Reasoning Over Relations Based on Chinese Knowledge Bases

Guoliang Ji, Yinghua Zhang, Hongwei Hao, and Jun Zhao

Institute of automation, Chinese academy of sciences
{jiguoliang2013,yinghua.zhang,hongwei.hao}@ia.ac.cn
{jzhao}@nlpr.ia.ac.cn

Abstract. Knowledge bases are useful resource for many applications, but reasoning new relationships between new entities based on them is difficult because they often lack the knowledge of new relations and entities. In this paper, we introduce the novel Neural Tensor Network (NTN)[1] model to reason new facts based on Chinese knowledge bases. We represent entities as an average of their constituting word or character vectors, which share the statistical strength between entities, such as 荔枝巢蛾 and 荔枝异形小卷蛾. The NTN model uses a tensor network to replace a standard neural layer, which strengthen the interaction of two entity vectors in a simple and efficient way. In experiments, we compare the NTN and several other models, the results show that all models' performance can be improved when word vectors are pre-trained from an unsupervised large corpora and character vectors don't have this advantage. The NTN model outperforms others and reachs high classification accuracy 91.1% and 89.6% when using pre-trained word vectors and random character vectors, respectively. Therefore, when Chinese word segmentation is a difficult task, initialization with random character vectors is a feasible choice.

Keywords: knowledge bases, reason, Neural Tensor Network, word representations, character representations

1 Introduction

With the advent of the era of big data, a lot of information stored in the form of structured data has been built in knowledge bases which are multi-relational graph data whose nodes represent entities and edges corresponds to relations [2]. Relations in knowledge bases are always be denote by triples in the form of (entity, relation, entity). Recently, knowledge bases are extremely useful resource for many nature natural language processing tasks such as coreference resolution, information retrieval, recommendation systems and social networks. However, the entities and relationships that exist in knowledge bases are always incompleteness for users due to various practical reasons. Hence, learning new facts based on the knowledge bases is a necessary way to improve them.

Much work(probability graph model and inductive logic programming and Markov Logic Network et al.) has focused on extending existing knowledge bases using pattern or classifiers applied to large text corpora[1]. However, not all knowledge is recorded by text, especially common sense which is obvious to people. For example, given the fact (荔枝巢蛾 门 节肢动物门), a person can infer the new fact (荔枝异形小卷蛾 门 节肢动物门) without needing to find any textual evidence. Therefore, learning new relations(triples) based on knowledge bases has been increasingly popular. Nickle et al.(2011 and 2013)[3, 4] introduced a tensor factorization method for learning on multi-relation data. Mukherjee et al.(2013)[2] used a matrix tri-factorization approach to extracting new facts in knowledge bases. Recently, Socher et al.(2013)[1] applied a neural tensor network to reasoing common sense, which is the base of our work.

In this paper, we introduce the neural tensor networks (NTN)[1] to accomplish common sense reasoning with the facts that exist in Chinese knowledge bases. This network model can accurately predict the additional relationships among entities that are not exist in knowledge bases. We represent every entity as a vector. For sharing statistical strength among the entities that contain similar substrings, we represent each entity as the average of its word or character vectors whose initial value are pre-trained with a neural network language model based on an unsupervised large scale corpora or are sampled from a zero mean Gaussian distribution. Each relation corresponds to a group of parameters of neural tensor networks. The entities and relationships can interact well through the neural tensor networks.

The main contribution of this paper is to apply the NTN model to reason new facts based on Chinese bases. The paper is organized as follows. Section 2 and section 3 introduce some related work and the Neural Tensor Network model , repectively. Section 4 discuss how to use the NTN model to reason on Chinese knowledge bases. Section 5 reports parameters' setting and results of experiments. Section 6 we summarize our contribute and consider the further work directions.

2 Related Work

This work mainly involves two areas of NLP research: semantic vector spaces and deep learning.

Semantic vector spaces Neural language models (Bengio et al.2003; Mnih and Hinon,2007; Collobert and Westion,2008; Schwenk and Gauvain,2002; Emami et al.,2003) have been shown to be very powerful at language modeling, a task where models are asked to accurately predict the next word given previously seen words[5]. By using distributed representations of words which model words' similarity, most of these models addresses the data sparseness problem that n-gram models encounter when large contexts are used. Collobert and Weston(2008) used a neural language model to compute scores $g(s)$ and $g(s^w)$ of the n-gram s and s^w, where s^w is s with the last word replaced by word w, and a ranking-loss training object to make g(s)to be larger than $g(s^w)$ by a margin of 1 for any other word w in the vocabulary. This model have showed increased performance and the word embedding produced by it have been used in many literatures. Huang and Socher(2013) added global information of documents to the neural language model of Collobert & Weston and produced multiple word prototypes. Our Chinese word and character vectors come from Huang and Socher's language model.

Deep learning The neural tensor network is related to several neural network models in deep learning literature. Ranzato and Hinton introduced a factored 3-way Restricted Boltzmann Machine which is also parameterized by a tensor[1]. D.Yu and L.Deng introduce a model with tensor layers for speech recognition[1]. Socher and Chen[1] introduced the neural tensor network for reasoning for knowledge bases completion, and they developed a recursive version of this model for sentiment analysis. Bowman[5] trained a recursive neural tensor networks model on a new corpus of constructed examples of logical reasoning in short sentences and the result is promising to capture logical reasoning. Nickle[3] introduced a tensor factorizaton method for multi-relational learning, where a knowledge base was regarded as a three dimensional tensor.

3 Neural Tensor Network Model

This section introduces the neural tensor network to reason relationships among entities by learning vector representations for them. As shown in Fig. 1(part I),each triple described as (e_1, R, e_2), where $e_i(i = 1, 2), R$ represent entities and relationship respectively, corresponds a tensor network whose inputs are $e_i(i = 1, 2)$. The model obtains a high confidence if they are in that relationship and a low one otherwise. The following are three sections: (i)model structure, (ii)training objective, (iii)classify relation triples.

3.1 Model Structure

In this model, we introduce the NTN model structure. First, we define a set of parameters indexed by R for each relation's scoring function and let e_1, e_2 be the vector representations of two entities. Then the Neural Tensor Network (NTN) computes the score of a given triple (e_1, R, e_2) through a bilinear tensor layer which relates two entity vectors across multiple dimensions.

$$score(e_1, R, e_2) = U_R^T f(e_1^T T_R^{[1:k]} e_2 + W_R \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + b_R) \tag{1}$$

where $f = tanh$ or $f = sigmoid$ that often be used in neural network models, $T_R^{[1:k]} \in R^{d\times d\times k}$ is a tensor and the bilinear product $e_1^T T_R^{[1:k]} e_2$ results in a vector $h \in R^k$, where each entry is computed by one slice $i = 1, \ldots, k$ of the tensor: $h_i = e_1^T T_R^{[1:k]} e_2$ [1]. The parameters $U_R \in R^k$, $W_R \in R^{k\times 2d}$ and $b_R \in R^k$ are the form of standard neural network. The main advantage of this model is that it can relate the two inputs multiplicatively, which incorporates the interaction of the two inputs efficiently.

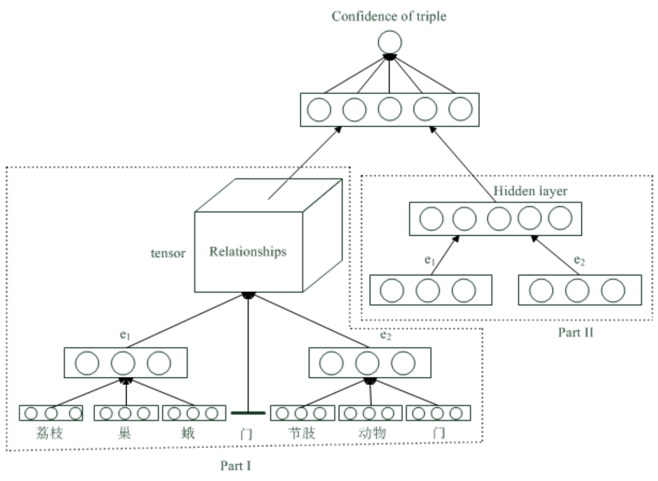

Fig. 1. Overview of the Neural Tensor Network model. Part(I) is a tensor network which enforce the interaction between two entities, part(II) is a standard neural network layer. We train the word vectors and predict whether a triple is true or not according to it's confidence.

3.2 Training Objective

In the training set, each triple denoted by $T^{(i)} = \left(e_1^i, R^{(i)}, e_2^i\right)$ should receive a higher confidence than a triple whose second entity is replaced with a random entity. Provided that there are N_R many different relations which are indexed by $R^{(i)}$ $(i = 1, \ldots, N_R)$. Each relation corresponds to a set of neural tensor net parameters. We call the triple with a random entity negative sample and represent it with $T_c^{(i)} = \left(e_1^{(i)}, R^{(i)}, e_c\right)$, ,where e_c was randomly sampled from the set of all entities. We denote the set of all relationships' NTN parameters by $\Omega = U, T, W, b, L$, where U, T, W, b are model parameters and L is word vectors. We minimize the following objective:

$$J = \sum_{i=1}^{N} \sum_{c=1}^{C} max\left(0, 1 - score\left(T^{(i)}\right) + score\left(T_c^{(i)}\right)\right) + \lambda \left\|\Omega\right\|_2^2 \qquad (2)$$

Where N is the total number of triples in the training set. We score the positive relation triple higher than its negative one up to a margin of 1[1]. In order to enhance the learning ability of the NTN model, for each positive triple we sample C random negative triples. We use the L_2 weight regularization for all neural tensor network parameters whose weighted hyper parameter is λ.

3.3 Classify relation triples

The goal is to predict the likely truth in the form of triples (e_1, R, e_2) according to their scores in the testing data. We should find a threshold t_R for every relation such that if $score\,(e_1, R, e_2) \geq t_R$, the triple is correct, otherwise it does not. Hence, we need to create a valid set to decide the thresholds and a testing set to evaluate the model.

4 Reasoning on Chinese Knowledge Bases

Before reasoning with Neural Tensor Network, we also need to do word segmentation for entities existing in Chinese knowledge bases. However, in our Chinese knowledge bases, entities are always phrases which are difficult to segment correctly. Therefore, we consider two kinds of Chinese word segmentation: word and character.

4.1 Word and Character Representations

We represent Chinese word and character as continuous vectors. We explore t-wo settings. In the first setting we simply initialize each word(character) vector $x \in R^n$ by sampling it from a zero mean Gaussian distribution: $x \sim N(0, \sigma^2)$. In the second setting, we pre-train the word (character) vectors with an unsupervised neural language model(Huang and Socher,2013). Fig. 2 shows the model. It makes use of local and global context to pre-train the word(character) vectors. These word(character) vectors are then stacked into a word embedding matrix $L \in R^{|V| \times n}$, where V represents the size of vocabulary and n is the dimension of vectors. These vectors will be revised to capture certain semantic during learning process.

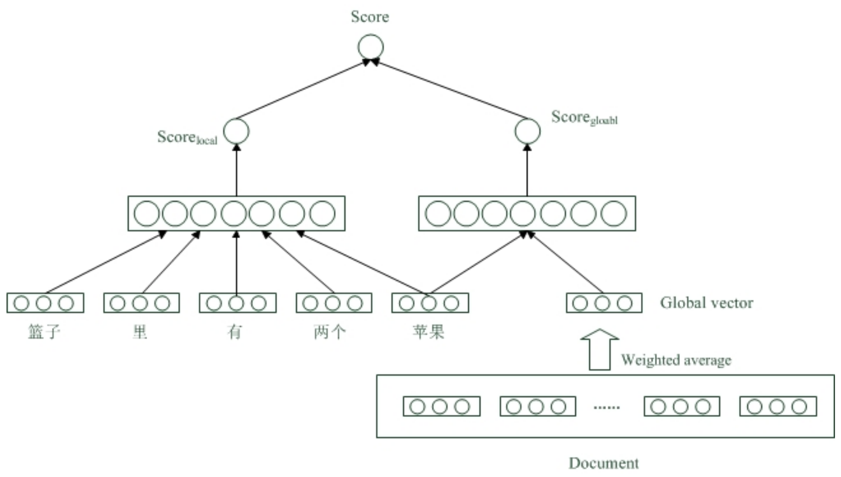

Fig. 2. An overview of the neural language model. The model makes use of local and global context to learn word representations based on large scale unsupervised corpora.

In both cases, each word(character) has an associated vocabulary index k into the embedding matrix which we use to retrieve the word's (character's) vector representation. Mathematically, if we want to get the *k-th* word's(character's)vector of the vocabulary, we only need to use a binary vector which is zero in all positions except at the *k-th* index,

$$x_i = b_k^T L \tag{3}$$

4.2 Entity Representation

Previous work[6–8] assigned a single vector representation to each entity of the knowledge base, which does not allow the sharing of statistical strength between the words describing each entity[1]. We represent each word(character) as a d-dimensional vector and compute an entity vector as the composition of its word(character) vectors. For example, entity 荔枝巢蛾 is consist of words 荔枝, 巢 and 蛾, or is consist of character 荔, 枝, 巢, and 蛾. In this task, we represent the entity vector by averaging its word or character vectors. For example,

$$v_{\text{荔枝巢蛾}} = \frac{1}{3}(v_{\text{荔枝}} + v_{\text{巢}} + v_{\text{蛾}}) \tag{4}$$

or

$$v_{\text{荔枝巢蛾}} = \frac{1}{4}(v_{\text{荔}} + v_{\text{枝}} + v_{\text{巢}} + v_{\text{蛾}}) \tag{5}$$

As word or character is the smallest unit, we train the model on the word's and character's level. The training err derivatives are also back-propagated to these vectors.

5 Experiments

Our goal is to predict whether a given triple holds, which can be seen as common sense reasoning or prediction in relationship networks[1]. For example, if 荔枝巢蛾 is in 动物界, it is also 荔枝异形小卷蛾. In this section, we first introduce the datasets and then show some experiments. We compare the NTN with other models and obtain several conclusions based on analyses of these results, such as whether to use word vectors or character vectors.

In our experiments, we cross validate using the valid set to find the best hyperparamters to get the highest accuracy. (i) vector initialization with a zero mean Gaussian distribution; (ii)$\lambda = 10^{-4}$; (iii) number of training iterations $T = 7000$; (iv) the dimensionality of word or character $d = 50$; (v) number of slices of tensor $k = 3$; (vi) number of neural nodes of hidden layer in single layer model $h = 200$.

We develop all code with Python and use theano to compute the derivatives of all parameters. Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently[10].It can be used for doing efficient symbolic differentiation, even for function with many inputs[10]. The NTN model is trained by taking derivatives with respect to all the above parameters. We use minibatched CG and L-BFGS algorithm for optimization which converges to a local optimum of our objective function.

5.1 Date Sets

Table 1. The statistics of datasets

#Dateset	#Relation	#Entity	#Train	#Valid	#Test
AniPla	12	12881	28110	7026	7030
Diet	8	23547	27000	6000	6000

Table.1 shows the statistics of the databases. In this task, we construct two datasets named AniPla and Diet sampled from Baidu Encyclopedia infobox. All data stored in the form of (e_1, R, e_2). In AniPla, the first entities are animal's and plant's names, the second entities are the qualities of animals and plants, for example,(龙猫 目 啮齿目) and (竹叶青 种 茶叶). In total, there are 12881 unique entities in 12 different relations. We use 28110 triples for training and 7030 triples for testing. The valid set has 7026 triples for compute the thresholds $t_R\,(R = 1, 2, \ldots, 12)$ for all relations. As Diet, the first entities are food's names, the second entities are the qualities of food, for example, (清蒸鱼 主要食材 鱼) and (香菇面 工艺 煮). In total, there are 23547 unique entities in 8 different relations. We use 27000 triples for training and 6000 triples for testing. The valid set has 6000 triples for compute the thresholds $t_R\,(R = 1, 2, \ldots, 8)$ for all relations. We do word segmentation for AniPla and Diet by manual handling and software, respectively.

5.2 Results

This section we compare the results of NTN and other models with different initialize methods of word(character) representations. The Tensor Factorization model[3] dosen't need to use word or character vectors. Table.2 and Table.3 show the resulting accuracy of each model.

Table 2. AniPla: Accuracy of models

Model	Wrod vectors(%)		Character vectors(%)	
	Random	Pre-trained	Random	Pre-trained
Single Layer model	80.5	84.6	85.2	82.1
Neural Tensor network	89.3	**91.1**	**89.6**	87.7
Tensor Factorization model	88.6			

Table 3. Diet: Accuracy of models

Model	Wrod vectors(%)		Character vectors(%)	
	Random	Pre-trained	Random	Pre-trained
Single Layer model	71.3	74.0	75.0	70.1
Neural Tensor network	73.3	**75.3**	**75.9**	73.2
Tensor Factorization model	75.1			

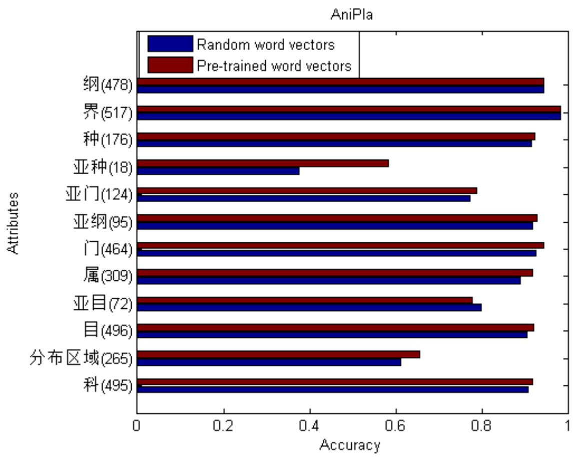

Fig. 3. Comparison of accuracy of diffierernt relations(AniPla). The number in the bracket represents the size of samples in the test set.

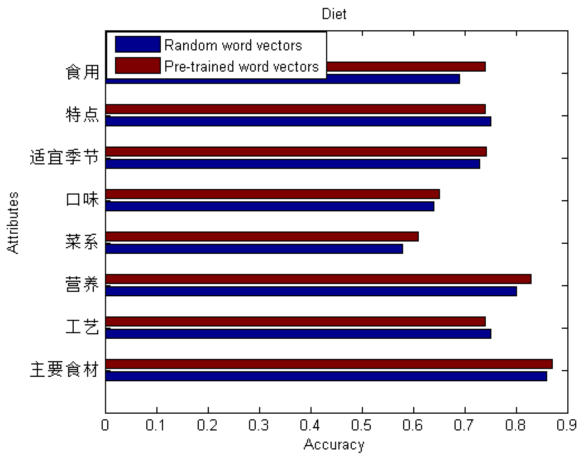

Fig. 4. Comparison of accuracy of differernt relations(Diet).

Table.2 shows that models obtain higher accuracy with pre-trained word vectors than with random vectors. In Table.3, pre-trained word vectors don't have any advantages than random vectors. We can conclude that software can't do word segmentation for entities perfectly.

First, we analyze the influence of word and character initializations. Table.2 reports the accuracy of each model. With unsupervised word vectors' initialization, the single layer and NTN models can reach high classification accuracy: 84.6% and 91.1% on the dataset respectively. However, with random initialization, they both obtain a lower accuracy. It shows that the models have improved accuracy with initialization from pre-trained word vectors. In Fig. 4, the red bar is longer than the blue bar among most relations that also confirms it. By contrast, pre-trained character vectors don't have significantly advantage than random vectors. This phenomenon shows that Chinese semantics are expressed through words, not characters.

We now compare the accuracy of different models. All the data in Table.2 and Table.3shows that the NTN model is more power than other models. Even with character vector and random initialization, the NTN model can achieve accuracy 89.6%, which shows that when Chinese word segmentation is a hard task (the entities in many Chinese datasets are always phrases which are difficult to segment), the NTN model with character vectors from random initialization is a feasible strategy.

Fig. 3 and Fig. 4 shows the accuracy among different relation types. Here we use the NTN model to evaluation. The accuracy varies among different relations. In AniPla, the accuracy rangs from 38.7% to 96.2% and the two most difficult relations are 分布区域 and 亚种. In Diet, the accuracy rangs from 61.0% to 87.1% and the two most difficult relations are 口味 and 菜系. According to Fig. 3 and Fig. 4, we can see the Diet is more difficult to reason than AniPla.

6 Conclusion

We introduce the Neural Tensor Network for common sense reasoning based on existing Chinese knowledge bases. This model constructs a tensor to enhance the interaction between entities in an efficient way and obtains a high prediction accuracy by training word vectors whose initial value come from a unsupervised large corpora. The future work is to improve these ideas to achieve higher accuracy and apply the NTN model to complete knowledge bases.

References

1. Socher, R., Danqi Chen, Manning, C.D.,Ng, Y.: Reasoning With Neural Tensor Networks for Knowledge Base Completion. In:*Advances in Neural Information Processing Systems 26* (2013)
2. Mukherjee, T., Pande, V., Kok, S.: Extracting New Facts in Knowledge Bases:-A matrix trifactorization approach. In: *ICML workshop on Structured Learning: Inferring Graphs from Structured and Unstructured Inputs* (2013)

3. Nickel, M., Tresp, V., Chen, Kriegel, H.P.: A Three-Way Model for Collective Learning on Multi-Relational Data. In:*Proceedings of the 28th International Conference on Machine Learning.* (2011)
4. Nickel, M., Tresp, V.: Logstic Tensor Factorization for Multi-Relational Data. In:*Proceedings of the 30th International Conference on Machine Learning.* (2013)
5. Huang, E.H., Socher, R., Manning, C.D., Ng, Y.:Improving Word Representations via Global Context and Multiple Word Prototypes. In:*Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012
6. Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning structured embeddings of knowledge bases. In:*AAAI.*(2011)
7. Jenatton, R., Le Roux, N., Bordes, A., Obozinski, G.: A latent factor model for highly multi-relational data. In:*NIPS.*(2012)
8. Bordes, A., Glorot, X., Weston, J., Bengio, Y.:Joint Learning of Words and Meaning Representations for Open-Text Semantic Parsing. In:*AISTATS.*(2012)
9. Bowman, S.R: Can recursive neural tensor networks learn logical reasoning? In:*International Conference on Learning Representations*, 2013
10. Deep Learning Tutorials, `http://deeplearning.net/tutorial/`
11. J. Bergstra, O., Breuleux, F., Bastien, P. Lamblin, R., Pascanu, G., Desjardins, J., Turian, D., Warde-Farley and Y., Bengio: Theano: A CPU and GPU Math Expression Compiler. In:*Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.
12. Socher, R., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
13. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
14. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
15. Deep Learning Tutorials, `http://deeplearning.net/tutorial/`