Title
Part-Of-Speech Tag Embedding for Modeling Sentences and Documents

Author
Yu, Dong Jin

Publication Date
2016

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Part-Of-Speech Tag Embedding for Modeling Sentences and Documents

A thesis submitted in partial satisfaction

of the requirements for the degree of Master of Science

in Computer Science

By

Dong Jin Yu

2016

ABSTRACT OF THE THESIS

Part-Of-Speech Tag Embedding for Modeling Sentences and Documents

By

Dong Jin Yu

Master of Science in Computer Science

University of California, Los Angeles, 2016

Professor Wei Wang, Chair

In sentence modeling, neural network approaches that leverage the tree-structural features of sentences have recently achieved state-of-the-art results. However, such approaches require complex architectures and are not easily extensible to document modeling. In this paper, we propose a very simple convolutional neural network model that incorporates Part-Of-Speech tag information (PCNN). While our model can be easily extensible to document modeling, it shows great performance on both sentence and document modeling tasks. As a result of sentiment analysis and question classification tasks, PCNN achieves the performance comparable to that of other more complex state-of-the-art models on sentence modeling and outperforms them on document modeling. We also make efforts to explore the effect of POS tag embeddings more thoroughly by conducting various experiments.

The thesis of Dong Jin Yu is approved.

Douglas S. Parker

Junghoo Cho

Wei Wang

University of California, Los Angeles

2016

Table of Contents

Acknowledgments or Preface

The work presented in this paper is based on:

Dong Jin Yu, Ariyam Das, Wei Wang, Fei Sha. Part-Of-Speech Tag Embedding for Modeling Sentences and Documents (in preparation for publication).

Dong Jin Yu was a project lead. Ariyam Das helped running experiments. Wei Wang and Fei Sha were advisors.

1 Introduction

Deep learning models have shown remarkable performance in computer vision [1], [2], [3] and speech recognition [4] in recent years. Following the success in those areas, deep learning models have also been successfully applied to natural language processing tasks, including language modeling [5], [6] and sentence modeling [7], [8]. In particular, the recent discoveries of various deep neural network architectures to model sentences have led to significant advancement in the field of natural language processing. Among those architectures are convolutional neural networks (CNN), recurrent neural networks, and recursive neural networks. Recently, tree-based convolutional neural network (TBCNN), dependency-based convolutional network (DBCNN), and tree-structured LSTM (TreeLSTM), all of which exploit tree representations of sentences, have achieved the state-of-the-art performance on sentence modeling [9], [10], [11]. Although those tree-structured models have been shown to perform well on sentence modeling by capturing structural features, it is not clear how to extend them to modeling documents which consist of multiple sentences. On the other hand, sequential models, such as sequential CNN and bidirectional LSTM [8], [11] have limited capacity for learning structural features of sentences although they can naturally extend to modeling documents. The model we introduce in this work compensates for the shortcomings of both sides. Our model, named PCNN, is based on Kim's sequential CNN [8] but incorporates POS tag information into the network by embedding the tags into a continuous, high-dimensional space. Because PCNN is a simple sequential CNN, it can be naturally applied to modeling documents of multiple sentences without any modification to architectural settings. Moreover, since PCNN incorporates POS tag information, we expect PCNN to infer the structural features of sentences to some

extent. To the best of our knowledge, there is no published work that uses POS tag embedding for either sentence or document modeling. Eventually, we show that PCNN is a simple, flexible model that can be used for modeling both sentences and documents through various experiments including sentiment analysis, question type classification, and document classification. The rest of the report is organized as follows. Section 2 is a literature review. In Section 3 we present our proposed approach. The experimental results are reported in Section 4. Then we perform some detailed analysis of our method and present some of its limitations in Section 5. Finally, we conclude and suggest the potential future directions in Section 6.

2 Literature Review

2.1 Deep Learning

Various deep learning models recently achieved great success in various areas such as object detection, image classification, and speech recognition. The depth adds the representational power to the model by introducing multiple non-linear layers that can capture complex patterns and decision boundaries from the input dataset. The presence of multiple non-linear layers has empirically shown that hierarchical feature representation is possible in deep learning models. The lower layers of deep learning models tend to capture more generic features. For example, in image classification tasks, lower layers learn how to distinguish edges and curves. On the other hand, the higher layers tend to capture more high-level, abstract features that are less sensitive to the pixel-level variations in the input images but are more directly related to the classification results. Two different inputs that cause completely different activation patterns in lower layers can still invoke the identical activation patterns in higher layers. This capability allows deep

learning models to recognize elephants in different poses, different sizes. The ability to capture such abstract features at higher layers is a result of hierarchy of feature representation that is made possible through the re-use of lower layer features. Although the findings discussed above are purely empirical observations without rigorous mathematical proofs, there also exist published works that are based on more concrete theoretical findings. Pascanu shows in [23] that deep models can represent highly complex functions that have much higher number of regions than functions represented by shallow models when the number of parameters stays the same. Even then, there is not enough theoretical findings to understand and explain completely why deep models work well. However, given the previous empirical successes of deep models, it is still worth exploring the capability of deep models by applying them to many different domains.

2.2 Deep Learning in Natural Language Processing

Following the success of deep learning in vision and speech, many deep learning researchers have shifted their focus to natural language processing. The following sections will review and compare some of the recent deep learning models that have pushed the boundaries of research.

2.2.1 Sequential Convolutional Neural Network

The first and most basic deep learning model to note is a sequential convolutional neural network. This type of model has been first introduced in [24] and [8]. Since our PCNN is an extension of Kim's sequential convolutional neural network (sequential CNN) in [8], we will focus on the introduction of the model introduced in [8] in details. The model converts each word in the sentence into a continuous, high-dimensional word vector.

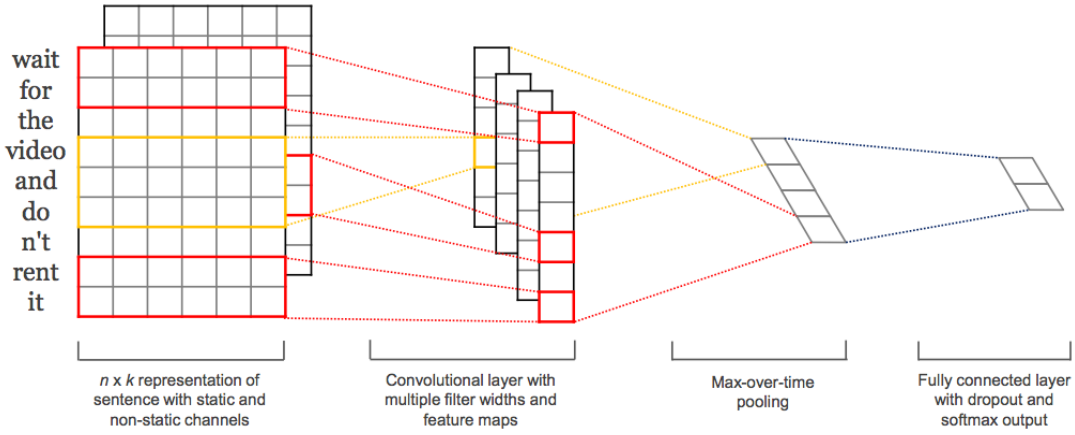

Figure 1: Kim's Sequential CNN model architecture

Therefore, for each sentence, we obtain a sentence matrix as shown in the figure above. The first layer of the model is a convolutional layer, which applies a filter matrix of h by k to a window of h words to produce a feature. The ith feature generated from a window of words $x_{i:i+h-1}$ is represented by

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b).$$

The filter is applied to every possible window of words in the sentence matrix in a sliding window fashion. Then, the next layer consists of the produced features

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}],$$

which is called feature map. The actual model contains multiple filters, which result in multiple feature maps. Then there is a max-over-time pooling layer, which takes the max value from each feature map and propagates it to the next layer. The idea is to extract the most important feature from each feature map. At the end, those extracted features are passed through a fully-connected layer and softmax layer to produce the probability distribution over class labels. This model uses

a very simple idea of convolutional layers, but it produced notable results and achieved the state-of-the-art performance in various sentence classification tasks. This model is considered sequential because the filters in the convolutional layer are applied to words sequentially in the order shown in the actual sentence. Thus, sequential CNNs present some limitations in their capability to capture structural features as well as long-distance dependencies of the sentence.

2.2.2 Tree-based Convolutional Neural Network

In order to compensate for the weaknesses of sequential CNNs, researchers have developed tree-based CNN models and achieved better results. Ma came up with dependency-based CNN in [10]. One of the weaknesses of sequential CNNs is that they can't capture long-distance relationships effectively. While the example sentence in the figure below is obviously positive, it is likely to be classified as negative by sequential CNNs because the sentence contains the highly negative word "shortcomings," and the distance between "Despite" and "shortcomings" is quite long.

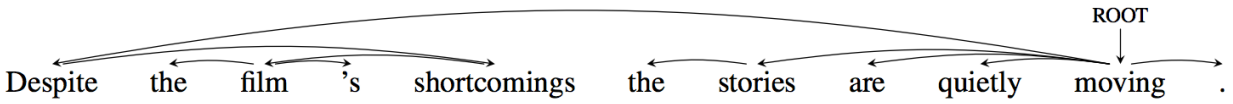

Figure 2: Dependency tree of an example sentence

However, by constructing the dependency tree of the sentence as shown in the figure above and applying convolution on ancestor paths in the tree, the model can capture the long-distance relationships in the presence of words such as "despite" or "even though." Another successful model that uses tree structures of sentences is proposed by Mou in [9]. Tree-based CNN

(TBCNN) [9] tries to take advantage of the inherent sentence structures (like constituency trees or dependency trees). Learning this structural information provides significant improvements over sentence modeling tasks. Moreover, these structures can be learnt effectively with short propagation paths, as opposed to recursive neural nets (RNNs) which face considerable difficulties in learning deep dependencies between words in the sentence because of the long propagation paths. As discussed, tree-based CNNs improve the performance of sequential CNNs by effectively capturing long-distance relationships and structural features of sentences. However, tree-based CNN models require complex algorithms and architectures that are quite difficult to implement in practice. Moreover, it is unclear how to extend tree-based CNN models to apply to document-level tasks.

2.2.3 Recursive Neural Network

The last branch of neural networks that are widely used in natural language processing is recursive neural network. While recurrent neural networks exploit depth in time by maintaining an internal state of the model throughout the training process, recursive neural networks leverage depth in structure. After constructing a tree representation of each sentence, features of the sentence are computed recursively starting from the leaf nodes in a bottom-up fashion. One of the most popular recursive neural networks, the Recursive Neural Tensor Network (RNTN), was introduced by Socher in [7]. A single tensor-based composition function is used repetitively to compute a vector representation at each node of the tree, and the final vector at the root of the tree is used to represent the entire sentence. By exploiting the recursive nature of sentences, this model captures different aspects of compositionality in language, and its computation on an

example sentence is presented in the figure below.

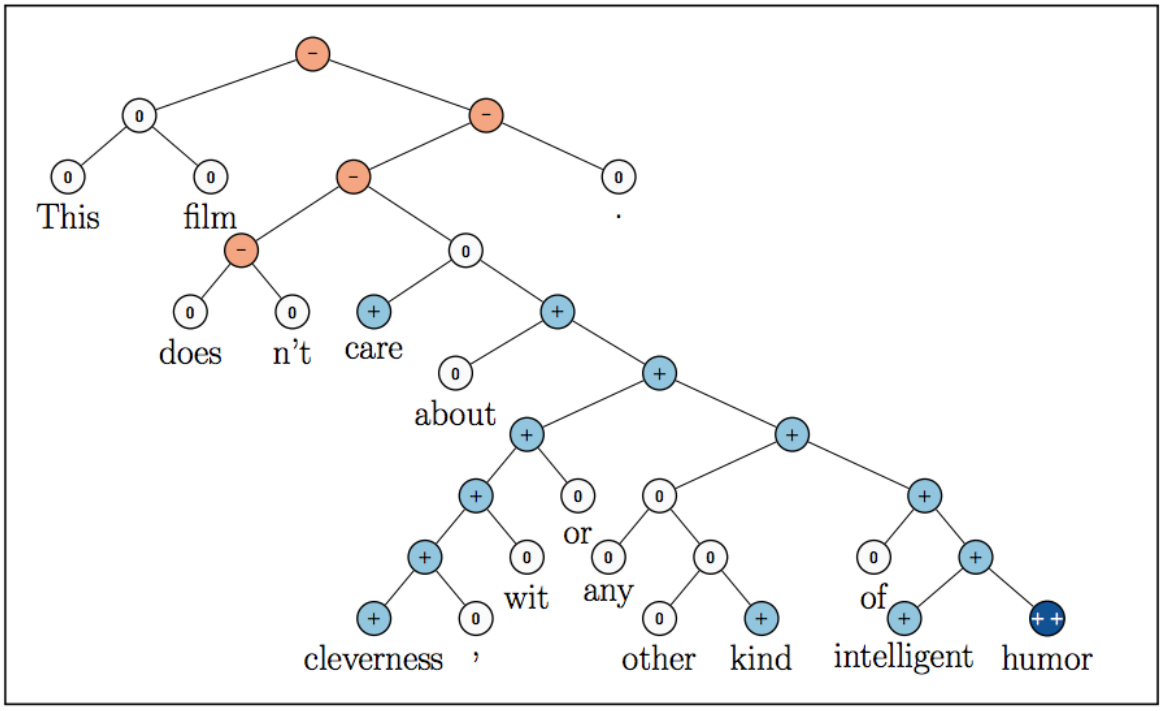

Figure 3: Example of the Recursive Neural Tensor Network accurately predicting sentiment

Following Socher's RNTN, Irsoy proposed a deep recursive neural network (DRNN) in [18]. Although recursive neural networks are deep in structure, they lack the capacity for hierarchical representation which often exists in modern deep learning architectures. DRNN is constructed by stacking multiple layers of individual recursive nets as shown in the figure below.

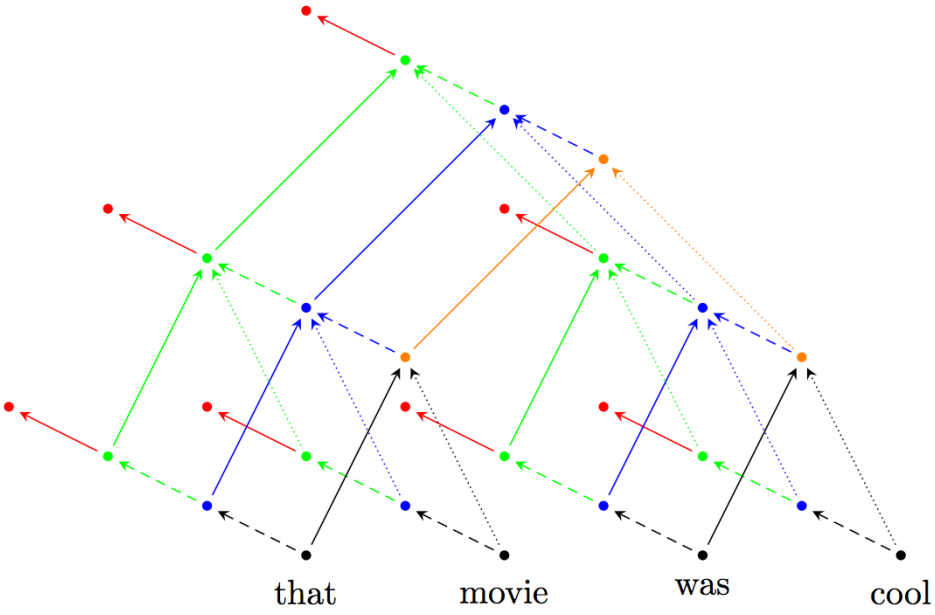

Figure 4: Operation of a 3-layer deep recursive neural network

The additional notion of depth added to DRNN allows the model to find more high-level, abstract features that are more directly related to classification results at higher layers through the hierarchical representation of hidden features. In 2015, Tai proposed a new model called tree-structured LSTM that combines recurrent neural net and recursive neural net architecture. While the standard LSTM constructs its state from the hidden state of the LSTM unit in the previous time step and the current input, the tree-structured LSTM computes its state from the hidden states of child units and the current input as shown in the figure below.

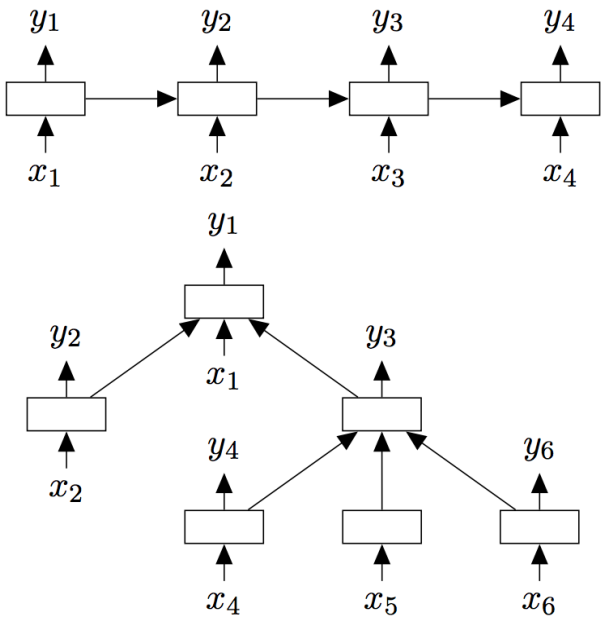

Figure 5: **Top**: A standard LSTM. **Bottom**: A tree-structued LSTM.

By maintaining the internal state at each node, tree-structured LSTM addresses the long propagation path problem of recursive neural networks. Moreover, by maintaining and updating the internal state in a bottom-up manner within the tree representation of the sentence, tree-structured LSTM exploits the inherent structural features of sentences. Lastly, Chen proposed a gated recursive neural network (GRNN) to avoid the need to parse each sentence into a tree representation in [25]. Instead, the model represents every sentence as a full binary tree structure and uses two kinds of gates, reset and update gates, at each node to control how to propagate the information from child nodes upward to their parent node. Although this model has an advantage over other tree-based models in that it does not need to use any external parser, the performance is not as successful.

3 Model

We extend the sequential CNN [8] to incorporate POS tag information. We propose 3 model variants, each of which has advantages over the others. All of the variants keep a separate embedding matrix for POS tags and have different ways to obtain a vector representation, V_t, for the t th word from the corresponding word embedding vector, W_t, and POS tag embedding vector, T_t.

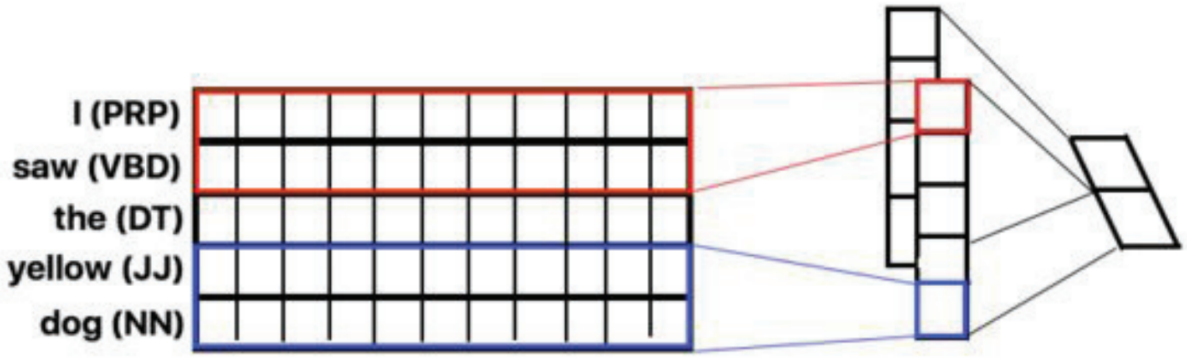

Figure 6: Model architecture with two filters for an example sentence. Each row in the sentence matrix is a vector representation of each word V_t.

3.1 PCNN-con

PCNN-con uses a concatenation strategy where each V_t is computed by simply concatenating the corresponding W_t and T_t. This simple concatenation strategy raises concerns about the extent to which word embeddings and POS tag embeddings can be mingled together to produce a meaningful vector representation for each token. However, the forward propagation through the convolutional layer and the fully-connected layer allows word embeddings and POS tag embeddings to interact with each other as the results of our experiments will show in Section 4.

Lastly, this model requires the simplest network architecture and the least number of parameters to train, leading to the shortest training time.

3.2 PCNN-mat

PCNN-mat keeps an additional 2-D trainable weight matrix, Q, used to compute V_t. The neighboring POS tag embedding vectors of a set window size form a matrix, which is then multiplied by W_t and Q. The following equation shows how to compute V_t (window size of 3 is used in the equation). The idea behind this model is that an additional trainable matrix Q will allow the model to learn meaningful V_t's by incorporating the local grammatical context information along with the word embedding.

$$V_t = [W_t]^T [Q][T_{t-1}, T_t, T_{t+1}]$$

3.3 PCNN-ten

PCNN-ten keeps an additional 3-D trainable weight tensor, Q. Q can also be thought of as F different 2-D weight matrices. Each of F slices of Q captures a compositional feature between the word and the POS tag. For each token, which consists of word embedding and POS tag embedding, Q extracts F compositional features that are meaningful to the objective function of the model. This model requires the highest number of parameters, but addresses the issue that word embeddings are often too generic and don't differentiate between the same words with different POS tags. Although this model makes the most sense intuitively, the complexity and size of the model may lead the model to work against our intuition. Whether the model can learn

meaningful values for the tensor with such large number of parameters from datasets of limited size is questionable. The below presents the equation for computing V_t.

$$V_t = \begin{pmatrix} W_t \\ T_t \end{pmatrix}^T_{1\times(d+m)} \left(Q \right)_{(d+m)\times F\times(d+m)} \begin{pmatrix} T_t \\ W_t \end{pmatrix}_{(d+m)\times 1}$$

3.4 Key Points

3.4.1 Initialization of Embeddings

The optimization problem present within deep neural net architectures is a non-convex problem. Recently, although gradient-based methods are widely used to find the optimal value in such a continuous, high-dimensional space of deep neural nets and showed successful results in various domains, it is inevitable that those found values are local optima. Thus, the initialization scheme has been an extremely important part of training deep neural nets. As discussed in Kim's work [8], the use of word embeddings, which were trained on GoogleNews dataset with word2vec [21], increases the performance of the CNN model on sentence classification task by significant amount. Inspired by Kim's results, we also used the identical word embeddings to initialize our PCNN. For POS tag embeddings, we tried to use word2vec to obtain POS tag embeddings for initialization. We converted each token in training set documents into a POS tag and tried to learn embeddings for each POS tag by treating each tag as a word. However, the POS tag embeddings trained that way did not improve performance of PCNN compared to POS tag embeddings initialized with uniformly distributed random values. How to train POS tag embeddings seems to be another potential thread of research in the future.

3.4.2 Regularization

We employ two main techniques to regularize the network: dropout and L2-norm rescaling. Firstly, dropout is a standard technique to regularize deep neural net models as it has an effect similar to that of an ensemble of an exponential number of different architectures by deactivating a different subset of neurons for each training step. More concrete mathematical studies show that dropout also improves the generalization bound for empirical risk minimization [22]. At dropout layers, each neuron is deactivated with a specified probability during training. During inference, the learned weight vectors are scaled by the inverse of the specified probability. Unlike the architecture introduced by [8], our architecture requires some computation in the embedding layer (to mingle word embeddings with POS tag embeddings). Thus, we use two dropout layers: one after the embedding layer and the other after the fully-connected layer. In addition to applying two dropout layers, we also apply L2-Norm rescaling technique to regularize our model further, following [8]. We have observed that the values of the elements in the weight matrix representative of the fully-connected layer tend to keep increasing after reaching the best performing state. Once the state is passed, our model performs very well on training data set, reaching 99.9% accuracy towards the end, while it performs poorly on test data set, showing a strong signal of overfitting. By putting an upper limit on the value of L2-norm of the weight matrix, our model is forced to rescale all the values in the weight matrix after each training step whenever the limit is reached. The presence of the limit leads to slower convergence, but it allows our model to reach higher accuracy at the end gradually.

4 Experiments

4.1 Sentence Classification

We first evaluate our models on sentence classification tasks. We use L2-constraint and dropout for regularization [12] and stochastic gradient descent for optimization. Although the length of the sentences varies, all the datasets contain only single sentence entries. We employ sentiment analysis and question type classification.

4.1.1 Datasets

1. MR: This dataset contains 10662 movie reviews with one sentence per review. Each review is classified into either positive or negative. Since no standard splits are provided, we use 10-fold cross validation [13].
2. SST: The Stanford sentiment treebank, a widely used dataset for evaluating sentence models, consists of 11855 movie reviews, each of which is given one of 5 labels (strongly positive, positive, neutral, negative, strongly negative). The train/dev/test splits are provided.
3. TREC: This dataset involves 5952 questions from which 500 are for testing. Each question is labelled as one of 6 different question types. Since validation set is not provided, we randomly choose 10% of the training set [14].

4.1.2 Training Details

We use the publicly available word2vec vectors that were trained on Google News using the continuous bag-of-words architecture [5]. The vectors are 300 dimensional. For POS tag vectors,

we use 24 dimensions. In order to promote a fair comparison with our baseline, CNN-non-static [8], we use the identical set of hyperparameters whenever possible. We use filter windows of 3, 4, 5 with 100 feature maps each, dropout rate of 0.5 after the convolutional layer, l2 constraint of 3 on the last fully-connected layer, and mini-batch size of 50. Lastly, stochastic gradient descent is used with Adadelta update rule [15]. For each epoch, the training set is randomly shuffled.

4.1.3 Results

Group	*Method*	*MR*	*SST*	*TREC*
Baseline	CNN-non-static [8]	81.5%	47.4%	93.6%
Our Work	**PCNN-con**	**81.7%**	**49.6%**	**95.8%**
	PCNN-mat	**81.2%**	**46.8%**	**94.4%**
	PCNN-ten	**81.3%**	**47.0%**	**94.2%**
CNN Variants	Dependency CNN (ancestor + sibling + sequential) [10]	81.9%	49.5%	95.4%
	Dynamic CNN [16]	-	48.5%	93.0%
	d-TBCNN [9]	-	51.4%	96.0%
Recursive NN	Deep RNN [17]	-	49.8%	-
	Constituency Tree-LSTM [11]	-	51.0%	-
Recurrent NN	Bidirectional LSTM [4]	-	49.1%	-

Table 1: Accuracy of sentence classification tasks in comparison with various models

Table 1 compares the results from our models to those of other state-of-the-art models in all 3 datasets. For MR, PCNN-con performs slightly better than our baseline. For SST, PCNN-con shows non-trivial improvement over our baseline and performs better than some of other complex models. PCNN-con is ranked 4th in the category after d-TBCNN, Constituency Tree-LSTM, and Deep RNN, all of which exploit the tree representations of sentences. It should be noted that Tree-LSTM and Deep RNN are trained on labeled phrases while PCNN is only trained on full sentences. For TREC, our PCNN-con again shows great improvement over our baseline and is ranked 2nd after only d-TBCNN. For all three datasets, PCNN-mat and PCNN-ten do not

perform well. One of the possible reasons is that the models are overly complicated and do not reflect a good way to mingle word and POS tag embeddings. Because of the additional trainable matrix/tensor before the embedding layer, the models may not be able to back-propagate gradients well enough to learn accurate word and POS tag embeddings. It is also possible that the datasets don't contain enough entries for those models to capture meaningful representations. However, the performance of PCNN-con achieves near stateof-the-art results overall. This finding should be even more highlighted by the fact that PCNN-con doesn't use tree representations of sentences and learns structural information only through POS tags. The results suggest that POS tags can be useful prior for sentence modeling. For the sake of completeness, it is important to mention that we performed similar experiments by embedding the POS tag in the tree-LSTM architecture. However, in the latter case, we did not get any significant improvements. This limitation is further discussed in Section 5.1.

4.2 Document Classification

4.2.1 Dataset

Yelp 2013: This dataset is a subset of the dataset introduced in Yelp Dataset Challenge3 in 2013. This is derived and used in [18] and contains 78,966 reviews. The average number of sentences and words per review is 10.89 and 189.3 respectively. The dataset is already split into training, development, and testing sets with a 80–10–10 split and preprocessed.

4.2.2 Training Details

We use the same word2vec vectors as in the sentence classification tasks. The dimension of POS tag vectors is 24. Different from sentence classification tasks, we use filter windows of 3, 4, 5, and 6 with 100 feature maps for each. The dropout is applied after the embedding layer with 0.3 as well as after the convolutional layer with 0.5.

4.2.3 Our Competitors

All the methods described here are implemented by Tang and used in his work in [18]. The results from these methods are used to compare with those of our models.

1. UPNN (no UP): This model employs sequential CNN [8] to represent each sentence. The resultant sentence vectors are fed into an average pooling layer to obtain the document representation.

2. UPNN (full): The main method was proposed by Tang in [18]. The model has a similar architecture to that of UPNN (no UP), but incorporates user and product embeddings.

3. RNTN + Recurrent: Recurrent neural network is used to represent documents by taking a sentence as an input at each time step where each sentence is represented with RNTN [7]. The average of hidden vectors of the recurrent neural network is used as a final feature vector for classification.

4. Paragraph Vector: Paragraph Vector model was introduced in the paper. [19]. The model is used to learn fixed-length vector representations of variable-length pieces of texts through unsupervised learning. The learned vectors can be used as feature vectors in classification tasks.

5. TextFeature: Hand-crafted text features, including but not limited to word/character ngrams, sentiment lexicon features, and negation features, are used [20].

6. Trigram: Unigrams, bigrams, and trigrams are used as features to train a support vector machine classifier.

4.2.4 Results

Group	*Method*	*Accuracy*
Our Work	**PCNN-con**	**62.8%**
	PCNN-mat	**62.0%**
Deep Neural Network Approaches	UPNN (no UP)	57.7%
	UPNN (full)	59.6%
	RNTN + Recurrent	57.4%
	Paragraph Vector	55.4%
Others	TextFeature	55.6%
	Trigram	56.9%

Table 2: Results of document classification tasks in comparison with various models

As shown in Table 2, PCNN-con and PCNN-mat are ranked the first and the second4 . Our models outperform UPNN (full), which incorporates user and product information. Moreover, our models also outperform RNTN + Recurrent and Paragraph Vector by far even though our models are much simpler. These promising results show that incorporating POS tag information alone can promote good performance on document modeling.

5 POS Tag Embedding Analysis

5.1 Tree-LSTM with POS Tag Embedding

We tried the Tree-LSTM architecture with POS tag embedding on the SST dataset to see if we can significantly improve the current best record stated in [11]. The Tree-LSTM units are essentially similar to LSTM units except that the former has multiple forget gates (one for each child) and as a result, its memory cell updates are dependent on the states of possibly many child units. In our experiments, we used both the architectures mentioned in [11] -- Dependency Tree-LSTM and Constituency Tree-LSTM (latter uses only binary Tree-LSTM units). In the constituency tee-LSTM, the leaf nodes take the corresponding word vectors as input, as opposed to a dependency tree-LSTM. All our experiments used similar architectures, hyperparameters and training plan (like learning rate of 0.1, dropout rate of 0.5, etc.) as mentioned in [11] except that we reduced the minibatch size to 15. Typically, in most of the runs, the constituency tree-LSTM outperforms dependency tree-LSTM. However, even after adding POS tag embedding, the best improvements in accuracy, that we obtained, were around 0.14% for dependency tree-LSTM and 0.09% for constituency tree-LSTM. It is important to mention a caveat that these aforementioned results for the tree-LSTMs are the best obtained in a single run. The improvement in mean accuracy averaged over five or more runs is marginal. Thus, we find that POS tag embedding can become less useful if the underlying architectures (like tree-LSTM [11] or tree-based CNN [9]) can represent the hidden sentence structures well. On the other hand, these architectures face severe challenges in being adopted for document modeling tasks. For such purposes, incorporating POS tags can be extremely beneficial.

5.2 Static vs. Non-static Representation

	Mean	*Max*
PCNN-con (static)	45.8%	47.0%
PCNN-con (non-static)	48.8%	49.6%

Table 3: Results of static vs. non-static representations on SST

In order to scrutinize the effect of POS tag embedding, we also trained PCNN-con models on SST dataset where we didn't allow gradients to back-propagate to POS tag embeddings. We allowed the gradients to back-propagate to word embeddings. Because the experiment is stochastic, we trained 5 different models by randomly choosing 10% of training set as a validation set each time and recorded the mean and the max performance. We did the same for standard PCNN-con where the gradients were allowed to back-propagate to both POS tag and word embeddings. We employed early stopping if there was no improvement on validation set for 3 consecutive epochs. Table 3 summarizes the results of the experiment.

6 Conclusion and Future Work

In this work, we introduced a simple but powerful idea of incorporating POS tags for sentence and document modeling tasks. This approach is very intuitive and it significantly improves upon the accuracies obtained by previous baseline methods like [8] as substantiated by our experimental results. Nevertheless, the power of POS tags can be limited if the underlying architectures [9], [10], [11] are able to learn the structure of the sentences well. However such tree-based architectures [9], [10], [11] are extremely difficult and challenging to extend to paragraph and document modeling tasks. And therein lies the attraction of POS embedding,

which can open up many potential new avenues of research. In future, we would further like to explore this idea by incorporating the positional dependencies of the POS tags through the use of sequential n-grams on the POS tags themselves. We also plan to try and use different weight matrices for various POS tags.

References/Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS*, 2012.

[2] C. Szegedy et al. Going deeper with convolutions. In *CVPR*, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[4] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*, 2013.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, 2013.

[6] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, 2014.

[7] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 2013.

[8] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, 2014.

[9] L.Mou, H.Peng, G.Li, Y.Xu, L.Zhang, and Z.Jin. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of EMNLP*, 2015.

[10] M. Ma, L. Huang, B. Xiang, and B. Zhou. Dependency-based convolutional neural networks for sentence embedding. In *Proceedings of ACL*, 2015.

[11] K.Tai, R.Socher, and C.Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, 2015.

[12] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of ICML*, 2014.

[13] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *arXiv preprint, arXiv: 1207.0580*, 2012.

[14] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, 2005.

[15] X. Li and D. Roth. Learning question classifiers. In *Proceedings of ACL*, 2002.

[16] M. Zeiler. Adadelta: An adaptive learning rate method. In *arXiv preprint, arXiv:1212.5701*, 2012.

[17] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *arXiv preprint, arXiv:1404.2188*, 2014.

[18] O. Irsoy and C. Cardie. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, 2014.

[19] D. Tang, B. Qin, and T. Liu. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of ACL*, 2015.

[20] Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. Sentiment analysis of short informal texts. In *Journal of Artificial Intelligence Research*, page 723–762, 2014.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

[22] S. Wager, W. Fithian, S. Wang, and P. S. Liang, “Altitude training: Strong bounds for single-layer dropout,” in *Advances in Neural Information Processing Systems 27*, pp. 100–108, Curran Associates, Inc., 2014.

[23] Pascanu, R., Montufar, G., and Bengio, Y. On the number of response regions of deep feed forward networks with piece-wise linear activations. 2013.

[24] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuglu, P. Kuksa. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research 12:2493–2537, 2011.

[25] Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Shiyu Wu, Xuanjing Huang. Sentence Modeling with Gated Recursive Neural Network. In *Proceedings of ACL*, 2015.