Context-Aware Tree-Based Convolutional Neural Networks for Natural Language Inference

Zhao Meng[1,2], Lili Mou[1,2], Ge Li[1,2(✉)], and Zhi Jin[1,2(✉)]

[1] Key Laboratory of High Confidence Software Technologies, Peking University, Ministry of Education, Beijing, China
doublepower.mou@gmail.com, {lige,zhijin}@sei.pku.edu.cn
[2] Software Institute, Peking University, Beijing, China
zhaomeng.pku@outlook.com

Abstract. Natural language inference (NLI) aims to judge the relation between a premise sentence and a hypothesis sentence. In this paper, we propose a context-aware tree-based convolutional neural network (TBCNN) to improve the performance of NLI. In our method, we utilize tree-based convolutional neural networks, which are proposed in our previous work, to capture the premise's and hypothesis's information. In this paper, to enhance our previous model, we summarize the premise's information in terms of both word level and convolution level by dynamic pooling and feed such information to the convolutional layer when we model the hypothesis. In this way, the tree-based convolutional sentence model is context-aware. Then we match the sentence vectors by heuristics including vector concatenation, element-wise difference/product so as to remain low computational complexity. Experiments show that the performance of our context-aware variant achieves better performance than individual TBCNNs.

Keywords: Context-awareness · Tree-based convolutional neural network · Natural language inference

1 Introduction

Natural language inference (NLI), also known as *recognizing textual entailment*, is an important task in natural language processing (NLP), and has profound impact on other NLP applications [1,2], including paraphrase detection [3], question answering [4], and automatic summarization [5]. Formally, NLI aims to judge the relation between two sentences, called a premise sentence and a hypothesis sentence, respectively. The objectives of NLI are `Entailment`, `Contradiction`, and `Neutral`, where `Entailment` means the hypothesis sentence can be entailed from the premise sentence, `Contradiction` means the two sentences are contradictory to each other, and `Neutral` means the two sentences are logically independent to each other [2]. Examples are illustrated in Table 1.

Traditional approaches to NLI mainly focus on feature engineering [4] and formal reasoning [6]. Recently, neural networks have become one of the mainstream

F. Lehner and N. Fteimi (Eds.): KSEM 2016, LNAI 9983, pp. 515–526, 2016.
DOI: 10.1007/978-3-319-47650-6_41

approaches to almost every NLP task, including natural language inference. Researchers have applied various neural models, e.g., recurrent neural networks (RNNs) [7,8], to capture sentence-level meanings; then heuristic matching or word-by-word attention mechanisms are used to classify the relation between the premise and the hypothesis. While attention mechanisms typically yield higher performance, they are more computationally intensive than heuristic matching in terms of complexity order: $\mathcal{O}(n^2)$ versus $\mathcal{O}(n)$, where n is the number of words in a sentence. Therefore, heuristic matching is still a hot research topic in the NLP community, especially when complexity is a major concern.

In our previous work [9], we apply a tree-based convolutional neural network (TBCNN) as the underlying sentence model and then match the premise and the hypothesis by heuristics like vector concatenation, element-wise product, and element-wise difference. Such approach has achieved state-of-the-art performance in the complexity of $\mathcal{O}(n)$, justifying the rationale of using TBCNN as the underlying sentence model.

However, the main shortcoming of this model is that the premise and the hypothesis are modeled independently, that is, when extracting features of the hypothesis by tree-based convolution, the model is unaware of the information of the premise. Evidence in the literature shows that context-awareness may be important in sentence pair modeling, which means that it is important to concern the information of the other sentence when we are modeling on one of the sentences: Rocktäschel et al. [8] propose a single-chain RNN that runs through both sentences, and achieve higher performance than two separate RNNs. This could be intuitively thought of as such that, in Table 1 for example, it is valuable to know the phrase *drinking orange juice* in the premise, when we model *drinking juice* in the hypothesis. Likewise, the phrase *An older man* provides a useful hint of the contradiction to *Two women*. Hence, we are curious whether such context-awareness can benefit our TBCNN model.

Table 1. Examples in the Stanford Natural Language Inference (SNLI) dataset. The classification objectives of NLI are `Entailment`, `Contradiction`, and `Neutral`. The `Neutral` class indicates two irrelevant sentences.

Premise	
An older man is drinking orange juice at a restaurant	
Hypothesis	Label
A man is drinking juice	Entailment
Two women are at a restaurant drinking wine	Contradiction
A man in a restaurant is waiting for his meal to arrive	Neutral

In this paper, we propose a context-aware tree-based convolutional neural network to improve the performance of NLI. Our idea is to summarize the premise's knowledge as fixed-size vectors, which are fed to the tree-based convolutional layer when we model the hypothesis. Then two sentences' information

is matched by heuristics as in [9]. In this way, the underlying sentence model is context-aware, but the overall complexity remains low, i.e., $\mathcal{O}(n)$, in contrast to word-by-word attention mechanisms. We conduct our experiments on a large open dataset, the Stanford Natural Language Inference (SNLI) Corpus [10], and achieve better performance than our previously published TBCNN model did.

2 Related Work

In past years, researchers have mainly focused on feature-based approaches to natural language inference. For example, Harabagiu et al. [4] use linguistic knowledge and lexical alignment to decide the extent to which a sentence can be entailed from another. Bos et al. [6], on the other hand, use formal reasoning by combining shallow (word overlap) and deep (semantic parsing) NLP methods. While reasoning approaches can search for a proof in logical forms for entailment recognition, their scope and accuracy are highly limited.

Recent advances in neural networks bring new methods to NLI, which can be viewed as a task of sentence pair modeling, that is, the goal is to determine the relation between a pair of sentences (the premise and the hypothesis). Typically, these approaches involve two steps: sentence modeling and matching.

2.1 Sentence Modeling

In this step, the goal is aimed at capturing the meaning of a sentence. Kalchbrenner et al. [11] and Kim [12] use convolutional neural networks (CNNs) to model sentences; in CNNs, a sliding window extracts features of neighboring words. Recurrent neural networks (RNNs) iteratively pick up words in a sentence by keeping one or a few hidden states [13]. Socher et al. [14] propose recursive neural networks—which utilize a tree structure and—by propagating information recursively from leaf nodes to the root to summarize a sentence as a vector. In our previous work [15], we propose a tree-based convolutional neural network (TBCNN), which combines the merits of CNNs and recursive nets: it is structure-sensitive as recursive nets and has short propagation paths like CNNs. We have achieved state-of-the-art performance in several sentence classification tasks with TBCNN, showing its effectiveness. Based on the above sentence models, many studies build sentence pair models upon RNNs [7,8], CNNs [16,17], etc.

2.2 Sentence Matching

To determine the relation between a pair of sentences, Zhang et al. [17] and Hu et al. [16] concatenate the vectors of each sentence; He et al. [18] use Euclidean distance, cosine, and element-wise absolute difference as features. Other researchers compute word-by-word similarity matrices [3,7].

Recently, Rocktäschel et al. [8] demonstrate that context-awareness is important in sentence matching. They propose several methods including single-chain

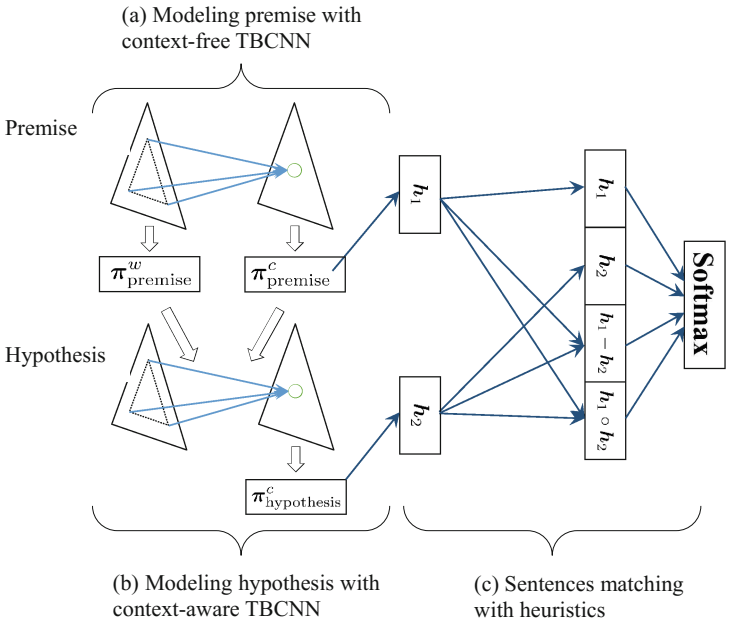

Fig. 1. The context-aware TBCNN model.

RNNs, static attention, and word-by-word attention, outperforming separate underlying sentence models. Wang et al. [19] further improve the performance by developing more elegant attention methods.

Although word-by-word attention and similarity matrices usually outperform simple matching heuristics, they are of higher overall complexity order, i.e., $\mathcal{O}(n^2)$. In this paper, we focus on $\mathcal{O}(n)$ methods: we enhance the TBCNN model with context-awareness, but remain in low complexity.

3 Our Approach

In this section, we describe our approach in detail. Subsection 3.1 provides an overview of our approach. Subsection 3.2 introduces the context-free tree-based convolutional neural network (TBCNN), which serves as the base model. We propose the context-aware TBCNN variant in Subsect. 3.3. Then we present matching heuristics and the training objective in Subsects. 3.4 and 3.5, respectively.

3.1 Overview

Figure 1 depicts the overview of our model. Concretely, our model has three main components:

– First, we apply TBCNN to capture the meaning of the premise (Fig. 1a). This part is essentially the same as the original context-free (i.e., individual) TBCNN model in [9].
– Then, we design another TBCNN to model the hypothesis (Fig. 1b). Contrary to previous work, the tree-based convolution here is aware of the premise's information. This is accomplished by summarizing the premise as fixed-size vectors, which are fed to the convolutional layer to interact with the hypothesis.
– After sentence modeling, we match the two sentences' vectors by heuristics including concatenation, element-wise product and difference. Finally, we use a softmax layer for classification. (See Fig. 1c.)

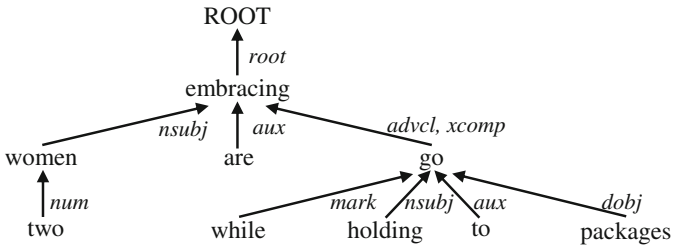

Fig. 2. Dependency parse tree

3.2 Tree-Based Convolution

The tree-based convolutional neural network (TBCNN) is proposed to model the parse trees of both programming languages [20] and natural languages [15]. In this part, we elaborate the process of tree-based convolution without context awareness, which is used to model the premise sentence in the NLI task.

First, we apply the Stanford parser[1] [21] to convert a sentence to a dependency tree, illustrated in Fig. 2. In our notations, an edge $a \xrightarrow{r} b$ refers to a being governed by b with the dependency type r. In total, we have approximately 30 different types (e.g., `nsubj`, `dobj`); other rare relations are mapped to a special type `default` in our study.

Then we perform the tree-based convolution over the dependency tree. We use pretrained word embeddings [22] as input signals. A subtree-based sliding window extracts structural features as the output of the convolution (denoted as $\boldsymbol{y}$). Formally, given a parent node p and its child nodes $c_1, \cdots, c_n$, we have

$$\boldsymbol{y} = f\left(W_p \boldsymbol{p} + \sum_{i=1}^{n} W_{r[c_i]} \boldsymbol{c} + \boldsymbol{b}\right) \tag{1}$$

[1] http://nlp.stanford.edu/software/lex-parser.shtml.

as detected features of the tree-based convolution at a certain position. Here, bold letters $\boldsymbol{p}$ and $\boldsymbol{c}_i$ refer to word embeddings of corresponding nodes; $\boldsymbol{b}$ is the bias vector. f is a nonlinear function; we used rectified linear units (ReLU) in our experiments, given by

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Notice that we have the same number of detected features $\boldsymbol{y}$'s and words in the original sentence, and that the number varies in different data samples. Hence, we apply dynamic pooling to summarize the features extracted by convolution. Specifically, a max-pooling operator takes the maximum value in each dimension of all features along the dependency tree. Suppose the tree-based convolutional layer extracts m features of n_c dimensions, the pooling layer outputs a vector $\boldsymbol{\pi}^c_{\text{premise}}$, its j-th dimension being

$$\boldsymbol{\pi}^c_{\text{premise}}[j] = \max\left\{\boldsymbol{y}_1[j], \boldsymbol{y}_2[j], \cdots, \boldsymbol{y}_m[j]\right\},\ 1 \leqslant j \leqslant n_c \tag{3}$$

The superscript c indicates that the features are pooled from those after convolution; the subscript suggests the features correspond to the premise. In all, $\boldsymbol{\pi}_{\text{premise}}$ provides summarized information of the premise and is used for sentence matching, described in Subsect. 3.4.

3.3 Context-Awareness for Tree-Based Convolution

As has been discussed in previous sections, context-awareness is important to sentence pair modeling, including the task of natural language inference. In this subsection, we propose a context-aware TBCNN variant to model the hypothesis sentence, that is to say, when we extract features of the hypothesis by tree-based convolution, we are equipped with the information of the premise sentence so that the second sentence model may focus on more relevant information rather than merely extract generic features of the sentence.

Concretely, we apply dynamic pooling to summarize the knowledge of the premise as fixed-size vectors and feed the vectors to the tree-based convolutional layer when modeling the hypothesis. More specially, we leverage two sets of features, which are of different abstraction levels, from the premise sentence:

- **Convolution-level features.** In Subsect. 3.2, we have summarized the premise's information as a vector $\boldsymbol{\pi}^c_{\text{premise}}$. It is natural to use such knowledge in the context-aware TBCNN model.
- **Word-level features.** In addition to the above convolution-level features, we also pool the raw word embeddings in the premise sentence as a vector, given by

$$\boldsymbol{\pi}^w_{\text{premise}}[j] = \max\left\{\boldsymbol{w}_1[j], \boldsymbol{w}_2[j], \cdots, \boldsymbol{w}_m[j]\right\},\ 1 \leqslant j \leqslant n_e \tag{4}$$

  Similar to Eq. (3), we have m vectors of word embeddings $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m$, and n_e denotes embeddings' dimension. The superscript w indicates that this set of features is of word level.

After obtaining the convolution-level and word-level features, $\boldsymbol{\pi}^c_{\text{premise}}$ and $\boldsymbol{\pi}^w_{\text{premise}}$, respectively, we feed them to the tree-based convolutional layer where we extract the hypothesis's features. We modify the convolution formula (1) as follows.

$$\boldsymbol{y} = f\left(W_p\boldsymbol{p} + \sum_{i=1}^{n} W_{r[c_i]}\boldsymbol{c}_i + W^c\boldsymbol{\pi}^c_{\text{premise}} + W^w\boldsymbol{\pi}^w_{\text{premise}} + \boldsymbol{b}\right) \tag{5}$$

In the above equation, W_p, $W_{r[c_i]}$, and $\boldsymbol{b}$ are the same weights and bias as in Eq. (1) because the tree-based convolution operator detects general structural information of sentences. However, during the interaction, we also provide feature detectors (convolution operators) with the premise's information $\boldsymbol{\pi}^c_{\text{premise}}$ and $\boldsymbol{\pi}^w_{\text{premise}}$, linearly transformed by the weight matrices W^c and W^w, respectively.

After the context-aware convolution process, we obtain a set of features over the dependency tree of the hypothesis sentence. Then we use max pooling to summarize them as a vector $\boldsymbol{\pi}^c_{\text{hypothesis}}$ in a same way as Eq. (3). (Details are not repeated here.) The vector $\boldsymbol{\pi}^c_{\text{hypothesis}}$, along with that of the premise sentence $\boldsymbol{\pi}^c_{\text{premise}}$, is used for sentence matching, as will be described in the next subsection.

3.4 Matching Heuristics

Before matching the two sentences' vectors, we transform them by a fully-connected hidden layer

$$\boldsymbol{h}_{\text{premise}} = f\left(W_h\boldsymbol{\pi}^c_{\text{premise}} + \boldsymbol{b}_h\right) \tag{6}$$

$$\boldsymbol{h}_{\text{hypothesis}} = f\left(W_h\boldsymbol{\pi}^c_{\text{hypothesis}} + \boldsymbol{b}_h\right) \tag{7}$$

This hidden layer is designed empirically and not the main point of this paper.

Then we apply several heuristics proposed in our previous work [9] to match the premise and the hypothesis:

– Concatenation of the two sentence vectors:

$$concat = [\boldsymbol{h}_{\text{premise}};\ \boldsymbol{h}_{\text{hypothesis}}]$$

– Element-wise difference:

$$diff = [\boldsymbol{h}_{\text{premise}} - \boldsymbol{h}_{\text{hypothesis}}]$$

– Element-wise product:

$$prod = [\boldsymbol{h}_{\text{premise}} \circ \boldsymbol{h}_{\text{hypothesis}}]$$

Then, they are further concatenated as the final features $\boldsymbol{m}$

$$\boldsymbol{m} = [concat; diff; prod]$$

which are fed to the softmax output layer. (In the above equations, semicolons refer to vector concatenation.)

In this way, we manage to integrate the premise's information to the hypothesis, but remain a low overall complexity of $\mathcal{O}(n)$. The computational complexity lies in the underlying (both context-free and context-aware) sentence models.

3.5 Training Objective

Finally, we feed the sentence matching vector $\boldsymbol{m}$ to a softmax layer as the output. We use standard cross-entropy loss as our cost function. Let m be the number of data samples in the training set and n_l be the number of labels. Suppose further $\boldsymbol{t}^{(i)}$ is the one-hot ground truth and $\boldsymbol{y}^{(i)}$ is the output of the softmax layer for the i-th data sample. The j-th element in $\boldsymbol{t}^{(i)}$ is on $(= 1)$ if the sample belongs to the j-th class. The training objective is

$$J = -\sum_{i=1}^{m}\sum_{j=1}^{n_l} t_j^{(i)} \log\left(y_j^{(i)}\right)$$

The network is trained by mini-batched stochastic gradient descent with backpropagation and regularized by dropout.

4 Evaluation

In this section, we describe the dataset of our experiment in Subsect. 4.1. We present our hyperparameters and settings in Subsect. 4.2. We compare our model with other models and analyze different context-aware TBCNN variants in detail in Subsect. 4.3.

Table 2. Statistics of the `SNLI` dataset.

Train	Validation	Test
550,152	10,000	10,000

4.1 Dataset

We use the Stanford Natural Language Inference (`SNLI`)[2] [10] to evaluate our context-aware TBCNN model. `SNLI` is a large dataset of more than 550k samples. All samples in `SNLI` are human-written sentences and are labeled manually. As illustrated in Table 1, `SNLI` has three categories of labels: `Entailment`, `Contradiction`, and `Neural`. `Entailment` means the hypothesis can be inferred

[2] http://nlp.stanford.edu/projects/snli/.

from the premise, while `Contradiction` means the two sentences have contradictory meanings. `Neural`, however, indicates that the premise and the hypothesis are irrelevant to each other. The labels are roughly equal-distributed in the dataset. We apply the official split for train/validation/test, which is listed in Table 2.

4.2 Experimental Settings

In this subsection, we details the experimental settings for our context-aware TBCNN. All layers including the word embeddings are 300-dimensional. Embeddings are pretrained on the Wikipedia corpus and fine-tuned during training. We use mini-batch stochastic gradient descent and set the mini-batch size to 50. The above values are chosen empirically mainly following [9]. We tune the following hyperparameters on the validation test: learning rate is chosen from {3, 1, 0.3, 0.1}. Power decay of learning rate is chosen from {1x, 0.9x, 0.3x}, which is the residual of learning rate after one epoch; intuitively, they can be thought of as no, slow, or fast decay. We do not add ℓ_2 penalty for convenience and simplicity. Instead, we use dropout [23] to regularize our model. The dropout rate is chosen from {0, 0.1, 0.2, 0.3, 0.4}. For efficiency of hyperparameter tuning, we do not conduct meaningless settings (e.g., a larger dropout rate when the model has ready been underfitting). Our context-aware TBCNN model reaches its peak performance when the learning rate is 0.3, the power decay is 0.9, and the dropout rate is 0.3. In the following part of our paper, we report the test accuracy that corresponds to the highest performance on the validation test.

In order to have a better understanding of the role of convolution-level features and word-level features when we model the hypothesis, we have an additional variant of context-aware TBCNN, where we only leverage the convolution-level features. That is to say, we only feed $\boldsymbol{\pi}^c_{premise}$ to the tree-based convolutional layer when modeling on the hypothesis. Word-level features are simply ignored in this variant. Thus, the output vector of the hypothesis's convolutional layer is:

$$\boldsymbol{y} = f\left(W_p\boldsymbol{p} + \sum_{i=1}^{n} W_{r[c_i]}\boldsymbol{c}_i + W^c\boldsymbol{\pi}^c_{\text{premise}} + \boldsymbol{b}\right) \tag{8}$$

The above equation is different from Eq. (5) in that we do not have the $W^w\boldsymbol{\pi}^w_{\text{premise}}$ term. By contrast, the full context-aware TBCNN proposed in Sect. 3 has two levels of abstraction of the premise.

4.3 Performance

We present the performance of our models in comparison with previously published results in Table 3. As we can see, our full context-aware TBCNN model outperforms the competing approaches which are of $\mathcal{O}(n)$ overall complexity,

Table 3. Context-awared TBCNN compared with other models.

Overall complexity	Model	Test accuracy (%)
$\mathcal{O}(n)$	Unlexicallized features [10]	50.4
	Lexicallized features [10]	78.2
	Vector sum + MLP [10]	75.3
	Vanilla RNN + MLP [10]	72.2
	LSTM RNN + MLP [10]	77.6
	CNN + cat [9]	77.0
	GRU w/skip-thought pretraining [24]	81.4
	Single-chain LSTM RNN [8] + two-way attention [8]	81.4
		82.4
	Non-context-aware TBCNN [9]	82.1
	Full context-aware TBCNN	**82.7**
$\mathcal{O}(n^2)$	LSTM + word-by-word attention [8]	83.5
	mLSTM [19]	86.1

Table 4. Test accuracies of TBCNN variants with different levels of context awareness.

Variants of model	Test accuracy (%)
TBCNN w/o context information [9]	82.1
TBCNN w/$\boldsymbol{\pi}^{c}_{\text{premise}}$	82.5
TBCNN w/$\boldsymbol{\pi}^{c}_{\text{premise}}$ and $\boldsymbol{\pi}^{w}_{\text{premise}}$	82.7

including two feature-based models (either unlexicalized or lexicalized), and several neural network-based models including RNNs and CNNs. Moreover, the proposed context-aware TBCNN model also outperforms the previous context-free TBCNN variant.

We compare TBCNN models of different levels of context awareness in Table 4 so as to have an in-depth analysis of context-awareness. If we only feed the premise's convolution-level features to the TBCNN model of hypothesis, we have an accuracy improvement of 0.4 %. This provides consistent evidence of the effectiveness of context-awareness. By furthering feeding the word-level features, we improve the model by another 0.2 %, indicating that more awareness of the premise results in higher performance.

We have to concede that our context-aware TBCNN model does not outperform LSTM models with intensive attention mechanisms. However, our complexity is lower than those in order, which is important is retrieval-and-reranking systems [25]. Our result is even comparable to one word-by-word attention model with LSTM-RNNs [8], showing the high performance of our model. Experiments show that context-awareness does improve TBCNN models in the NLI task. Intuitively, context-awareness helps the model find some relevant parts of the premise and the hypothesis when the model is judging the relation between the two sentences. For example, in Table 1, the model is aware of *a man* and *drinking*

orange juice of the premise sentence when modeling on the hypothesis, which contains the information of *two women* and *drinking wine*. Hence it is easier for the model to judge that the two sentences are contradictory to each other, or, the hypothesis sentence does not logically follow the premise sentence [2].

5 Conclusion

In this paper, we proposed a context-aware TBCNN model for NLI. The model can leverage different levels of abstraction from the premise when modeling on the hypothesis. Such abstraction includes convolution-level features and word-level features. Our experiments have shown that context-awareness is helpful when applied on TBCNN model. Moreover, the overall complexity of our context-aware TBCNN model remains low despite the newly added mechanism of context-awareness.

Acknowledgments. We would like to thank anonymous reviewers for insightful comments. This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201 and the National Natural Science Foundation of China under Grant Nos. 61232015, 91318301, 61421091, and 61502014.

References

1. MacCartney, B.: Natural language inference. Ph.D. thesis, Stanford University (2009)
2. Bowman, S.R.: Modeling natural language semantics in learned representations. Ph.D. thesis, Stanford University (2016)
3. Socher, R., Huang, E.H., Pennin, J., Manning, C.D., Ng, A.Y.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Advances in Neural Information Processing Systems, pp. 801–809 (2011)
4. Harabagiu, S., Hickl, A.: Methods for using textual entailment in open-domain question answering. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, pp. 905–912 (2006)
5. Harabagiu, S., Hickl, A., Lacatusu, F.: Negation, contrast and contradiction in text processing. In: Proceedings of the 20th AAAI Conference on Artificial Intelligence, pp. 755–762 (2006)
6. Bos, J., Markert, K.: Combining shallow and deep nlp methods for recognizing textual entailment. In: Proceedings of the First PASCAL Challenges Workshop on Recognising Textual Entailment, Southampton, UK, pp. 65–68 (2005)
7. Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L., Cheng, X.: A deep architecture for semantic matching with multiple positional sentence representations. arXiv preprint arXiv:1511.08277 (2015)
8. Rocktäschel, T., Grefenstette, E., Hermann, K.M., Kočiskỳ, T., Blunsom, P.: Reasoning about entailment with neural attention. In: Proceedings of the International Conference on Learning Representations (2015)
9. Mou, L., Men, R., Li, G., Xu, Y., Zhang, L., Yan, R., Jin, Z.: Natural language inference by tree-based convolution and heuristic matching. In: Proceedings of the 54th Annual Meeting of Association for Computational Linguistics (2016)

10. Bowman, S.R., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (2015)
11. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
12. Yin, W., Schütze, H.: Convolutional neural network for paraphrase identification. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 901–911 (2015)
13. Xu, Y., Mou, L., Li, G., Chen, Y., Peng, H., Jin, Z.: Classifying relations via long short term memory networks along shortest dependency paths. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1785–1794 (2015)
14. Socher, R., Perelygin, A., Wu, J.Y., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642 (2013)
15. Mou, L., Peng, H., Li, G., Xu, Y., Zhang, L., Jin, Z.: Discriminative neural sentence modeling by tree-based convolution. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 2315–2325 (2015)
16. Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: Advances in Neural Information Processing Systems, pp. 2042–2050 (2014)
17. Zhang, B., Su, J., Xiong, D., Lu, Y., Duan, H., Yao, J.: Shallow convolutional neural network for implicit discourse relation recognition. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 2230–2235 (2015)
18. He, H., Gimpel, K., Lin, J.: Multi-perspective sentence similarity modeling with convolutional neural networks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1576–1586 (2015)
19. Wang, S., Jiang, J.: Learning natural language inference with LSTM. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1442–1451 (2016)
20. Mou, L., Li, G., Zhang, L., Wang, T., Jin, Z.: Convolutional neural networks over tree structures for programming language processing. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence (2016)
21. de Marneffe, M.C., MacCartney, B., Manning, C.D.: Generating typed dependency parses from phrase structure parses. In: Proceedings of the Language Resource and Evaluation Conference, pp. 449–454 (2006)
22. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
23. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
24. Vendrov, I., Kiros, R., Fidler, S., Urtasun, R.: Order-embeddings of images and language. arXiv preprint arXiv:1511.06361 (2015)
25. Yan, R., Song, Y., Wu, H.: Learning to respond with deep neural networks for retrieval based human-computer conversation system. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (2016)