Dependency Sensitive Convolutional Neural Networks for Modeling Sentences and Documents

Rui Zhang
Department of EECS
University of Michigan
Ann Arbor, MI, USA
ryanzh@umich.edu

Honglak Lee
Department of EECS
University of Michigan
Ann Arbor, MI, USA
honglak@eecs.umich.edu

Dragomir Radev
Department of EECS
and School of Information
University of Michigan
Ann Arbor, MI, USA
radev@umich.edu

Abstract

The goal of sentence and document modeling is to accurately represent the meaning of sentences and documents for various Natural Language Processing tasks. In this work, we present Dependency Sensitive Convolutional Neural Networks (DSCNN) as a general-purpose classification system for both sentences and documents. DSCNN hierarchically builds textual representations by processing pretrained word embeddings via Long Short-Term Memory networks and subsequently extracting features with convolution operators. Compared with existing recursive neural models with tree structures, DSCNN does not rely on parsers and expensive phrase labeling, and thus is not restricted to sentence-level tasks. Moreover, unlike other CNN-based models that analyze sentences locally by sliding windows, our system captures both the dependency information within each sentence and relationships across sentences in the same document. Experiment results demonstrate that our approach is achieving state-of-the-art performance on several tasks, including sentiment analysis, question type classification, and subjectivity classification.

1 Introduction

Sentence and document modeling systems are important for many Natural Language Processing (NLP) applications. The challenge for textual modeling is to capture features for different text units and to perform compositions over variable-length sequences (e.g., phrases, sentences, documents). As a traditional method, the bag-of-words model treats sentences and documents as unordered collections of words. In this way, however, the bag-of-words model fails to encode word orders and syntactic structures.

By contrast, order-sensitive models based on neural networks are becoming increasingly popular thanks to their ability to capture word order information. Many prevalent order-sensitive neural models can be categorized into two classes: Recursive models and Convolutional Neural Networks (CNN) models. Recursive models can be considered as generalizations of traditional sequence-modeling neural networks to tree structures. For example, (Socher et al., 2013) uses Recursive Neural Networks to build representations of phrases and sentences by combining neighboring constituents based on the parse tree. In their model, the composition is performed in a bottom-up way from leaf nodes of tokens until the root node of the parsing tree is reached. CNN based models, as the second category, utilize convolutional filters to extract local features (Kalchbrenner et al., 2014; Kim, 2014) over embedding matrices consisting of pretrained word vectors. Therefore, the model actually splits the sentence locally into n-grams by sliding windows.

However, despite their ability to account for word orders, order-sensitive models based on neural networks still suffer from several disadvantages. First, recursive models depend on well-performing parsers, which can be difficult for many languages or noisy domains (Iyyer et al., 2015; Ma et al., 2015). Besides, since tree-structured neural networks are vulnerable to the vanishing gradient problem (Iyyer et al., 2015), recursive models require heavy label-

Proceedings of NAACL-HLT 2016, pages 1512–1521,
San Diego, California, June 12-17, 2016.

ing on phrases to add supervisions on internal nodes. Furthermore, parsing is restricted to sentences and it is unclear how to model paragraphs and documents using recursive neural networks. In CNN models, convolutional operators process word vectors sequentially using small windows. Thus sentences are essentially treated as a bag of n-grams, and the long dependency information spanning sliding windows is lost.

These observations motivate us to construct a textual modeling architecture that captures long-term dependencies without relying on parsing for both sentence and document inputs. Specifically, we propose Dependency Sensitive Convolutional Neural Networks (DSCNN), an end-to-end classification system that hierarchically builds textual representations with only root-level labels.

DSCNN consists of a convolutional layer built on top of Long Short-Term Memory (LSTM) networks. DSCNN takes slightly different forms depending on its input. For a single sentence (Figure 1), the LSTM network processes the sequence of word embeddings to capture long-distance dependencies within the sentence. The hidden states of the LSTM are extracted to form the low-level representation, and a convolutional layer with variable-size filters and max-pooling operators follows to extract task-specific features for classification purposes. As for document modeling (Figure 2), DSCNN first applies independent LSTM networks to each subsentence. Then a second LSTM layer is added between the first LSTM layer and the convolutional layer to encode the dependency across different sentences.

We evaluate DSCNN on several sentence-level and document-level tasks including sentiment analysis, question type classification, and subjectivity classification. Experimental results demonstrate the effectiveness of our approach comparable with the state-of-the-art. In particular, our method achieves highest accuracies on MR sentiment analysis (Pang and Lee, 2005), TREC question classification (Li and Roth, 2002), and subjectivity classification task SUBJ (Pang and Lee, 2004) compared with several competitive baselines.

The remaining part of this paper is the following. Section 2 discusses related work. Section 3 presents the background including LSTM networks and convolution operators. We then describe our architectures for sentence modeling and document modeling in Section 4, and report experimental results in Section 5.

2 Related Work

The success of deep learning architectures for NLP is first based on the progress in learning distributed word representations in semantic vector space (Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014), where each word is modeled with a real-valued vector called a word embedding. In this formulation, instead of using one-hot vectors by indexing words into a vocabulary, word embeddings are learned by projecting words onto a low dimensional and dense vector space that encodes both semantic and syntactic features of words.

Given word embeddings, different models have been proposed to learn the composition of words to build up phrase and sentence representations. Most methods fall into three types: unordered models, sequence models, and Convolutional Neural Networks models.

In unordered models, textual representations are independent of the word order. Specifically, ignoring the token order in the phrase and sentence, the bag-of-words model produces the representation by averaging the constituting word embeddings (Landauer and Dumais, 1997). Besides, a neural-bag-of-words model described in (Kalchbrenner et al., 2014) adds an additional hidden layer on top of the averaged word embeddings before the softmax layer for classification purposes.

In contrast, sequence models, such as standard Recurrent Neural Networks (RNN) and Long Short-Term Memory networks, construct phrase and sentence representations in an order-sensitive way. For example, thanks to its ability to capture long-distance dependencies, LSTM has re-emerged as a popular choice for many sequence-modeling tasks, including machine translation (Bahdanau et al., 2014), image caption generation (Vinyals et al., 2014), and natural language generation (Wen et al., 2015). Besides, RNN and LSTM can be both converted to tree-structured networks by using parsing information. For example, (Socher et al., 2013) applied Recursive Neural Networks as a variant of the standard RNN structured by syntactic trees to the

sentiment analysis task. (Tai et al., 2015) also generalizes LSTM to Tree-LSTM where each LSTM unit combines information from its children units.

Recently, CNN-based models have demonstrated remarkable performances on sentence modeling and classification tasks. Leveraging convolution operators, these models can extract features from variable-length phrases corresponding to different filters. For example, DCNN in (Kalchbrenner et al., 2014) constructs hierarchical features of sentences by one-dimensional convolution and dynamic k-max pooling. (Yin and Schütze, 2015) further utilizes multichannel embeddings and unsupervised pretraining to improve classification results.

3 Preliminaries

In this section, we describe two building blocks for our system. We first discuss Long Short-Term Memory as a powerful network for modeling sequence data, and then formulate convolution and max-over-time pooling operators for the feature extraction over sequence inputs.

3.1 Long Short-Term Memory

Recurrent Neural Network (RNN) is a class of models to process arbitrary-length input sequences by recursively constructing hidden state vectors $\mathbf{h}_t$. At each time step t, the hidden state $\mathbf{h}_t$ is an affine function of the input vector $\mathbf{x}_t$ at time t and its previous hidden state $\mathbf{h}_{t-1}$, followed by a non-linearity such as the hyperbolic tangent function:

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + b) \tag{1}$$

where $\mathbf{W}$, $\mathbf{U}$ and b are parameters of the model.

However, traditional RNN suffers from the exploding or vanishing gradient problems, where the gradient vectors can grow or decay exponentially as they propagate to earlier time steps. This problem makes it difficult to train RNN to capture long-distance dependencies in a sequence (Bengio et al., 1994; Hochreiter, 1998).

To address this problem of capturing long-term relations, Long Short-Term Memory (LSTM) networks, proposed by (Hochreiter and Schmidhuber, 1997) introduce a vector of memory cells and a set of gates to control how the information flows through the network. We thus have the input gate $\mathbf{i}_t$, the forget gate $\mathbf{f}_t$, the output gate $\mathbf{o}_t$, the memory cell $\mathbf{c}_t$, the input at the current step t as $\mathbf{x}_t$, and the hidden state $\mathbf{h}_t$, which are all in $\mathbb{R}^d$. Denote the sigmoid function as σ, and the element-wise multiplication as $\odot$. At each time step t, the LSTM unit manipulates a collection of vectors described by the following equations:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}^{(i)}\mathbf{x}_t + \mathbf{U}^{(i)}\mathbf{h}_{t-1} + b^{(i)}\right) \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}^{(f)}\mathbf{x}_t + \mathbf{U}^{(f)}\mathbf{h}_{t-1} + b^{(f)}\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}^{(o)}\mathbf{x}_t + \mathbf{U}^{(o)}\mathbf{h}_{t-1} + b^{(o)}\right) \\
\mathbf{u}_t &= \tanh\left(\mathbf{W}^{(u)}\mathbf{x}_t + \mathbf{U}^{(u)}\mathbf{h}_{t-1} + b^{(u)}\right) \\
\mathbf{c}_t &= \mathbf{i}_t \odot \mathbf{u}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1} \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{2}$$

Note that the gates $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t \in [0,1]^d$ and they control at time step t how the input is updated, how much the previous memory cell is forgotten, and the exposure of the memory to form the hidden state vector respectively.

3.2 Convolution and Max-over-time Pooling

Convolution operators have been extensively used in object recognition (LeCun et al., 1998), phoneme recognition (Waibel et al., 1989), sentence modeling and classification (Kalchbrenner et al., 2014; Kim, 2014), and other traditional NLP tasks (Collobert and Weston, 2008). Given an input sentence of length s: $[w_1, w_2, ..., w_s]$, convolution operators apply a number of filters to extract local features of the sentence.

In this work, we employ one-dimensional wide convolution described in (Kalchbrenner et al., 2014). Let $\mathbf{h}_t \in \mathbb{R}^d$ denote the representation of w_t, and $\mathbf{F} \in \mathbb{R}^{d \times l}$ be a filter where l is the window size. One-dimensional wide convolution computes the feature map $\mathbf{c}$ of length $(s + l - 1)$

$$\mathbf{c} = [c_1, c_2, ..., c_{s+l-1}] \tag{3}$$

for the input sentence.

Specifically, in wide convolution, we stack $\mathbf{h}_t$ column by column, and add $(l-1)$ zero vectors to both ends of the sentence respectively. This formulates an input feature map $\mathbf{X} \in \mathbb{R}^{d \times (s+2l-2)}$. Thereafter, one-dimensional convolution applies the filter $\mathbf{F}$ to each set of consecutive l columns in $\mathbf{X}$ to produce

$(s - l - 1)$ activations. The k-th activation is produced by

$$\mathbf{c}_k = f\left(b + \sum_{i,j}\left(\mathbf{F} \odot \mathbf{X}_{k:k+l-1}\right)_{i,j}\right) \quad (4)$$

where $\mathbf{X}_{k:k+l-1} \in \mathbb{R}^{d \times l}$ is the k-th sliding window in $\mathbf{X}$, and b is the bias term. $\odot$ performs element-wise multiplications and f is an nonlinear function such as Rectified Linear Units (ReLU) or the hyperbolic tangent.

Then, the max-over-time pooling selects the maximum value in the feature map

$$c_{\mathbf{F}} = \max(\mathbf{c}) \quad (5)$$

as the feature corresponding to the filter $\mathbf{F}$.

In practice, we apply many filters with different window sizes l to capture features encoded in l-length windows of the input.

4 Model Architectures

Convolutional Neural Networks have demonstrated state-of-the-art performances in sentence modeling and classification. Despite the fact that CNN is an order-sensitive model, traditional convolution operators extract local features from each possible window of words through filters with predefined sizes. Therefore, sentences are effectively processed like a bag of n-grams, and long-distance dependencies can be only captured if we have long enough filters.

To capture long-distance dependencies, much recent effort has been dedicated to building tree-structured models from the syntactic parsing information. However, we observe that these methods suffer from three problems. First, they require an external parser and are vulnerable to parsing errors (Iyyer et al., 2015). Besides, tree-structured models need heavy supervisions to overcome vanishing gradient problems. For example, in (Socher et al., 2013), input sentences are labeled for each subphrase, and softmax layers are applied at each internal node. Finally, tree-structured models are restricted to sentence level, and cannot be generalized to model documents.

In this work, we propose a novel architecture to address these three problems. Our model hierarchically builds text representations from input words without parsing information. Only labels at the root level are required at the top softmax layer, so there is no need for labeling subphrases in the text. The system is not restricted to sentence-level inputs: the architecture can be restructured based on the sentence tokenization for modeling documents.

4.1 Sentence Modeling

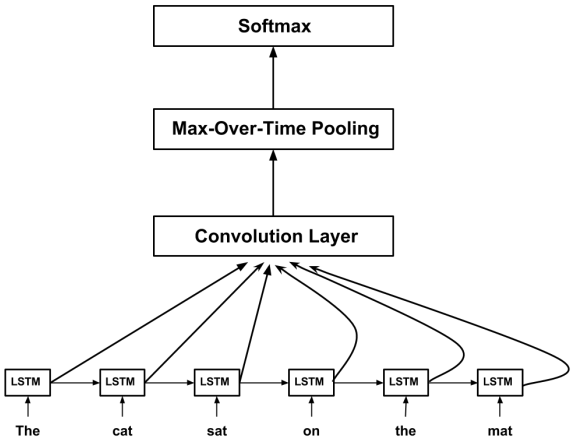


Figure 1: An example for sentence modeling. The bottom LSTM layer processes the input sentence and feed-forwards hidden state vectors at each time step. The one-dimensional wide convolution layer and the max-over-time pooling operation extract features from the LSTM output. For brevity, only one version of word embedding is illustrated in this figure.

Let the input of our model be a sentence of length s: $[w_1, w_2, ..., w_s]$, and c be the total number of word embedding versions. Different versions come from pre-trained word vectors such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014).

The first layer of our model consists of LSTM networks processing multiple versions of word embedding. For each version of word embedding, we construct an LSTM network where the input $\mathbf{x}_t \in \mathbb{R}^d$ is the d-dimensional word embedding vector for w_t. As described in the previous section, the LSTM layer will produce a hidden state representation $\mathbf{h}_t \in \mathbb{R}^d$ at each time step. We collect hidden state representations as the output of LSTM layers:

$$\mathbf{h}^{(i)} = [\mathbf{h}_1^{(i)}, \mathbf{h}_2^{(i)}, ..., \mathbf{h}_t^{(i)}, ..., \mathbf{h}_s^{(i)}] \quad (6)$$

for $i = 1, 2, ..., c$.

A convolution neural network follows as the second layer. To deal with multiple word embeddings, we use filter $\mathbf{F} \in \mathbb{R}^{c \times d \times l}$, where l is the window size. Each hidden state sequence $\mathbf{h}^{(i)}$ produced by the i-th version of word embeddings forms one channel of the feature map. These feature maps are stacked as c-channel feature maps $\mathbf{X} \in \mathbb{R}^{c \times d \times (s+2(l-1))}$.

Similar to the single channel case, activations are computed as a slight modification of equation 4:

$$\mathbf{c}_k = f\left(b + \sum_{i,j,r} \left(\mathbf{F} \odot \mathbf{X}_{k:k+l-1}\right)_{i,j,r}\right) \quad (7)$$

A max-over-time pooling layer is then added on top of the convolution neural network. Finally, the pooled features are used in a softmax layer for classification. A sentence modeling example is illustrated in Figure 1.

4.2 Document Modeling

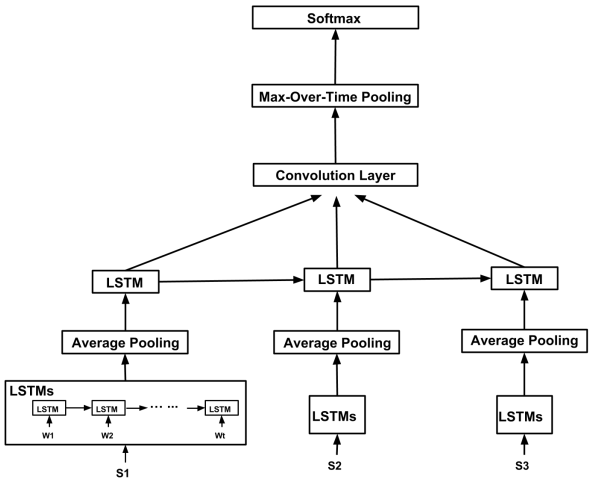


Figure 2: A schematic for document modeling hierarchy, which can be viewed as a variant of the one for sentence modeling. Independent LSTM networks process subsentences separated by punctuation. Hidden states of LSTM networks are averaged as the sentence representations, from which the high-level LSTM layer creates the joint meaning of sentences.

Our model is not restricted to sentences; it can be restructured to model documents. The intuition comes from the fact that as the composition of words builds up the semantic meaning for sentences, the composition of sentences establishes the semantic meaning for documents (Li et al., 2015).

Now suppose that the input of our model is a document consisting of n subsentences: $[s_1, s_2, ..., s_n]$. Subsentences can be obtained by splitting the document using punctuation (comma, period, question mark, and exclamation point) as delimiters.

We employ independent LSTM networks for each subsentence in the same way as the first layer of the sentence modeling architecture. For each subsentence we feed-forward the hidden states of the corresponding LSTM network to the average pooling layer. Take the first sentence of the document as an example,

$$\mathbf{h}_{s1}^{(i)} = \frac{1}{\text{len}(s1)} \sum_{j=1}^{\text{len}(s1)} \mathbf{h}_{s1,j}^{(i)} \quad (8)$$

where $\mathbf{h}_{s1,j}^{(i)}$ is the hidden state of the first sentence at time step j, and $\text{len}(s1)$ denotes the length of the first sentence. In this way, after the averaging pooling layers, we have a representation sequence consisting of averaged hidden states for subsentences,

$$\mathbf{h}^{(i)} = [\mathbf{h}_{s1}^{(i)}, \mathbf{h}_{s2}^{(i)}, ..., \mathbf{h}_{sn}^{(i)}] \quad (9)$$

for $i = 1, 2, ..., c$.

Thereafter, a high-level LSTM network comes into play to capture the joint meaning created by the sentences.

Similar as sentence modeling, a convolutional layer is placed on top of the high-level LSTM for feature extraction. Finally, a max-over-time pooling layer and a softmax layer follow to pool features and perform the classification task. Figure 2 gives the schematic for the hierarchy.

5 Experiments

5.1 Datasets

Movie Review Data (MR) proposed by (Pang and Lee, 2005) is a dataset for sentiment analysis of movie reviews. The dataset consists of 5,331 positive and 5,331 negative reviews, mostly in one sentence. We follow the practice of using 10-fold cross validation to report results.

Stanford Sentiment Treebank (SST) is another popular sentiment classification dataset introduced

Method	MR	SST-2	SST-5	TREC	SUBJ	IMDB
SVM (Socher et al., 2013)	—	79.4	40.7	—	—	—
NB (Socher et al., 2013)	—	81.8	41.0	—	—	—
NBSVM-bi (Wang and Manning, 2012)	79.4	—	—	—	93.2	91.2
SVM_S (Silva et al., 2011)	—	—	—	95.0	—	—
Standard-RNN (Socher et al., 2013)	—	82.4	43.2	—	—	—
MV-RNN (Socher et al., 2012)	79.0	82.9	44.4	—	—	—
RNTN (Socher et al., 2013)	—	85.4	45.7	—	—	—
DRNN (Irsoy and Cardie, 2014)	—	86.6	49.8	—	—	—
Standard-LSTM (Tai et al., 2015)	—	86.7	45.8	—	—	—
bi-LSTM (Tai et al., 2015)	—	86.8	49.1	—	—	—
Tree-LSTM (Tai et al., 2015)	—	88.0	51.0	—	—	—
SA-LSTM (Dai and Le, 2015)	80.7	—	—	—	—	92.8
DCNN (Kalchbrenner et al., 2014)	—	86.8	48.5	93.0	—	—
CNN-MC (Kim, 2014)	81.1	88.1	47.4	92.2	93.2	—
MVCNN (Yin and Schütze, 2015)	—	89.4	49.6	—	93.9	—
Dep-CNN (Ma et al., 2015)	81.9	—	49.5	95.4	—	—
Neural-BoW (Kalchbrenner et al., 2014)	—	80.5	42.4	88.2	—	—
DAN (Iyyer et al., 2015)	80.3	86.3	47.7	—	—	89.4
Paragraph-Vector (Le and Mikolov, 2014)	—	87.8	48.7	—	—	92.6
WRRBM+BoW(bnc) (Dahl et al., 2012)	—	—	—	—	—	89.2
Full+Unlabeled+BoW(bnc) (Maas et al., 2011)	—	—	—	—	88.2	88.9
DSCNN	81.5	89.1	49.7	95.4	93.2	90.2
DSCNN-Pretrain	82.2	88.7	50.6	95.6	93.9	90.7

Table 1: Experiment results of DSCNN compared with other models. Performance is measured in accuracy (%). Models are categorized into five classes. The first block is baseline methods including SVM and Naive Bayes and their variations. The second is the class of Recursive Neural Networks models. Constituent parsers and phrase-level supervision are needed. The third category is LSTMs. CNN models are fourth block, and the last category is a collection of other models achieving state-of-the-art results. **SVM**: Support Vector Machines with unigram features (Socher et al., 2013) **NB**: Naive Bayes with unigram features(Socher et al., 2013) **NBSVM-bi**: Naive Bayes SVM and Multinomial Naive Bayes with bigrams (Wang and Manning, 2012) **SVM_S**: SVM with features including uni-bi-trigrams, POS, parser, and 60 hand-coded rules (Silva et al., 2011) **Standard-RNN**: Standard Recursive Neural Network (Socher et al., 2013) **MV-RNN**: Matrix-Vector Recursive Neural Network (Socher et al., 2012) **RNTN**:Recursive Neural Tensor Network (Socher et al., 2013) **DRNN**: Deep Recursive Neural Network (Irsoy and Cardie, 2014) **Standard-LSTM**: Standard Long Short-Term Memory Network (Tai et al., 2015) **bi-LSTM**: Bidirectional LSTM (Tai et al., 2015) **Tree-LSTM**: Tree-Structured LSTM (Tai et al., 2015) **SA-LSTM**: Sequence Autoencoder LSTM (Dai and Le, 2015). For fair comparison, we report the result on **MR** trained without unlabeled data from IMDB or Amazon reviews. **DCNN**: Dynamic Convolutional Neural Network with k-max pooling (Kalchbrenner et al., 2014) **CNN-MC**: Convolutional Neural Network with static pretrained and fine-tuned pretrained word-embeddings (Kim, 2014) **MVCNN**: Multichannel Variable-Size Convolution Neural Network (Yin and Schütze, 2015) **Dep-CNN**: Dependency-based Convolutional Neural Network (Ma et al., 2015). Dependency parser is required. The result is for the combined model ancestor+sibling+sequential. **Neural-BoW** : Neural Bag-of-Words Models (Kalchbrenner et al., 2014) **DAN**: Deep Averaging Network (Iyyer et al., 2015) **Paragraph-Vector**: Logistic Regression on Paragraph-Vector (Le and Mikolov, 2014) **WRRBM+BoW(bnc)**: word representation Restricted Boltzmann Machine combined with bag-of-words features (Dahl et al., 2012) **Full+Unlabeled+BoW(bnc)**:word vector based model capturing both semantic and sentiment, trained on unlabeled examples, and with bag-of-words features concatenated (Maas et al., 2011)

by (Socher et al., 2013). The sentences are labeled in a fine-grained way (SST-5): {very negative, negative, neutral, positive, very positive}. The dataset has been split into 8,544 training, 1,101 validation, and 2,210 testing sentences. Without neutral sentences, SST can also be used in binary mode (SST-2), where the split is 6,920 training, 872 validation, and 1,821 testing.

Furthermore, we apply DSCNN on question type classification task on TREC dataset (Li and Roth, 2002), where sentences are questions in the following 6 classes: {abbreviation, entity, description, location, numeric}. The entire dataset consists of 5,452 training examples and 500 testing examples.

We also benchmark our system on the subjectivity classification dataset (SUBJ) released by (Pang and Lee, 2004). The dataset contains 5,000 subjective sentences and 5,000 objective sentences. We report 10-fold cross validation results as the baseline does.

For document-level dataset, we use Large Movie Review (IMDB) created by (Maas et al., 2011). There are 25,000 training and 25,000 testing examples with binary sentiment polarity labels, and 50,000 unlabeled examples. Different from Stanford Sentiment Treebank and Movie Review dataset, every example in this dataset has several sentences.

5.2 Training Details and Implementation

We use two sets of 300-dimensional pre-trained embeddings, word2vec[1] and GloVe[2], forming two channels for our network. For all datasets, we use 100 convolution filters each for window sizes of 3, 4, 5. Rectified Linear Units (ReLU) is chosen as the nonlinear function in the convolutional layer.

For regularization, before the softmax layers, we employ Dropout operation (Hinton et al., 2012) with dropout rate 0.5, and we do not perform any l_2 constraints over the parameters. We use the gradient-based optimizer Adadelta (Zeiler, 2012) to minimize cross-entropy loss between the predicted and true distributions, and the training is early stopped when the accuracy on validation set starts to drop.

As for training cost, our system processes around 4000 tokens per second on a single GTX 670 GPU. As an example, this amounts to 1 minute per epoch on the TREC dataset, converging within 50 epochs.

[1]https://code.google.com/p/word2vec/

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

5.3 Pretraining of LSTM

We experiment with two variants of parameter initialization of sentence level LSTMs. The first variant (DSCNN in Table 1) initializes the weight matrices in LSTMs as random orthogonal matrices. In the second variant (DSCNN-Pretrain in Table 1), we first train sequence autoencoders (Dai and Le, 2015) which read input sentences at the encoder and reconstruct the input at the decoder. We pretrain separately on each task based on the same train/valid/test splits. The pretrained encoders are used to be the start points of LSTM layers for later supervised classification tasks.

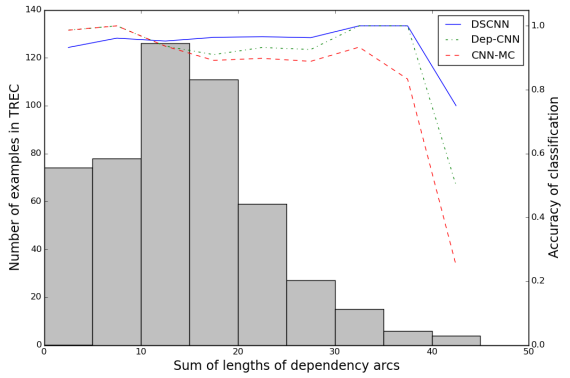


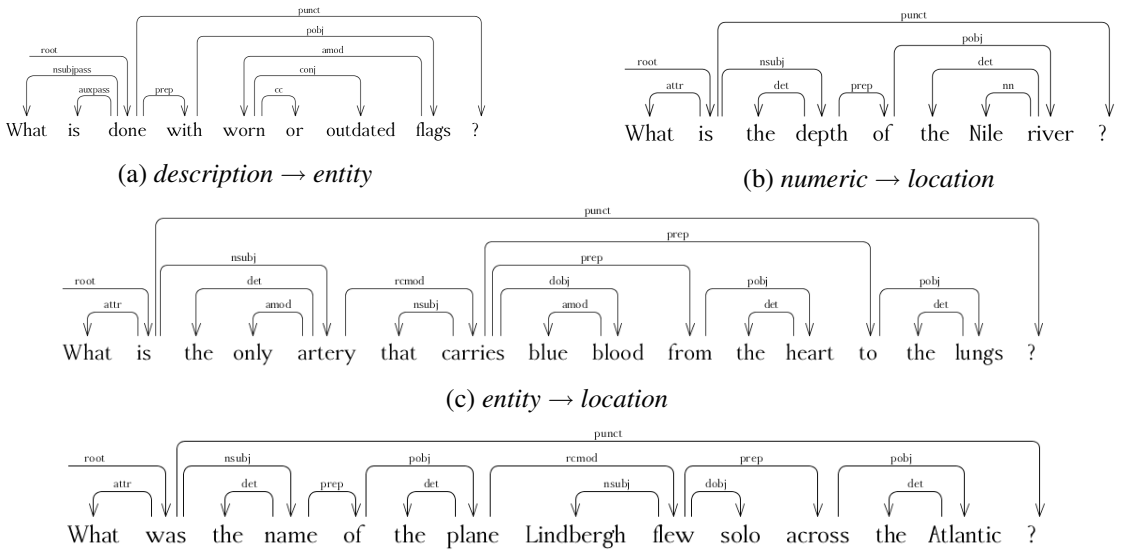
Figure 3: Number of sentences in TREC, and classification performances of DSCNN-Pretrain/Dep-CNN/CNN-MC as functions of dependency lengths. DSCNN and Dep-CNN clearly outperforms CNN-MC when the dependency length in the sentence grows.

5.4 Results and Discussions

Table 1 reports the results of DSCNN on different datasets, demonstrating its effectiveness in comparison with other state-of-the-art methods.

5.4.1 Sentence Modeling

For sentence modeling tasks, DSCNN beats all baselines on MR and TREC, and achieves the same best result on SUBJ as MVCNN. In SST-2, DSCNN only reports a slightly lower accuracy than MVCNN. In MVCNN, however, the author uses more resources including five versions of word embeddings. For SST-5, DSCNN is second only to



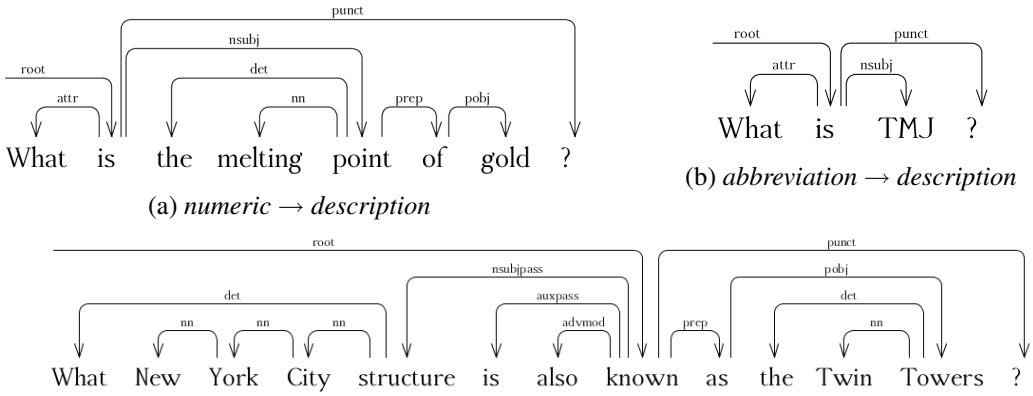
(a) *description → entity*

(b) *numeric → location*

(c) *entity → location*

(d) *entity → human*

Figure 4: TREC examples that are misclassified by CNN-MC but correctly classified by DSCNN. For example, CNN-MC labels (a) as *entity* while the ground truth is *description*. Dependency Parsing is done by ClearNLP (Choi and Palmer, 2012).



(a) *numeric → description*

(b) *abbreviation → description*

(c) *location → entity*

Figure 5: TREC examples that are misclassified by DSCNN. For example, DSCNN labels (a) as *description* while the ground truth is *numeric*. Dependency Parsing is done by ClearNLP (Choi and Palmer, 2012).

Tree-LSTM, which nonetheless relies on parsers to build tree-structured neural models.

The benefit of DSCNN is illustrated by its consistently better results over the sequential CNN models including DCNN and CNN-MC. The superiority of DSCNN is mainly attributed to its ability to maintain long-term dependencies. Figure 3 depicts the correlation between the dependency length and the classification accuracy. While CNN-MC and DSCNN are similar when the sum of dependency arc lengths is below 15, DSCNN gains obvious advantages when dependency lengths grow for long and complex sentences. Dep-CNN is also more robust than CNN-MC, but it relies on the dependency parser and predefined patterns to model longer linguistic structures.

Figure 4 gives some examples where DSCNN makes correct predictions while CNN-MC fails. In the first example, CNN-MC classifies the question as *entity* due to its focus on the noun phrase "worn or outdated flags", while DSCNN captures the long dependency between "done with" and "flags", and

assigns the correct label *description*. Similarly in the second case, due to "Nile", CNN-MC labels the question as *location*, while the dependency between "depth of" and "river" is ignored. As for the third example, the question involves a complicated and long attributive clause for the subject "artery". CNN-MC gets easily confused and predicts the type as *location* due to words "from" and "to", while DSCNN keeps correct. Finally, "Lindbergh" in the last example make CNN-MC bias to *human*.

We also sample some misclassified examples of DSCNN in Figure 5. Example (a) fails because the numeric meaning of "point" is not captured by the word embedding. Similarly, in the second example, the error is due to the out-of-vocabulary word "TMJ" and it is thus apparently difficult for DSCNN to figure out that it is an abbreviation. Example (c) is likely to be an ambiguous or mistaken annotation. The finding here agrees with the discussion in DepCNN work (Ma et al., 2015).

5.4.2 Document Modeling

For document modeling, the result of DSCNN on IMDB against other baselines is listed on the last column of Table 1. Documents in IMDB consist of several sentences and thus very long: the average length is 241 tokens per document and the maximum length is 2526 words (Dai and Le, 2015). As a result, there is no result reported using CNN-based models due to prohibited computation time, and most previous works are unordered models including variations of bag-of-words.

DSCNN outperforms bag-of-words model (Maas et al., 2011), Deep Averaging Network (Iyyer et al., 2015), and word representation Restricted Boltzmann Machine model combined with bag-of-words features (Dahl et al., 2012). The key weakness of bag-of-words prevents those models from capturing long-term dependencies.

Besides, Paragraph Vector (Le and Mikolov, 2014) and SA-LSTM (Dai and Le, 2015) achieve better results than DSCNN. It is worth mentioning that both methods, as unsupervised learning algorithms, can gain much positive effects from unlabeled data (they are using 50,000 unlabeled examples in IMDB). For example in (Dai and Le, 2015), with additional data from Amazon reviews, the error rate of SA-LSTM on MR dataset drops by 3.6%.

6 Conclusion

In this work, we present DSCNN, Dependency Sensitive Convolutional Neural Networks for purpose of text modeling at both sentence and document levels. DSCNN captures long-term inter-sentence and intra-sentence dependencies by processing word vectors through layers of LSTM networks, and extracts features by convolutional operators for classification. Experiments show that DSCNN consistently outperforms traditional CNNs, and achieves state-of-the-art results on several sentiment analysis, question type classification and subjectivity classification datasets.

Acknowledgments

We thank anonymous reviewers for their constructive comments. This work was supported by a University of Michigan EECS department fellowship and NSF CAREER grant IIS-1453651.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Jinho D Choi and Martha Palmer. 2012. Guidelines for the clear style constituent to dependency conversion. Technical report, Technical Report 01-12, University of Colorado at Boulder.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167. ACM.

George E Dahl, Ryan P Adams, and Hugo Larochelle. 2012. Training restricted boltzmann machines on word observations. *arXiv preprint arXiv:1202.5695*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. *arXiv preprint arXiv:1511.01432*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012.

Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Proceedings of NIPS, 2014*, pages 2096–2104.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL-IJCNLP*, volume 1, pages 1681–1691.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751, Doha, Qatar, October.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

X. Li and D. Roth. 2002. Learning question classifiers. In *COLING*, pages 556–562.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Proceedings of ACL-IJCNLP*, volume 2, page 174.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL-HLT*, pages 142–150, Portland, Oregon, USA, June.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of EMNLP*, 12:1532–1543.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP-CoNLL*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, volume 1631, page 1642. Citeseer.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*.

Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. 1989. Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(3):328–339.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of ACL: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Wenpeng Yin and Hinrich Schütze. 2015. Multichannel variable-size convolution for sentence classification. In *Proceedings of CoNLL*, pages 204–214, Beijing, China, July.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.