Natural Language Inference by Tree-Based Convolution and Heuristic Matching

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, Zhi Jin

1Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, China; Software Institute, Peking University, China
{doublepower.mou, menruimr}@gmail.com
{lige, xuyan14, zhanglu, zhijin}@sei.pku.edu.cn
2Baidu Inc., China yanrui02@baidu.com

Abstract

In this paper, we propose the TBCNN-pair model to recognize entailment and contradiction between two sentences. In our model, a tree-based convolutional neural network (TBCNN) captures sentence-level semantics; then heuristic matching layers like concatenation, element-wise product/difference combine the information in individual sentences. Experimental results show that our model outperforms existing sentence encoding-based approaches by a large margin.

1 Introduction

Recognizing entailment and contradiction between two sentences (called a premise and a hypothesis) is known as natural language inference (NLI) in MacCartney (2009). Provided with a premise sentence, the task is to judge whether the hypothesis can be inferred (entailment), or the hypothesis cannot be true (contradiction). Several examples are illustrated in Table 1.

NLI is in the core of natural language understanding and has wide applications in NLP, e.g., question answering (Harabagiu and Hickl, 2006) and automatic summarization (Lacatusu et al., 2006; Yan et al., 2011a; Yan et al., 2011b). Moreover, NLI is also related to other tasks of sentence pair modeling, including paraphrase detection (Hu et al., 2014), relation recognition of discourse units (Liu et al., 2016), etc.

Traditional approaches to NLI mainly fall into two groups: feature-rich models and formal reasoning methods. Feature-based approaches typically leverage machine learning models, but require intensive human engineering to represent lexical and syntactic information in two sentences

<table>
<thead>
<tr>
<th>Premise</th>
<th>Two men on bicycles competing in a race.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypothesis</td>
<td>People are riding bikes. E</td>
</tr>
<tr>
<td></td>
<td>Men are riding bicycles on the streets. C</td>
</tr>
<tr>
<td></td>
<td>A few people are catching fish. N</td>
</tr>
</tbody>
</table>

Table 1: Examples of relations between a premise and a hypothesis: Entailment, Contradiction, and Neutral (irrelevant).

(MacCartney et al., 2006; Harabagiu et al., 2006). Formal reasoning, on the other hand, converts a sentence into a formal logical representation and uses interpreters to search for a proof. However, such approaches are limited in terms of scope and accuracy (Bos and Markert, 2005).

The renewed prosperity of neural networks has made significant achievements in various NLP applications, including individual sentence modeling (Kalchbrenner et al., 2014; Mou et al., 2015) as well as sentence matching (Hu et al., 2014; Yin and Schütze, 2015). A typical neural architecture to model sentence pairs is the “Siamese” structure (Bromley et al., 1993), which involves an underlying sentence model and a matching layer to determine the relationship between two sentences. Prevailing sentence models include convolutional networks (Kalchbrenner et al., 2014) and recurrent/recursive networks (Socher et al., 2011b). Although they have achieved high performance, they may either fail to fully make use of the syntactical information in sentences or be difficult to train due to the long propagation path. Recently, we propose a novel tree-based convolutional neural network (TBCNN) to alleviate the aforementioned problems and have achieved higher performance in two sentence classification tasks (Mou et al., 2015). However, it is less clear whether TBCNN can be harnessed to model sentence pairs for implicit logical inference, as is in the NLI task.

In this paper, we propose the TBCNN-pair neural model to recognize entailment and contradiction between two sentences. We lever-
age our newly proposed TBCNN model to capture structural information in sentences, which is important to NLI. For example, the phrase “riding bicycles on the streets” in Table 1 can be well recognized by TBCNN via the dependency relations dobj(riding,bicycles) and prep_on(riding,street). As we can see, TBCNN is more robust than sequential convolution in terms of word order distortion, which may be introduced by determiners, modifiers, etc. A pooling layer then aggregates information along the tree, serving as a way of semantic compositionality. Finally, two sentences’ information is combined by several heuristic matching layers, including concatenation, element-wise product and difference; they are effective in capturing relationships between two sentences, but remain low complexity.

To sum up, the main contributions of this paper are two-fold: (1) We are the first to introduce tree-based convolution to sentence pair modeling tasks like NLI; (2) Leveraging additional heuristics further improves the accuracy while remaining low complexity, outperforming existing sentence encoding-based approaches to a large extent, including feature-rich methods and long short term memory (LSTM)-based recurrent networks.\footnote{Code is released on: \url{https://sites.google.com/site/tbcnninference/}}

## 2 Related Work

Entailment recognition can be viewed as a task of sentence pair modeling. Most neural networks in this field involve a sentence-level model, followed by one or a few matching layers. They are sometimes called “Siamese” architectures (Bromley et al., 1993).

Hu et al. (2014) and Yin and Schütze (2015) apply convolutional neural networks (CNNs) as the individual sentence model, where a set of feature detectors over successive words are designed to extract local features. Wan et al. (2015) build sentence pair models upon recurrent neural networks (RNNs) to iteratively integrate information along a sentence. Socher et al. (2011a) dynamically construct tree structures (analogous to parse trees) by recursive autoencoders to detect paraphrase between two sentences. As shown, inherent structural information in sentences is oftentimes important to natural language understanding.

The simplest approach to match two sentences, perhaps, is to concatenate their vector representations (Zhang et al., 2015; Hu et al., 2014, Arc-I). Concatenation is also applied in our previous work of matching the subject and object in relation classification (Xu et al., 2015; Xu et al., 2016). He et al. (2015) apply additional heuristics, namely Euclidean distance, cosine measure, and element-wise absolute difference. The above methods operate on a fixed-size vector representation of a sentence, categorized as sentence encoding-based approaches. Thus the matching complexity is $O(1)$, i.e., independent of the sentence length. Word-by-word similarity matrices are introduced to enhance interaction. To obtain the similarity matrix, Hu et al. (2014) (Arc-II) concatenate two words’ vectors (after convolution), Socher et al. (2011a) compute Euclidean distance, and Wan et al. (2015) apply tensor product. In this way, the complexity is of $O(n^2)$, where $n$ is the length of a sentence; hence similarity matrices are difficult to scale and less efficient for large datasets.

Recently, Rocktäschel et al. (2016) introduce several context-aware methods for sentence matching. They report that RNNs over a single chain of two sentences are more informative than separate RNNs; a static attention over the first sentence is also useful when modeling the second one. Such context-awareness interweaves the sentence modeling and matching steps. In some scenarios like sentence pair re-ranking (Yan et al., 2016), it is not feasible to pre-calculate the vector representations of sentences, so the matching complexity is of $O(n)$. Rocktäschel et al. (2016) further develop a word-by-word attention mechanism and obtain a higher accuracy with a complexity order of $O(n^2)$.

## 3 Our Approach

We follow the “Siamese” architecture (like most work in Section 2) and adopt a two-step strategy to classify the relation between two sentences. Concretely, our model comprises two parts:

- A tree-based convolutional neural network models each individual sentence (Figure 1a). Notice that, the two sentences, premise and hypothesis, share a same TBCNN model (with same parameters), because this part aims to capture general semantics of sentences.
- A matching layer combines two sentences’ information by heuristics (Figure 1b). After individual sentence models, we design a sentence matching layer to aggregate information. We use simple heuristics, including concate-
3.1 Tree-Based Convolution

The tree-based convolutional neural network (TBCNN) is first proposed in our previous work (Mou et al., 2016)\(^2\) to classify program source code; later, we further propose TBCNN variants to model sentences (Mou et al., 2015). This subsection details the tree-based convolution process.

The basic idea of TBCNN is to design a set of subtree feature detectors sliding over the parse tree of a sentence; either a constituency tree or a dependency tree applies. In this paper, we prefer the dependency tree-based convolution for its efficiency and compact expressiveness.

Concretely, a sentence is first converted to a dependency parse tree.\(^3\) Each node in the dependency tree corresponds to a word in the sentence; an edge \(a \rightarrow b\) indicates \(a\) is governed by \(b\). Edges are labeled with grammatical relations (e.g., \text{nsubj}) between the parent node and its children (de Marneffe et al., 2006). Words are represented by pretrained vector representations, also known as word embeddings (Mikolov et al., 2013a).

Now, we consider a set of two-layer subtree feature detectors sliding over the dependency tree. At a position where the parent node is \(p\) with child nodes \(c_1, \cdots, c_n\), the output of the feature detector, \(y\), is

\[
y = f \left( W_p p + \sum_{i=1}^{n} W_r[c_i] c_i + b \right)
\]

Let us assume word embeddings \((p \text{ and } c_i)\) are of \(n_e\) dimensions; that the convolutional layer \(y\) is \(n_e\)-dimensional. \(W \in \mathbb{R}^{n_e \times n_e}\) is the weight matrix; \(b \in \mathbb{R}^{n_e}\) is the bias vector. \(r[c_i]\) denotes the dependency relation between \(p\) and \(c_i\). \(f\) is the non-linear activation function, and we apply ReLU in our experiments.

After tree-based convolution, we obtain a set of feature maps, which are one-one corresponding to original words in the sentence. Therefore, they may vary in size and length. A dynamic pooling layer is applied to aggregate information along different parts of the tree, serving as a way of \textit{semantic compositionality} (Hu et al., 2014). We use the max pooling operation, which takes the maximum value in each dimension.

Then we add a fully-connected hidden layer to further mix the information in a sentence. The obtained vector representation of a sentence is denoted as \(h\) (also called a \textit{sentence embedding}). Notice that the same tree-based convolution applies to both the premise and hypothesis.

Tree-based convolution along with pooling enables structural features to reach the output layer with short propagation paths, as opposed to the recursive network (Socher et al., 2011b), which is also structure-sensitive but may suffer from the problem of long propagation path. By contrast, TBCNN is effective and efficient in learning such structural information (Mou et al., 2015).

3.2 Matching Heuristics

In this part, we introduce how vector representations of individual sentences are combined to capture the relation between the premise and hypothesis. As the dataset is large, we prefer \(O(1)\) matching operations because of efficiency concerns. Concretely, we have three matching heuristics:

- Concatenation of the two sentence vectors,
- Element-wise product, and
- Element-wise difference.

The first heuristic follows the most standard procedure of the “Siamese” architectures, while the latter two are certain measures of “similarity” or

---

\(^2\)Preprinted on arXiv on September 2014
(http://arxiv.org/abs/1409.5718v1)

\(^3\)Parsed by the Stanford parser
(http://nlp.stanford.edu/software/lex-parser.shtml)
“closeness.” These matching layers are further concatenated (Figure 1b), given by

\[ m = [h_1; h_2; h_1 - h_2; h_1 \circ h_2] \]

where \( h_1 \in \mathbb{R}^{n_c} \) and \( h_2 \in \mathbb{R}^{n_c} \) are the sentence vectors of the premise and hypothesis, respectively; \( \circ \) denotes element-wise product; semicolons refer to column vector concatenation. \( m \in \mathbb{R}^{4n_c} \) is the output of the matching layer.

We would like to point out that, with subsequent linear transformation, element-wise difference is a special case of concatenation. If we assume the subsequent transformation takes the form of \( W[h_1, h_2]^\top \), where \( W = [W_1, W_2] \) is the weights for concatenated sentence representations, then element-wise difference can be viewed as such as \( W_0(h_1 - h_2) = [W_0 - W_0][h_1, h_2]^\top \). (\( W_0 \) is the weights corresponding to element-wise difference.) Thus, our third heuristic can be absorbed into the first one in terms of model capacity. However, as will be shown in the experiment, explicitly specifying this heuristic significantly improves the performance, indicating that optimization differs, despite the same model capacity. Moreover, word embedding studies show that linear offset of vectors can capture relationships between two words (Mikolov et al., 2013b), but it has not been exploited in sentence-pair relation recognition. Although element-wise distance is used to detect paraphrase in He et al. (2015), it mainly reflects “similarity” information. Our study verifies that vector offset is useful in capturing generic sentence relationships, akin to the word analogy task.

4 Evaluation

4.1 Dataset

To evaluate our TBCNN-pair model, we used the newly published Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015).\(^4\) The dataset is constructed by crowdsourced efforts, each sentence written by humans. Moreover, the SNLI dataset is magnitudes of larger than previous resources, and hence is particularly suitable for comparing neural models. The target labels comprise three classes: Entailment, Contradiction, and Neutral (two irrelevant sentences). We applied the standard train/validation/test split, containing 550k, 10k, and 10k samples, respectively. Figure 2 presents additional dataset statistics, especially those relevant to dependency parse trees.

4.2 Hyperparameter Settings

All our neural layers, including embeddings, were set to 300 dimensions. The model is mostly robust when the dimension is large, e.g., several hundred (Collobert and Weston, 2008). Word embeddings were pretrained ourselves by \texttt{word2vec} on the English Wikipedia corpus and finetuned during training as a part of model parameters. We applied \( \ell_2 \) penalty of \( 3 \times 10^{-4} \); dropout was chosen by validation with a granularity of 0.1 (Figure 2). We see that a large dropout rate (\( \geq 0.3 \)) hurts the performance (and also makes training slow) for such a large dataset as opposed to small datasets in other tasks (Peng et al., 2015). Initial learning rate was set to 1, and a power decay was applied. We used stochastic gradient descent with a batch size of 50.

4.3 Performance

Table 3 compares our model with previous results. As seen, the TBCNN sentence pair model, followed by simple concatenation alone, outperforms existing sentence encoding-based approaches (without pretraining), including a feature-rich method using 6 groups of human-engineered features, long short term memory

\(^4\)http://nlp.stanford.edu/projects/snli/

\(^5\)We applied collapsed dependency trees, where prepositions and conjunctions are annotated on the dependency relations, but these auxiliary words themselves are removed.
Table 3: Accuracy of the TBCNN-pair model in comparison with previous results (Bowman et al., 2015; Vendrov et al., 2015; Rocktäschel et al., 2015). “cat” refers to concatenation; “-” and “◦” denote element-wise difference and product, resp.

<table>
<thead>
<tr>
<th>Model</th>
<th>Test acc. (%)</th>
<th>Matching complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlexicalized features</td>
<td>50.4</td>
<td>$\mathcal{O}(1)$</td>
</tr>
<tr>
<td>Lexicalized features</td>
<td>78.2</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>Vector sum + MLP</td>
<td>75.3</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>Vanilla RNN + MLP</td>
<td>72.2</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>LSTM RNN + MLP</td>
<td>77.6</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>CNN + cat</td>
<td>77.0</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>GRU w/ skip-thought pretraining</td>
<td>81.4</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>TBCNN-pair + cat</td>
<td>79.3</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>TBCNN-pair + cat,◦,-</td>
<td>82.1</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>Single-chain LSTM RNNs</td>
<td>81.4</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>+ static attention</td>
<td>82.4</td>
<td>$\mathcal{O}(n)$</td>
</tr>
<tr>
<td>LSTM + word-by-word attention</td>
<td>83.5</td>
<td>$\mathcal{O}(n^2)$</td>
</tr>
</tbody>
</table>

Table 3: Accuracy of the TBCNN-pair model in comparison with previous results (Bowman et al., 2015; Vendrov et al., 2015; Rocktäschel et al., 2015). “cat” refers to concatenation; “-” and “◦” denote element-wise difference and product, resp.

(LSTM)-based RNNs, and traditional CNNs. This verifies the rationale for using tree-based convolution as the sentence-level neural model for NLI.

Table 4 compares different heuristics of matching. We first analyze each heuristic separately: using element-wise product alone is significantly worse than concatenation or element-wise difference; the latter two are comparable to each other.

Combining different matching heuristics improves the result: the TBCNN-pair model with concatenation, element-wise product and difference yields the highest performance of 82.1%. As analyzed in Section 3.2, the element-wise difference matching layer does not add to model complexity and can be absorbed as a special case into simple concatenation. However, explicitly using such heuristic yields an accuracy boost of 1–2%. Further applying element-wise product improves the accuracy by another 0.5%.

The full TBCNN-pair model outperforms all existing sentence encoding-based approaches, including a 1024d gated recurrent unit (GRU)-based RNN with “skip-thought” pretraining (Vendrov et al., 2015). The results obtained by our model are also comparable to several attention-based LSTMs, which are more computationally intensive than ours in terms of complexity order.

4.4 Complexity Concerns

For most sentence models including TBCNN, the overall complexity is at least $\mathcal{O}(n)$. However, an efficient matching approach is still important, especially to retrieval-and-reranking systems (Yan et al., 2016; Li et al., 2016). For example, in a retrieval-based question-answering or conversation system, we can largely reduce response time by performing sentence matching based on pre-computed candidates’ embeddings. By contrast, context-aware matching approaches as described in Section 2 involve processing each candidate given a new user-issued query, which is time-consuming in terms of most industrial products.

In our experiments, the matching part (Figure 1b) counts 1.71% of the total time during prediction (single-CPU, C++ implementation), showing the potential applications of our approach in efficient retrieval of semantically related sentences.

5 Conclusion

In this paper, we proposed the TBCNN-pair model for natural language inference. Our model relies on the tree-based convolutional neural network (TBCNN) to capture sentence-level semantics; then two sentences’ information is combined by several heuristics including concatenation, element-wise product and difference. Experimental results on a large dataset show a high performance of our TBCNN-pair model while remaining a low complexity order.

Acknowledgments

We thank all anonymous reviewers for their constructive comments, especially those on complexity issues. We also thank Sam Bowman, Edward Grefenstette, and Tim Rocktäschel for their discussion. This research was 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, 61421091, and 61502014.
References


