An In-Depth Experimental Comparison of RNTNs and CNNs for Sentence Modeling

Zahra Ahmadi[1], Marcin Skowron[2], Aleksandrs Stier[1], and Stefan Kramer[1]

[1] Institut für Informatik, Johannes Gutenberg-Universität, Mainz, Germany
[2] Austrian Research Institute for Artificial Intelligence, Vienna, Austria
zaahmadi@uni-mainz.de, marcin.skowron@ofai.at,
stier@students.uni-mainz.de, kramer@informatik.uni-mainz.de

Abstract. The goal of modeling sentences is to accurately represent their meaning for different tasks. A variety of deep learning architectures have been proposed to model sentences, however, little is known about their comparative performance on a common ground, across a variety of datasets, and on the same level of optimization. In this paper, we provide such a novel comparison for two popular architectures, Recursive Neural Tensor Networks (RNTNs) and Convolutional Neural Networks (CNNs). Although RNTNs have shown to work well in many cases, they require intensive manual labeling due to the vanishing gradient problem. To enable an extensive comparison of the two architectures, this paper employs two methods to automatically label the internal nodes: a rule-based method and (this time as part of the RNTN method) a convolutional neural network. This enables us to compare these RNTN models to a relatively simple CNN architecture. Experiments conducted on a set of benchmark datasets demonstrate that CNN outperforms the RNTNs based on automatic phrase labeling, whereas the RNTN based on manual labeling outperforms the CNN. The results corroborate that CNNs already offer good predictive performance and, at the same time, more research on RNTNs is needed to further exploit sentence structure.

1 Introduction

One aim of modeling sentences is to analyze and represent their semantic content for classification purposes. Neural network-based sentence modeling approaches have been increasingly considered for their significant advantages of removed requirements for feature engineering, and preservation of the order of words and syntactic structures, in contrast to the traditional bag-of-words model, where sentences are encoded as unordered collections of words. These neural network approaches range from basic Neural Bag-of-Words (NBoW), which ignores word ordering to more representative compositional approaches such as Recursive Neural Networks (RecNNs) (e.g. [4]), Convolutional Neural Networks (CNNs) (e.g. [6]), and Recurrent Neural Networks (RNNs) models (e.g. [9]) or a combination of them.

RecNNs work by feeding an external parse tree to the network. They are a generalization of classic sequence modeling networks to tree structures and have shown excellent abilities to model word combinations in a sentence. However, they depend on well-performing parsers to provide the topological structure, which are not available for many languages or do not perform well in noisy domains. Further, they often require labeling of all phrases in sentences to reduce the *vanishing gradient problem* [5]. Yet RecNNs implicitly model the interaction among input words, whereas Recursive Neural Tensor Networks (RNTNs) have been proposed to allow more explicit interactions [11]. On the other hand, CNNa are alternative models which apply one-dimensional convolution kernels in sequential order to extract local features. Each sentence is treated individually as a bag of n-grams, and long-range dependency information spanning multiple sliding windows is therefore lost. Another limitation of CNN models is their requirement for the exact specification of their architecture and hyperparameters [12].

We conducted extensive experiments over a range of benchmark datasets to compare the two network architectures: RNTNs and CNNs. Our goal is to provide an in-depth analysis of how these models perform across different settings. Such a comparison is missing in the literature, likely because recursive networks often require labor-intensive manual labeling of phrases. Such annotations are unavailable for many benchmark datasets. In the next section, we propose two methods to label the internal phrases automatically. Later, we investigate whether there is an effect of using constituency parsing instead of dependency parsing in the RNTN model. In this way, we aim to contribute to a better understanding of the limitations of the two network models and provide a foundation for their further improvement.

2 Method

Recursive Neural Tensor Network Architecture. RNTNs [11] are a generalization of RecNNs where the interactions among input vectors are encoded in a single composition function (Figure 1a). Here, we propose two methods for the automatic labeling of the phrases for RNTNs:

- **Rule-based method:** The RNTN model was first proposed for sentiment analysis purposes. Hence, our first approach uses a rule-based method to determine the valence of a phrase. We use four types of dictionaries: A dictionary of sentiments carrying terms (from unigrams to phrases consisting of n-gram words) with a corresponding sentiment score in the range of $[-k, +k]$, a negation dictionary, a dictionary of intensifier terms with a weight range of $[1, +k]$, and a dictionary of diminishers with a weight range of $[-k, -1]$. The analysis of a phrase is conducted from the end, backward to the beginning: If any sentiment term is found, we update the sentiment of the phrase from neutral to the value of the sentiment term in the dictionary. Then we search backwards for an intensifier or diminisher term. We increase or decrease the absolute value of the sentiment based on the weight of the intensifier/diminisher term and if required we adjust the score to a pre-defined range.In the next step, we adjust the score for a negation term. If one is found and there is no intensifier/diminisher before the sentiment term, the sentiment is reversed; otherwise if the phrase includes both the negation term and an intensifier/diminisher, the sentiment is set to weak negative. As an example, consider both *"not very good"* and *"not very bad"* terms, where both sentiments are weak negative.
- **CNN-based method:** An alternative approach to labeling the phrases is to use a pre-trained CNN model. We use the architecture proposed here (see below for the

description) to train a model on the sentence level, and use the resulting model to label the internal phrases for the RNTN. In this way, the RNTN can be applied to domains other than sentiment classification as well. The CNN model receives the complete sentences and their label as training data and will label the internal phrases in the test phase.

Convolutional Neural Network Architecture. Deep convolutional neural networks have led to a series of breakthrough results in image classification. Although recent evidence shows that network depth is of crucial importance to obtain better results [3, 2], most of the models in the sentiment analysis and sentence modeling literature use a simple architecture, e.g. [6] uses a one-layer CNN. Inspired by the success of CNNs in image classification, our goal is to expand the convolution and Max-Pooling layers in order to achieve better performance by deepening the models and adding higher non-linearity to the structure. However, deeper models are also more difficult to train [3]. To reduce the computational complexity, we choose small filter sizes. In our experiments, we use a simple CNN model that consists of six layers (Figure 1b): The first layer applies $1 \times d$ filters on the word vectors, where d is the word vector dimension. The essence of adding such a layer to the network is to derive more meaningful features from word vectors for every single word before feeding them to the rest of the network. This helps us achieving better performance since the original word vectors capture only sparse information about the words' labels. In contrast to our proposed layer, Kim uses a so-called *non-static* approach to modify the word vectors through the training phase [6].

The second layer of our CNN model is again a convolution layer with the filters of size $2 \times d$. The output of this layer is fed into a Max-Pooling layer with pooling size and stride 2. The reason for applying such a Max-Pooling layer in the middle layers of the network is to reduce the dimensionality and to speed up the training phase. This layer does not have notable effect on the accuracy of the resulting model. Next, on the fourth layer, convolving filters of size $2 \times d$ with a padding size 1 are again applied to the output of previous layer. Padding preserves the original input size. The next layer applies Max-Pooling to the whole input at once. Using bigger pooling sizes leads to better results [12]. Finally, the last layer is a fully connected SoftMax layer which outputs the probability distribution over the labels.

3 Experiments

3.1 Experimental Settings

In our experiments, we use the pre-trained Glove [10] word vector models[3]: On the SemEval-2016 dataset, we use Twitter specific word vectors. On other datasets, we use the model trained on the web data from Common Crawl which contains a case-sensitive vocabulary of size 2.2 million. Experiments show that RNTNs work best when the word vector dimension is set between 25 and 35 [11]. Hence, in all the experiments, the size of the word vector, the minibatch and the epochs were set to 25, 20 and 100, respectively. We use $f = tanh$ and a learning rate of 0.01 in all the RNTN models. In CNN

[3] `http://nlp.stanford.edu/projects/glove/`

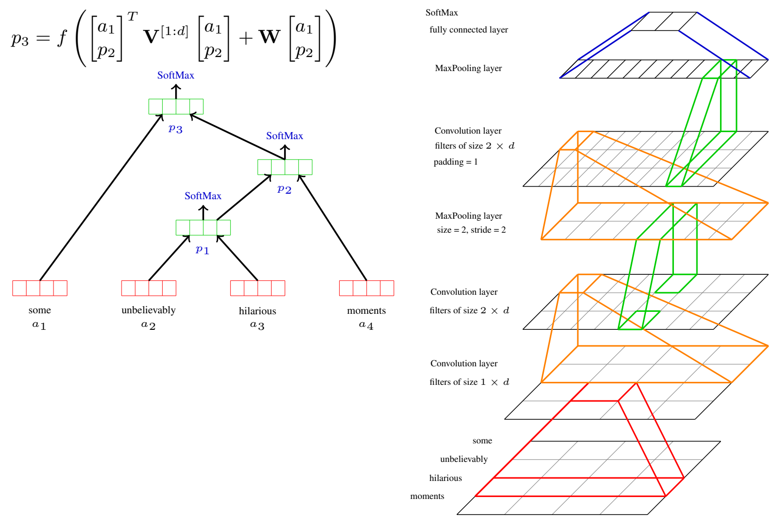
$p_3 = f\left(\begin{bmatrix}a_1\\p_2\end{bmatrix}^T \mathbf{V}^{[1:d]} \begin{bmatrix}a_1\\p_2\end{bmatrix} + \mathbf{W}\begin{bmatrix}a_1\\p_2\end{bmatrix}\right)$
SoftMax
p_3
SoftMax
p_2
SoftMax
p_1
some a_1
unbelievably a_2
hilarious a_3
moments a_4
SoftMax
fully connected layer
MaxPooling layer
Convolution layer filters of size 2 × d padding = 1
MaxPooling layer size = 2, stride = 2
Convolution layer filters of size 2 × d
Convolution layer filters of size 1 × d
some
unbelievably
hilarious
moments

Fig. 1: (a) An example of an RNTN architecture with word vector dimension of size 4 for sentiment classification of a given input sequence, which is parsed by a constituency parser. $\mathbf{V}$ and $\mathbf{W}$ are the tensor matrix and the recursive weight matrix, respectively. (b) Our proposed 6-layered CNN architecture. d is the dimension of the word vector.

models, the number of filters in the convolutional layers are set to 100, 200 and 300, respectively; and the maximum length of the sentences is 32. For shorter sentences, they are padded with zero vectors. In RNTN models which use constituency parsers, we use the Stanford parser [7]. For those models which use dependency parsers, we use the Tweebo parser [8] – a dependency parser specifically developed for Twitter data – for the SemEval-2016 dataset and on the rest of the datasets, we use the Stanford neural network dependency parser [1]. In rule-based methods, we use a dictionary of sentiments consisting of $6,360$ entries with a maximum 2-grams words and a sentiment range of $[-3, +3]$, a negation dictionary consisting of 28 entries, a dictionary of intensifier terms consisting of 47 words with a weight range of $[1, 3]$, and a dictionary of diminishers consisting of 26 entries with a weight range of $[-3, -1]$.

3.2 Task 1: Sentiment Analysis

In the first task, we compare the models on a set of commonly used sentiment analysis benchmark datasets: The Movie Review (**MR**) dataset[4] that has positive or negative class, each contains 5331 instances. As the MR dataset does not have a separate test set, we use 10-fold cross-validation in the experiments. An extended version of the MR dataset relabeled by Socher et al. [11] in the Stanford Sentiment Treebank (**SST-5**)[5] has five fine-grained labels: negative, somewhat negative, neutral, somewhat positive and positive. **SST-5** contains 8544 training sentences, 1101 validation sentences and

[4] `https://www.cs.cornell.edu/people/pabo/movie-review-data/`
[5] `http://nlp.stanford.edu/sentiment/Data`

Table 1: Performance comparison on all datasets. Accuracy and F-measure are averaged over all the classes. n/a indicates non-defined cases as one of the classes was misclassified completely resulting in an undefined value. If an experiment was not applicable, the cell is left with a dash.

Dataset	RNTN										CNN		CNN (Kim model)		Rule-based	
	Constituency parser						Dependency parser									
	Rule		CNN		Manual		Rule		CNN							
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
MR	0.63	0.63	0.70	0.70	-	-	0.50	0.50	0.49	0.49	**0.71**	**0.71**	**0.71**	**0.71**	0.64	0.64
SemEval-2016	0.53	0.45	0.52	0.51	-	-	0.52	0.45	0.50	0.49	0.56	0.56	**0.60**	**0.57**	0.53	0.52
SST-5	0.30	0.28	0.34	0.21	**0.41**	**0.32**	0.30	0.29	0.30	n/a	0.37	0.26	0.39	**0.32**	0.31	0.29
TREC	-	-	0.72	n/a	-	-	-	-	0.33	n/a	**0.86**	**0.86**	0.54	0.57	-	-
Subj	-	-	0.76	0.76	-	-	-	-	0.42	0.42	**0.89**	**0.89**	0.88	0.88	-	-

(a) Rule-based vs. CNN-based RNTN (b) manually labeled RNTN vs. CNN

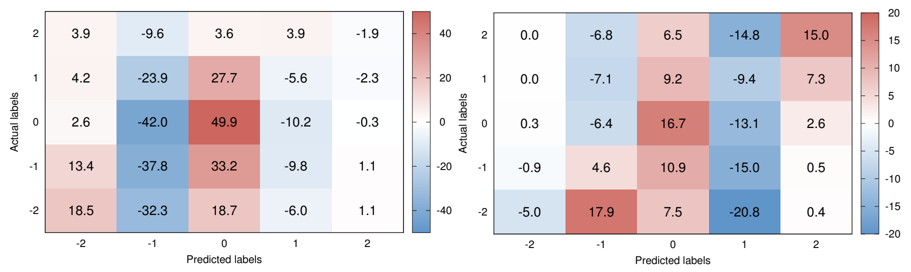

Fig. 2: (a) Heatmap of difference of rule-based RNTN and CNN-based RNTN confusion matrices on the SST-5 phrase set. The numbers are the percentage of normalized differences based on the total number of phrases for each label. (b) Heatmap of difference of the manually labeled RNTN and the CNN model confusion matrices on the SST-5 test set. The numbers in each cell indicate the percentage of normalized differences based on the total number of sentences for each label.

2210 test set. The **SemEval-2016**[6] dataset is a set of tweets labeled as either of the three negative, neutral and positive labels. It has $12,644$ training tweets, 3001 validation tweets and $20,632$ test instances.

- **Comparison of automatic labeling methods:** We first use the manually labeled SST-5 dataset to test the effectiveness of our automatic labeling methods. We extract all the possible phrases of the whole dataset with respect to their parse trees and use our rule-based method to label them. In the next step we train the CNN model on the set of training instances and use the resulting model to label the phrases. The accuracy of the rule-based and the CNN labeling methods are 69% and 40%, respectively. As we see, the overall accuracy of the CNN-based model is significantly lower than that of the rule-based method. To have a better understanding of the classification performance, we look into their confusion matrices. We subtract the corresponding elements of the CNN-based confusion matrix from that of the rule-based variant and normalize them by dividing by the total number of phrases for each label (i.e. $\frac{conf^{i,j}_{rule} - conf^{i,j}_{cnn}}{total_i}$ where i and j are the actual and predicted labels, respectively).

[6] `http://alt.qcri.org/semeval2016/task4/`

Figure 2a illustrates the resulting heatmap. Red color indicates cases where more phrases are predicted by the rule-based method than by the CNN-based method while the blue color indicates the opposite case. We observe that the CNN is a better model to correctly classify somewhat positive (1) and somewhat negative (-1) classes than the rule-based method. In turn, the rule-based method is superior in the classification of the neutral (0) and negative (-2) classes. To have a better interpretation of the numbers in the heatmap, it is beneficial to look at the distribution of labels in the whole population: 2.6%, 11.3%, 67.7%, 14.3% and 4.1% for -2 to $+2$ labels.

- **Constituency parser vs. dependency parser:** The output of a dependency parser is a Directed Acyclic Graph (DAG). However, RNTNs accept a binary-branching parse tree as an input. Therefore, we have binarized the output of dependency parser by starting from the word which does not point to any other word as its parent, and recursively binarize its children list by adding empty nodes when necessary. While analyzing the effect of using a dependency parser instead of a constituency parser in RNTNs (Table 1), a significant loss of performance is visible in some datasets (e.g. MR). This is particularly noticeable when the labeling method is CNN (e.g. 70% to 49% in MR). The reason for this could be the difference of the word order resulting from a dependency parser compared to the n-gram features extracted by the CNN.
- **RNTN vs. CNN:** Table 1 shows a detailed comparison of the RNTN automatic labeled variants to the CNN model and the rule-based method. We have reported the average accuracy and F-measure over all classes. With the same settings of parameters, we see a better performance of the CNN model on the MR and SemEval-2016 datasets. The largest performance (in terms of F-measure) improvement can be observed on the SemEval-2016 dataset, 0.51 to 0.56, for the best performing RNTN and CNN approaches. The possible reasons may be related to the enormously large number of parameters that have to be optimized in the tensor and the effects of the applied automatic labeling of phrases used on the RNTN. Therefore, a future research direction could try to reduce this space and find a better initialization.
- **Effect of automatic labeling on RNTN performance:** Table 1 also presents the performance of the manually labeled RNTN on the SST-5 dataset. As we can see, automatic labeling results in a significant degradation of performance on SST-5. Comparing the results with the CNN model shows that the manually labeled RNTN outperforms the CNN architecture in terms of overall accuracy and F-measure. To have a closer look into the confusion matrix of both methods, we generate a heatmap similar to Figure 2a, this time subtracting the CNN confusion matrix elements from that of RNTN method (i.e. $\frac{conf_{rntn}^{i,j} - conf_{cnn}^{i,j}}{total_i}$). Blue color indicates more prediction of sentences by the CNN model than by the RNTN while the red color indicates the reverse case. Figure 2b indicates that the RNTN has a tendency to classify more instances into neutral (0) and positive (2) labels and it is better at correct prediction of somewhat negative (-1), neutral and positive labels while the CNN is better at classifying negative (-2) and somewhat positive (1) labels. Here, the distribution of sentences over labels is closer to the uniform distribution: 12.6%, 28.6%, 17.6%, 23.1% and 18.1% for -2 to $+2$ labels. Unfortunately, currently there is no other dataset that is manually labeled at the phrase level. A future direction includes further evaluation of the impact of the phrase labeling accuracy on various datasets.

3.3 Task 2: Sentence Categorization

We test this task on two datasets: The TREC[7] question dataset, where the goal is to classify a question into six coarse-grained question types (whether a question is about an entity, a person, a location, numeric information, abstract concepts or an abbreviation), and the Subj[8] dataset, where the goal is to classify a sentence as being objective or subjective. The TREC dataset has 5452 training instances and 500 test sentences. The Subj dataset contains $10,000$ sentences in total but it does not have a separate test set, therefore we use 10-fold cross-validation. The results are reported in the bottom section of Table 1. In these experiments only CNN-based methods are applicable. We observe that the CNN model outperforms RNTN versions, and dependecy parsing drastically reduces the performance of the RNTN.

3.4 Comparison of CNN architectures

In the next experiment, we compare our proposed deep CNN architecture to a one layer CNN to find out the cases where the deep structure is beneficial. The one layer CNN architecture [6] has several parallel filters of different sizes and a max-pooling layer. In our experiments, we have used 100 filters of size 3, 4, and 5. Classification results (see next to last column of Table 1) indicate that the performance of the one layer architecture is comparable to the proposed deep architecture on the MR dataset and that it performs better on the rest of sentiment datasets. The performance of Kim's architecture on the SST-5 dataset is comparable to the RNTN based on manual labeling. These results highlight the importance of keyphrase recognition in sentiment tasks, where applying larger filters is more beneficial than having several layers of small filters. However, on the other sentence categorization datasets, i.e. TREC and Subj, the proposed deep CNN outperforms the flat architecture.

4 Conclusions

In this paper we studied two well-known deep architectures, CNNs and enhanced versions of RNTNs, in the context of sentence modeling. In order to avoid the labor-intensive task of manually labeling the internal phrases for recursive networks, we proposed two methods to automatically label them for training and tuning phases: a rule-based method which is specifically used for sentiment prediction and a CNN based method for general purposes. Considering this part of study, the evaluation results on the SST-5 dataset indicate that the CNN method has a tendency to assign a positive or negative polarity to the phrases while the rule-based method classifies many of them as neutral. Based on the presented automatic labeling methods of internal nodes, we conducted an in-depth study of the RNTN model and compared the model to a relatively simple deep CNN architecture. Experimental results conducted on an extensive set of standard benchmark datasets demonstrate that the proposed CNN model outperforms

[7] `http://cogcomp.cs.illinois.edu/Data/QA/QC/`
[8] `https://www.cs.cornell.edu/people/pabo/movie-review-data/`

the RNTN variants with automatic phrase labeling, whereas the RNTN with manual labeling (if available) outperforms the CNN. However, in that case, a one layer CNN with several filters of different sizes is comparable to the manually labeled RNTN. These results demonstrate that the syntactic structure of a sentence will help in the classification performance when it is possible to accurately label the internal nodes of a parse tree, otherwise CNN is more successful at representing the meaning of the sentence with respect to the task. The findings show that there is still room for improvement of RNTN variants in terms of determining tensor functions in a more informed manner.

Acknowledgement

The authors thank PRIME Research for supporting the first author during her research time. The second author is supported by the Austrian Science Fund (FWF): P27530-N15.

References

1. Chen, D., Manning, C.D.: A fast and accurate dependency parser using neural networks. In: Proceedings of Empirical Methods in Natural Language Processing. pp. 740–750 (2014)
2. Conneau, A., Schwenk, H., Barrault, L., Lecun, Y.: Very deep convolutional networks for text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. pp. 1107–1116 (2017)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
4. Irsoy, O., Cardie, C.: Deep recursive neural networks for compositionality in language. In: Advances in Neural Information Processing Systems. pp. 2096–2104 (2014)
5. Iyyer, M., Manjunatha, V., Boyd-Graber, J., Daumé III, H.: Deep unordered composition rivals syntactic methods for text classification. In: Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics. pp. 1681–1691 (2015)
6. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of Empirical Methods in Natural Language Processing. pp. 1746–1751 (2014)
7. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics. pp. 423–430 (2003)
8. Kong, L., Schneider, N., Swayamdipta, S., Bhatia, A., Dyer, C., Smith, N.A.: A dependency parser for tweets. In: Proceedings of Empirical Methods in Natural Language Processing. pp. 1001–1012 (2014)
9. Li, J., Luong, M.T., Jurafsky, D., Hovy, E.: When are tree structures necessary for deep learning of representations? In: Proceedings of Empirical Methods in Natural Language Processing and Computational Natural Language Learning. pp. 2304–2314 (2015)
10. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of Empirical Methods in Natural Language Processing. pp. 1532–1543 (2014)
11. Socher, R., Perelygin, A., Wu, J.Y., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.P.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of Empirical Methods in Natural Language Processing. pp. 1631–1642 (2013)
12. Zhang, Y., Wallace, B.: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. CoRR abs/1510.03820 (2015)