BERT model, and the other four are based on a 340M parameters large BERT model. The performance of every single model is reported. It is evident that different data partition causes quite a large performance variance. Thus, ensemble is necessary. The result shows that as the number of models increases the ensemble effect improves. Also, we experiment by searching the optimal model combination assuming the test label is known to show the performance limitation (the Oracle performance).

4 Conclusion

In this work, we adopt multiple techniques to improve a suggestion mining system. The core of our system is a variant of Self-Attention Network (SAN), which originates from the machine reading comprehension field. Based on this model, techniques, such as contextualized embedding, back-translation, linguistic features, and auxiliary loss, are investigated to improve the system performance further. Experimental results illustrate the effect of our system. Our model achieves a performance of F1=76.3, and rank 4th among 34 participating systems.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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)*.

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.

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)*, volume 1, pages 2227–2237.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018. Bi-directional block self-attention for fast and memory-efficient sequence modeling. *arXiv preprint arXiv:1804.00857*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.