Neural Networks for Explanatory Sentence Recognition

Neural Networks for Explanatory Sentence Recognition

Yang Lin[a], Chao Wang[b], Yao Tong[c], Rui Zhang[d], Bowene Gu[e] and Leilei Zhao[f]

State Grid Heilongjiang Electric Power Co., Ltd. Electric Power Research Institute, Heilongjiang 150000, China

[a]1104490433@qq.com, [b]1229104738@qq.com, [c]2001-tongyao@163.com, [d]407112939@qq.com, [e]3466800@qq.com, [f]hi_zll@xinlang.com

Abstract. Opinion mining has been received extensive interests in natural language processing community. Previous work drives much attention on the polarities of opinions, as well as their holders and targets. Little work concerns the reason that lead opinion holders express such opinions, which are also as important for opinion mining. In this work, the first neural models for explanatory sentence recognition is proposed, detecting whether a given opinionated sentence contains the explanatory information for the expressed opinion. Experimented results show that the proposed neural models achieve better performances than baseline discrete models on two Chinese datasets.

1. Introduction

Opinion mining, which extracts structural opinion elements, including holders, expressions, targets as well as polarities, has been investigated extensively in recent years [1, 2]. In the opinionated sentences, a number of them tend to illustrate their reasons for such opinions. For example, "I dislike the phone game, *because it is too large*", the bold part is the explanatory segment for the main elements. For example, in the case that producers want to improve their products.

In this work, the task of explanatory sentence (ES) recognition is concerned, which aims to detect whether an opinion sentence contains an explanatory part. Table 1 shows several examples of the task. As shown, the task can be modeled as a simple classification problem. Thus, the key point remaining is the feature selection procedure.

Table 1. Examples of ES and non-ES

Opinionated Data	Type
IPhone screen is good, the resolution is really high.	ES
This phone is awesome.	non-ES

Traditionally, features are defined according to manually-designed feature templates, and use one-hot vectors with high-dimensions. This representation method suffer from the feature sparsity problems, as low-frequency features are usually trained inefficiently [3]. Recently, neural-based models have been shown promising results for a number of natural language processing tasks. In this work, a discrete baseline with human-designed features is suggested. Then several neural network models are propose , include a convolution neural network(CNN) model, a gated recurrent neural(GRNN) model, and a long-short term memory(LSTM) model. Also, this work investigates several different settings of certain neural network structures. We conduct experiments on two datasets phone and hotel, both of which are

Chinese language. The experiments show that neural network models can achieve significantly better results on both datasets. While the neural models exploiting RNNs are not consistently better than the CNN-based model. This work perform detailed analysis work as well, explaining the key settings of the neural network structures.

2. Related Work

2.1. The Application of Explanatory Sentence Recognition

Kim et al. [2] first proposed explanatory sentences extraction in the paper published by SIGIR'13. They applied two general methods to scoring explanatoriness. He Yu et al. [4] applied an automatic coding technology, used word embeddings as input and apply sum-pooling to code sentence feature automatically. Automatic learned features are used as input under the framework of supported vector machines (SVM). Fang et al. [5] decomposed explanatory sentence into two sub-problems: sentence informative sorting and structured emotional analysis, and perform joint prediction through duality decomposition to evaluate emotional explanation.

2.2. Methods of Explanatory Sentence Recognition

Explanatory sentence recognition can be modeled as a classification task, and there are many studies on it. Anthony Khoo et al. [6] compared three various popular classification algorithms with various popular feature selection methods. Lin Jianghao [7] adopted two kinds of feature selection methods to carry on sentence classifications, one was using constructed emotion dictionary to extract feature, and second one was based on syntactic dependency. Yoon Kim [8] applied CNNs and pre-trained word embeddings to classifier and achieved excellent results on multiple benchmarks. Nal Kalchbrenner et al. [9] proposed dynamic convolutional neural network and dynamic k-max pooling to semantic modelling of sentences. Liu Dexi et al. [10] applied N-gram feature to classify Chinese microblogging emotional words and their experiment results showed this method is better than traditional covariance method.

3. Models

3.1. Baseline Model

There are so many feature selection methods but most of them lack robustness and complex [4, 5], so this work choose n-gram as the sentence features in the baseline. N-gram is a simple and reliable feature in classification tasks [10]. In the experiments, N-gram feature templates are Uni-gram, Bi-gram and Tri-gram.

3.2. Overview of neural models

The choosed neural models are illustrated in Figure 1. First of all, models encode words of sentence $w_1...w_n$ into word representations e (w_1)...e (w_n). The model get word embeddings from look-up table by the embedding look-up function e (•). Then neural networks encode word representations into hidden layer features $H = \{h_i, (1 \leq i \leq n)\}$. In this paper, different kinds of neural networks are applied, and introduce them in next sections. Next, this work uses $max, min $ and avg pooling methods to sample hidden layer features to get the sentence feature F_s:

$$F_s = p_{max}(H) \oplus p_{min}(H) \oplus p_{avg}(H) \tag{1}$$

Where, p (.) is a pooling method, F_s is the input into classifier to get the result.

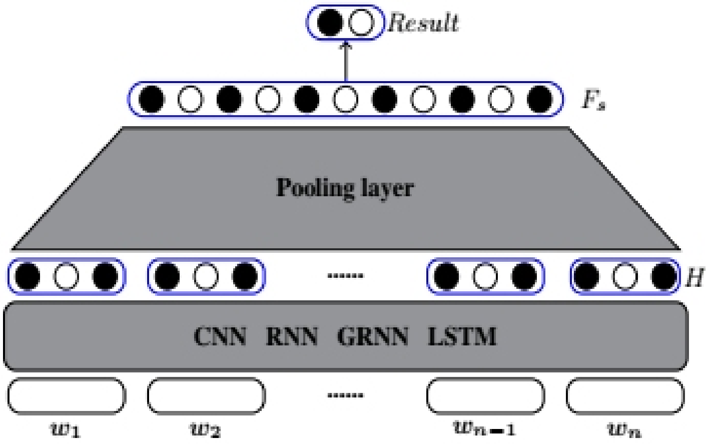

Figure 1. Overview of neural models

3.3. Convolution neural network

There are a set of nodes in convolution neural network (CNN). They extract contextual information from word representations. A node of CNN concatenate a number of contiguous word representations and then through non-linear hidden layer:

$$h_i = \tanh (W \bullet (e (w_{i-2}) \oplus e (w_{i-1}) \oplus e (w_i) \oplus e (w_{i+1}) \oplus e(w_{i+2})) + b) \tag{2}$$

Where, the matrix W and vector b are model parameters, and tanh() is activation function. CNN can extract the local features and combine them automatically.

3.4. Recurrent neural network

The encoding method in recurrent neural network (RNN) is totally different from CNN. In RNN, the encoding result $H = \{h_i, (1 \le i \le n)\}$, cooperate with word representations to encode themselves as follow:

$$h_i = \tanh (W \bullet h_{i-1} + U \bullet e (w_i) + b) \tag{3}$$

RNN can extract global features from entities, because information can be transferred between adjoint nodes.

3.5. Gated recurrent neural network

Gated recurrent neural network (GRNN) is extended from RNN by gated mechanism. Gates are just control information fluidity between input node and output node as follow:

$$ri = \sigma (Wr \bullet hi-1 + Vr \bullet e (wi) + br) \tag{4}$$

$$ui = \sigma (Wu \bullet hi-1 + Vu \bullet e (wi) + bu) \tag{5}$$

$$yi = \tanh (Wy \ (ri \odot hi-1 + Vy \bullet e (wi) + by) \tag{6}$$

$$hi =(1 - ui) \odot hi-1 + ui \odot yi) \tag{7}$$

Where, W_*, V_* and b_* are model parameters.

3.6. Long short term memory

Long short term memory (LSTM) is also extended from RNN. One node in LSTM is consisted by three gates and a cell as follow:

$$ii = \sigma (Wi \bullet e (wi) + Ui \bullet hi\text{-}1 + Vi \bullet ci\text{-}1 + bi) \quad (8)$$

$$fi = \sigma (Wf \bullet e (wi) + Uf \bullet hi\text{-}1 + Vf \bullet ci\text{-}1 + bf) \quad (9)$$

$$ci = \tanh (Wc \bullet e (wi) + Uc \bullet hi\text{-}1 + bc) \odot ii + ci\text{-}1 \odot fi) \quad (10)$$

$$oi = \sigma (Wo \bullet e (wi) + Uo \bullet hi\text{-}1 + Vo \bullet ci\text{-}1 + bo) \quad (11)$$

$$hi = \tanh (ci) \odot oi \quad (12)$$

Where, i_i, f_i, o_i are gates, c_i is cell, and W_*, V_*, b_* are model parameters.σ is activation function sigmoid ( ), and $\odot$ means element-wise product. Gates control the information flowed into the cell, and cell can store information transferred from other nodes. LSTM can transfer information to remote nodes to solve long-term dependency problem.

3.7. Bi-directional RNNs

This work applies three kinds of RNNs encode word representations with two different directions [11]. The previous and following contextual information can be written into hidden features $H_L = \{h_{li}, (1 \leq i \leq n)\}$ and $H_R = \{h_{ri}, (1 \leq i \leq n)\}$ respectively. Then the work concatenates two sets of vectors H_L and H_R into new features $H' = \{h'_i, (1 \leq i \leq n)\}$. This method can only be applied to RNNs as the H_L and H_R of CNN are equivalent.

3.8. Character features

The character features shown their effectiveness in Twitter sentiment analysis [12].This work propose to enhance the original sentence neural features with character features.Also, this paper apply convolution layer and pooling layer to extract the character features C_s of sentences and combine them with original sentence features F_s.

4. Training Method

During training, the models minimize loss function according to annotated training instances and the optimal method is Adagrad. The cross-entropy as the loss function of the model is applied as follow:

$$L(\theta) = -\sum gti \ \log (pti) \quad (13)$$

Where, θ is a set of model parameters, p_{ti} means the probability of the ith label of the training instance and g_{ti} is gold answer of it. The training method of the model is online learning in which training instances are available in a sequential order and use to update model parameters for future training instances at each step. To prevent over-fitting in the neural models, this work drops the some feature of word embeddings and hidden layers randomly during training.

5. Experiments

5.1. Data & Evaluation metrics

The experimental corpus are consisted of 20000 pieces instances, which are collected from Internet and annotated by manual. During training, we split 10% of the corpus as development sets to tune the hyper-parameters and report ten-fold cross-validation results.

Table 2 shows statistics of experimental corpus.

Table 2. Statistics of experimental data

Domain	Avg Words	Avg Characters	ES	Non-ES	Total
Phone	22	33	5000	5000	10000
Hotel	16	23	5000	5000	10000

5.2. Parameter settings

Using accuracy to evaluate the performance of models, there are many hyper parameters in the neural models. This work adjusts them according to the performance in development sets. The hidden layer size is set as 100 and the dropout rate as 0.1 for all neural models. Word context of convolution layers is 5. Learning rate in the neural models for Adagrad is 0.01 and regularization parameter is 10^{-8}.

5.3. Word & character embeddings

This work collected phone and hotel service comments on the web and use those raw texts to train the word embeddings by Word2vec [13], respectively. We input word embeddings into neural models and control fine-tune them or not. During the training, if those word embeddings could not be fine-tuned, they are stable, and if they are fine-tuned, those are treated as model parameters. About character embeddings, this work don't use pre-training strategy, and character lookup table is initialized by random values.

6. Results Analysis

6.1. Influence of Fine-tuning

In the experiments, this work apply two different kinds of strategies to word embeddings, fine-tuning word embeddings or not. When word embedding is fine-tuning, the word embeddings of in-vocabulary words are treated as model parameters and adjust value during training. This strategy can improve accuracy of model. But the words out of vocabulary are useless to model because hidden layer of models are tuned with word embeddings of in-vocabulary words. Figure 2 shows effectiveness of fine-tuning strategy on neural network models on hotel datasets with random initialization. And if the fine-tuning strategy is applied to pre-training models, the model without fine-tuning shows better performance. Because with fine-tuning, only training data set can use pre-training word embeddings, which can makes fine-tuning models disadvantageous in handling test data sets with many out of vocabulary words.

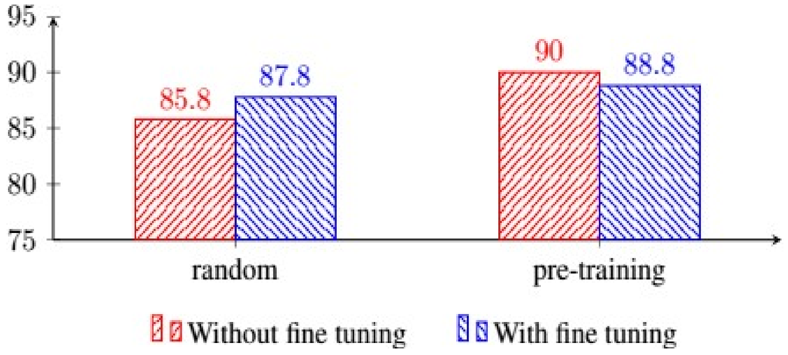

Figure 2. Influence of fine-tuning

6.2. Pre-training & character features

The word representations and character features can influent performance of neural models [9, 10]. This work enhances original neural sentence features with character features which extracted by CNN. In figure 4, character features are applied to neural models, and it can improve the accuracy about 2-3% on both datasets. And there are two initialization methods of neural models, pre-training and random value. Comparing two methods, pre-training can improve about 3-4% in accuracy. When this work applies both of them to neural models together, it can improve the accuracy about 5%.

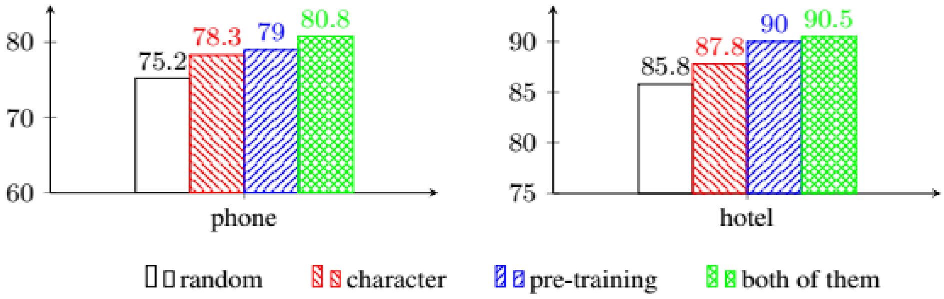

Figure 3. Influence of different strategies

6.3. *Performance using different models*

In this experiment, the baseline models and neural models are compared. In order to mask other interference factors, the lookup table of neural models are initialized by word embedings. Their performances shown in Table 3. The accuracy of all neural models are higher than baseline model. And in different domain, the hotel datasets always shows higher accuracy than phone in same model.

Table 3. Accuracies of different models

Domain	Baseline	CNN	RNN	GRNN	LSTM
Phone	75.8	80.8	79.6	81	80.7
Hotel	86.1	90.5	88.8	89.7	90.0

6.4. *Performances using Bi-RNNs*

This work applies different kinds of RNNs in neural models. In figure 3, it could be seen that the accuracy of RNN is less than GRNN and LSTM, because GRNN and LSTM have gated mechanism to control information fluidity, and this mechanism can solve long-term dependency problem. Comparing LSTM and GRNN, both of them are extended by RNN. But LSTM has more parameters than GRNN, so LSTM need more data to get better performance. The training data is no enough, so GRNN shows the more competitive results here. When bi-direction method are applid in three kinds of RNNs, accuracy of them improved about 1% because bi-direction method can extract more comprehensive features.

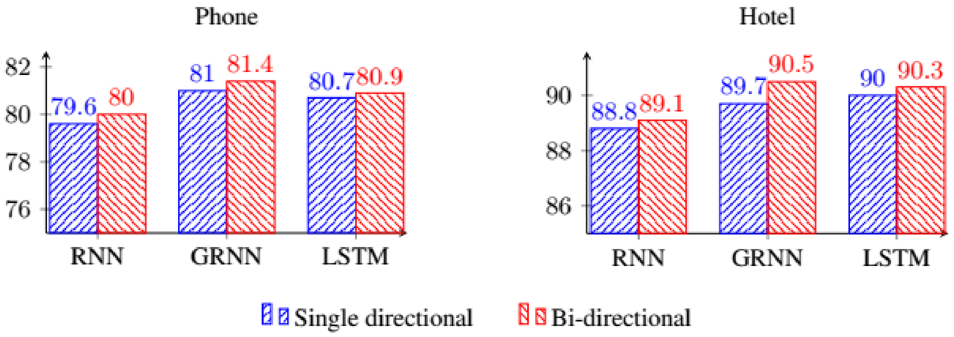

Figure 4. Influences of different RNNs

6.5. *Influence of multiply convolution layers*

Multiply convolution layers can be applied to CNNs. In this experiment, the word context of convolution layers are adjusted from 5 to 3, because smaller word context can reduce model parameters, and deep

CNNs have more model parameters than normal CNNs, but the scale of data is not enough. During training, this work applies fune-tuning to models. Figure 5 shows performances with different deep CNNs. The performance increases with the number of convolution layers in hotel dataset, because multiply convolution layers can combine more complex syntactic features. In phone datasets, 2 layer CNN shows better performance than 3 and 4 layer CNNs. This is because 3 and 4 layer CNNs has more model parameters and they are easy to over-fitting.

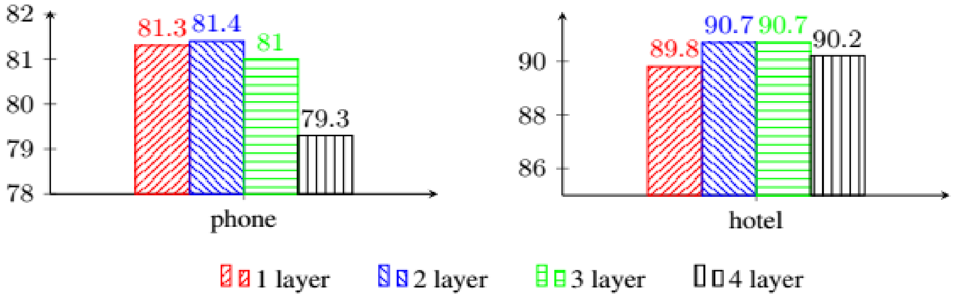

Figure 5. Performances with different deep CNNs in hotel datasets: x layer means apply x convolution layers to CNNs

7. Summary

In this work, the recognition of explanatory sentences with neural models are explored, which show more competitive results than baseline with higher accuracy. In neural models, this work explores different kinds of strategies applied to neural models and study effectiveness of those strategies, including pre-training, character features and fine-tuning. Different kinds of neural networks are also compared which verify the effectiveness of gated mechanism, bi-directional feature encoding methods and multiply layers.

References

[1] Kim H D. General unsupervised explanatory opinion mining from text data [J]. Dissertations & Theses - Gradworks, 2013, pp. 102.

[2] Hyun Duk Kim, Malu G Castellanos, Meichun Hsu, ChengXiang Zhai, Umeshwar Dayal, and Riddhiman Ghosh. Ranking explanatory sentences for opinion summarization. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, ACM, 2013, pp. 1069 – 1072.

[3] Fabrizio Sebastiani. Machine learning in automated text categorization. ACM computing surveys (CSUR), 2002, 34 (1), pp. 1 – 47.

[4] Yu He, Da Pan, and Guohong Fu. Chinese explanatory opinionated sentence recognition based on auto-encoding features. Journal of Peking University (Natural Science Edition), 2015, 51 (2), pp. 234 – 240.

[5] Lei Fang, Qiao Qian, Minlie Huang, and Xiaoyan Zhu. Ranking sentiment explanations for review summarization using dual decomposition. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, ACM, 2014, pp. 1931 – 1934.

[6] Anthony Khoo, Yuval Marom, and David Albrecht. Experiments with sentence classification. In Proceedings of the 2006 Australasian language technology workshop, 2006, pp. 18 – 25.

[7] Lin Jianghao, Yang Aimin, Y Yongmei, et al. Classification of microblog sentiment based on na¨ıve byaesian. Computer Engineering & Science, 2012, 34 (9), pp. 160 – 165.

[8] Yoon Kim. Convolutional neural networks for sentence classification, 2014, 5882.

[9] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences, 2014, pp. 1404 - 2188.

[10] Dexi Liu, Jianyun Nie, jing Zhang, Xiaohua Liu, ChangxuanWan, and Guoqiong Liao. Chinese

microblogging emotional word extraction: N-gram for the classification of features. Chinese Journal of Information, 2016, 30 (4), pp. 193 – 205.

[11] Alex Graves and J¨urgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks, 2005, 18 (5), pp. 602 – 610.

[12] SaifMMohammad, Svetlana Kiritchenko, and Xiaodan Zhu. Nrc-canada: Building the state of-the-art in sentiment analysis of tweets, 2013, pp. 6242.

[13] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 2014, 15 (1), pp. 1929 – 1958.