High-order neural network in entity relationship extraction

Wei Liu*, Hongchang Chen, and Ruiyang Huang

National Digital Switching System Engineering & Technological Research Center, China

Abstract. In this paper, a kind of high-order neural network is proposed to extract entity relations in natural language. In this kind of network, different parameters absorb non-overlapping information from separated data respectively, which makes parameters more significant for understanding. This neural network can alleviate overfitting problem in some degree. When solving the task of entity relationship extraction, this network can give a result no worse than current methods.

1 Introduction

Information extraction is an important task especially in natural language processing (NLP) field. Extracting entity relations from natural language is really a significant part of information extraction. Recently there seems a bottleneck existing in promoting the performance of relation extraction models. End to end relation extraction method is rediscovered and combined with LSTM to promote the performance [1], and showing comparable performance to CNN-based models on nominal relation classification (SemEval-2010 Task 8). This end to end model has been proved effective in SemEval-2017 Task 10 [2], whereas the same bottleneck seems still exists. A great many of reasons can cause this happen, such as inappropriate representation of entities and relations, inaccurate entities recognition or overfitting of models. Neural relation extraction with selective attention over instances [3] is a well-behaving model in this task. Its trained model can describe its train dataset well with a high precise greater than 95%, but harvests a precise of less than 80%. The precise and recall on test dataset wouldn't climb up simultaneously, even though the model can describe the train dataset almost perfectly, which means there existing an extreme overfitting.

As a traditional task, relation extraction has been studied for decades by a great many of researchers and a landmark Deep Neural Network [4] especially approaches based on convolutional neural network [5] can separate these studies into two stages. Before DNN, Hand-built patterns [6] and handcrafted features [7] are popular approaches, which may be greatly complicated but can be understood. DNN-based models can work well generally on relation extraction tasks, meanwhile many novel traditional ideas are abandoned. Approaches based on LSTM [1] and CNN [3] seems not compatible with traditional handcrafted features based methods flexibly, which is really a great loss. It will happen easily when the parameters of DNN based approaches are difficult to understand.

* Corresponding author: wliu.ndsc@gmail.com

In this paper, a neural model whose parameters can be explained primarily is proposed, which has a promotion on the performance of relation extraction for its inherent ability of overcoming overfitting. Different from traditional neural network, the network in this paper has two parts (not layers), which are called orders in the remaining of this paper. The first order employs a CNN-based neural network. In the second order, the parameters of the first order will participate to train new parameters as input, under the supervision of the first order. Higher orders are not constructed in the experiments for its serious time consuming, but it is easily to construct according to the theories. Once each order is constructed and trained completely, the parameters in different order achieved different meanings. Moreover, the meanings are not correlative with each other in stochastic. With this method, each group of parameters in each order can descript a different feature, which is not the very feature that we are familiar with though. Meanwhile, the model is born with an ability of overcoming overfitting for the higher orders can always supervise the train process of the lower orders, which seems like the introspection of human beings.

Fig. 1. Second-order neural network.

Figure 1 shows the structure of second-order neural network. At below of this figure, it means the training process of the first order. Once the parameters of first order network are trained, the record of parameters change will be inputted to the second order network. The output of the second order network is the kernel of first order network and will update the first order network. When testing this model, we only use the first order network with the kernel outputted by the second order network.

The experimental results will show that there is a real promotion in relation extraction when the second order even a simplest mean-method is constructed. And there is a further promotion, when a more complex model is employed in the second order. Moreover, the proposed model can still behave better when the training dataset is reduced, which may be significant when it is used to process cross-sentence N-ary relation extraction [8] or other tasks with small quantity train dataset.

The contributions of this paper can be summarized as follows:

- An approach whose parameters can be explained is proposed and works well on relation extraction task.
- A new way to overcoming overfitting problem is discovered.
- The proposed model has a powerful expansibility and portability.

2 High order Neural Network

2.1 First-order Neural Network

In a task, the training data is noted as $\mathbf{D}$, and if the whole neural network is represented as $M(\cdot)$ simply, then we can compute the output of this network:

$$\mathbf{y} = M(\boldsymbol{\theta}, \mathbf{D}) \tag{1}$$

where all the parameters are noted as θ simply.

In the stage of training, an optimism or cost function is employed to conduct the parameters in a so-called corrected direction. We note this function as $J(\cdot)$, our goal is to find the best parameter θ_0 :

$$\theta_0 = \arg \min_{\theta} J[M(\theta, D)] \quad (2)$$

Once we get the θ_0 , we finish our training work.

2.2 Second-order Neural Network

Usually 1st-order neural network can satisfy the task, but for a complex task, it seems so simple that we have to overlying too many layers. Now let's do some change.

In a task, the training data is sliced into two parts D_1, D_2 , also we note a neural network as $M_1(\cdot)$ simply, the final output can be computed as following:

$$y = M_1[\theta(t), D_1] \quad (3)$$

where all the parameters are noted as $\theta(t)$ simply.

Also, in the stage of training we need a cost function $J(\cdot)$ to conduct the training steps. Now our goal is to find the best $\theta(t_0)$:

$$\theta(t_0) = \arg \min_t J[M(\theta(t), D_1)] \quad (4)$$

While the $\theta(t_0)$ is the best parameter for $M_1(\cdot)$, $M_1(\cdot)$ with $\theta(t_0)$ is not the best model for the whole task, for the training data D_2 have no contribution for $\theta(t_0)$. $\theta(t_0)$ only tells what the model can learn from D_1 . But we can give an expression to the best parameter noted as θ_0 :

$$\theta_0 = \sum_i \alpha_i e_i \quad (5)$$

where α_i is component coefficient of e_i , and $e_1, e_2, \dots, e_i, \dots, e_n$ are a group of linear independent vectors filling the whole space that $\theta(t_0)$ can fill. This group of vectors tell what the model cannot learn from the training data D_1 . And the value of parameter $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_n]$ will learn from D_2 .

While it is not always convenient to do find a group of these vectors $e_1, e_2, \dots, e_i, \dots, e_n$, we can always do some fitting work which all the machine learning methods do well including neural network.

Suppose $\theta(t)$ is initialized with $\theta(0)$ in the start of training and stopped at $\theta(t_0)$, as $\theta(0)$ is randomized, $\theta(t)$ can be treated as a stochastic process with $t \in [0, t_0]$. Obviously $\theta(t)$ is related to $M_1(\cdot), J(\cdot)$ and D_1 .

Theory: $\theta(t)$ is an independent increment process if.f. data in D_1 is independent and being exploited independently.

Proof:

$$f(M_1(t), J(t), D_1(t)) = \frac{d\theta}{dt} = \frac{\partial \theta}{\partial M_1} \cdot \frac{\partial M_1}{\partial t} + \frac{\partial \theta}{\partial J} \cdot \frac{\partial J}{\partial t} + \frac{\partial \theta}{\partial D_1} \cdot \frac{\partial D_1}{\partial t} \quad (6)$$

$$\forall t_1, t_2, t_3, t_4, 0 \leq t_1 < t_2 \leq t_3 < t_4 \leq t_0,$$

$$\forall t \in (t_3, t_4), s \in (t_1, t_2), E[D_1(t) \cdot D_1(s)] = E[D_1(t)] \cdot E[D_1(s)] \text{ is true, then}$$

$$E\{[\theta(t_4) - \theta(t_3)|\theta(t_3)] \cdot [\theta(t_2) - \theta(t_1)|\theta(t_1)]\} \quad (7)$$

$$= E\left\{ \int_{t_3}^{t_4} f(M_1(t), J(t), D_1(t)) dt \int_{t_1}^{t_2} f(M_1(s), J(s), D_1(s)) ds \right\}$$

$$\begin{aligned}
&= E \left\{ \int_{t_3}^{t_4} f(M_1(t), J(t), D_1(t)) dt \right\} \cdot E \left\{ \int_{t_1}^{t_2} f(M_1(s), J(s), D_1(s)) ds \right\} \\
&= E[\theta(t_4) - \theta(t_3) | \theta(t_3)] \cdot E[\theta(t_2) - \theta(t_1) | \theta(t_1)]
\end{aligned}$$

Moreover, if we sample $\theta(t)$ at t_i , $0 < t_{i-1} < t_i < t_o$ and employ SGD to exploit D_1 , then $\theta(0), \theta(t_2), \theta(t_3), \dots, \theta(t_n)$ will behave as a Markov chain, where $t_n = t_o$.

Here is our fitting work, we employed another neural network $M_2(\cdot)$ to finish the task: find $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_n]$, s.t.

$$\theta_o \approx \sum_{i=3}^n \alpha_i [\theta(t_i) - \theta(t_{i-1})] + \alpha_2 \theta(t_2) + (\alpha_1 - \alpha_2) \theta(0) \quad (8)$$

For regulation, let $\theta(t_0) = 0$ and $\theta(t_1) = \theta(0)$, then

$$\theta_o \approx \sum_{i=1}^n \alpha_i [\theta(t_i) - \theta(t_{i-1})] \quad (9)$$

The independence of each element in $\{\theta(t_i) - \theta(t_{i-1})\}$ ensure that this kind of α exists with great probability.

To ensure $M_2(\cdot)$ collaborate with $M_1(\cdot)$ efficiently, D_2 should supervise the training result of $M_2(\cdot)$ through the intermediary $M_1(\cdot)$. And $M_1(\cdot)$ should produce enough training data for $M_2(\cdot)$ by choosing different initialization value of $\theta(0)$.

Concretely, suppose D_θ is the training data produced by $M_1(\cdot)$,

$$\alpha = \arg \min_{\alpha} J\{M_1[M_2(\alpha, D_\theta), D_2]\} \quad (10)$$

Once we get the α , a approximation of θ_o can be computed with D_θ .

2.3 Higher order Neural Network

While higher order neural network may consume a great deal of computing resource, it is worth when the complexity of each single order is reduced, besides meaning or feature of parameters in each order is related to different data which makes it more explicit to solve different problems.

It is a natural progression of derivation from 2nd-order neural network to Nth-order neural network. Training data D has been splitted into N package $D_1, D_2, \dots, D_N$ manually and N 1st-order neural network has been chosen, which are noted as $M_1(\cdot), M_2(\cdot), \dots, M_N(\cdot)$ respectively. Parameters to $M_i(\cdot)$ are noted as θ_i . If we choose $J(\cdot)$ as the cost function, θ_{io} , the best value of θ_i will be generated from the following formulas:

$$\theta_{io} = \arg \min_{\theta_i} J\{M_i[\theta_i(\theta_i), D_i]\} \quad (11)$$

$$\theta_{j-1}(\theta_i) = M_j[\theta_j(\theta_i), D_{\theta_{j-1}}], j = 2, 3, \dots, i-1 \quad (12)$$

$$\theta_{i-1}(\theta_i) = M_i[\theta_i, D_{\theta_{i-1}}] \quad (13)$$

where $D_{\theta_{j-1}}$, $j = 2, 3, \dots, i$ are the training data produced by M_{j-1} .

In the iteration, there is a great advantage that different parameters are trained by different data, after which these parameters can collaborate with others efficiently. This property makes it possible to let people understand the meaning of parameters. Moreover, the parameters to i th order only learn the knowledge which has not learnt by the first $i-1$ orders, for the same information will not produce differential coefficient to the higher order. Besides there is a windfall that it is compatible with other machine learning models which means any order of the model may allow replacing by a classic machine learning method. While its consumption does not increase too fast for the training data will be split into smaller, too higher order neural network is still not recommended.

3 Experiments

Our experiments are intended to demonstrate that the proposed model can release the overfitting problem and achieve a better performance

3.1 Dataset and Evaluation Metrics

The model is evaluated on the dataset developed by Riedel et al. in 2010 [9], which was also used by Hoffmann [10] and Surdeanu [11] and Lin [3]. By aligning Freebase relationships with the corpus of the New York Times (NYT), this dataset is created easily. The entity mentions are found using the entity tagger named Stanford [12]. To demonstrate the effectiveness, we separate the dataset into two parts, one was produced before 2007, which is used for training, and the other part is produced in 2007, which is used for testing. The model will be evaluated using precise-recall curve, which is common used in entity relation extraction tasks.

3.2 Experimental Settings

3.2.1 First order settings

The first order employs the CNN+ATT model proposed by Lin et al in 2016 [3] and inherits the parameter settings and preprocessed data, which include the word embedding developed by word2vec tool and relation instance sentences. The words appearing less than 100 times are deleted and multiple words entity mentions are concatenated.

3.2.2 Second order settings

In this experiment, three different models are employed in the second order respectively, so that we can compare the performance.

- Mean model

$$\theta_o \approx \frac{1}{n-d+1} \sum_{i=d}^n \theta(t_i) \quad (14)$$

Here, we only consider the last $(n-d+1)$ terms of $\theta(t_i)$, because it is very random in the start.

- Max model

$$\theta_{ko} \approx \max \{ \theta_k(t_i) | d \leq i \leq n \} \quad (15)$$

Here, θ_{ko} and $\theta_k(t_i)$ are the k dimension of θ_o and $\theta(t_i)$ respectively. And we also only consider the last $(n-d+1)$ terms of $\theta(t_i)$, because it is very random in the start.

- CNN model

In this experiment, because of our experimental conditions restrict, we employ the simplest CNN that has only one 3×3 kernel and one convolutional layer. As shown in figure 2, taken the sequence $\{\theta(t_i)\}$ as input data, through a convolutional layer and a max pooling layer and a non-linear layer, θ_o is generalized. Specially, after the convolutional operation, a matrix with the same size to input layer is generalized. It means that there exists a danger of out-of-range. When it happens, we regard all out-of-range inputs as zero.

Fig. 2. CNN model employed in the second order.

3.3 Effect of HNN

As shown in figure 3 with the precise-recall curves, the HNN models significantly outperforms the BGRU+ATT model over the entire range of the recall. HNN_MAX and HNN_MEAN have the same performance, and the HNN_CNN is even better. In contrast to BGRU+ATT, HNN_CNN has an improvement of greater than 3% at each recall. On the level of low recall, the precise is impressive, while it drops quickly when the recall is greater. When the recall reaches 0.32 approximately, all the precise is less than 50%.

The figure 4 shows the result of the proposed model trained by half size of the data used in figure 3 model. It is shown that there is a greater difference between HNN_CNN and BGRU+ATT than the result shown in figure 3, which means the performance of HNN_CNN descend no worse than the performance of BGRU+ATT. That is to say that the overfitting problem is indeed released in a degree when HNN is employed.

3.4 Further explanation

When human is thinking about a question, they seldom achieve the best answer after his first thinking. However, when a man turns back and rethinks the same question, he will achieve great promotion. It is worth mentioning that the rethinking progress is always along with the original thinking orbit. The second order of the proposed model is equipped with this ability. When the finding the relationship of target entities, it is easy to be affected by various modifiers, for instance appositive and parenthesis, in the sentences. As a result, the model ends with a local optimal solution easily. The second order network makes an examination to each optimal step when the first order network being trained., which makes the whole model achieve a better result.

Fig.3. Performance comparison of each model trained with the whole dataset.

Fig.4. Performance comparison of each model trained with half of the dataset.

4 Conclusion and future works

In this paper, we develop a new neural network HNN to solve the task of relation extraction in NLP. The experiment result suggests that our model significantly outperforms the best CNN model at the task of relation extraction, which means the over fitting problem is released.

In the future, we will explore the following directions: There is a great time consuming in our model, though it achieves a better performance. In the future, we will optimize the structure and computing functions to release this problem. We believe that there is a great probability that higher order neural network can perform better. We will develop higher order neural network if we can discover a method releasing the time consuming problem.

References

1. Miwa M, Bansal M. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures[J]. (2016)
2. Ammar W, Peters M, Bhagavatula C, et al. The AI2 system at SemEval-2017 Task 10 (ScienceIE): semi-supervised end-to-end entity and relation extraction[C]// International Workshop on Semantic Evaluation. 2017:592-596.
3. Lin Y, Shen S, Liu Z, et al. Neural Relation Extraction with Selective Attention over Instances[C]// Meeting of the Association for Computational Linguistics. 2016:2124-2133.
4. Socher R, Huval B, Manning C D, et al. Semantic compositionality through recursive matrix-vector spaces[C]// Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. 2012:1201-1211.
5. Zeng D, Liu K, Lai S, et al. Relation classification via convolutional deep neural network[J]. (2014)
6. Asma Ben Abacha and Pierre Zweigenbaum. Automatic extraction of semantic relations between medical entities: a rule-based approach. J. Biomedical Semantics, 2(S-5): S4, (2011)
7. Hong, Gumwon. Relation Extraction Using Support Vector Machine[M]// Natural Language Processing – IJCNLP 2005. Springer Berlin Heidelberg, 2005:366-377.
8. Peng N, Poon H, Quirk C, et al. Cross-Sentence N-ary Relation Extraction with Graph LSTMs[J]. (2017)
9. Riedel, S., Yao, L., & McCallum, A. (2010, September). Modeling relations and their mentions without labeled text. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 148-163). Springer, Berlin, Heidelberg.
10. Hoffmann R, Zhang C, Ling X, et al. Knowledge-based weak supervision for information extraction of overlapping relations[C]// Meeting of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2011:541-550.
11. Surdeanu M, Tibshirani J, Nallapati R, et al. Multi-instance multi-label learning for relation extraction[C]// Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. 2012:455-465.
12. Finkel J R, Grenager T, Manning C. Incorporating non-local information into information extraction systems by Gibbs sampling[C]// Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2005:363-370.