Deep LSTM based Feature Mapping for Query Classification

Yangyang Shi and **Kaisheng Yao** and **Le Tian** and **Daxin Jiang**
Microsoft
{yangshi,kaisheng.yao,letian,djiang}@microsoft.com

Abstract

Traditional convolutional neural network (CNN) based query classification uses linear feature mapping in its convolution operation. The recurrent neural network (RNN), differs from a CNN in representing word sequence with their ordering information kept explicitly. We propose using a deep long-short-term-memory (DLSTM) based feature mapping to learn feature representation for CNN. The DLSTM, which is a stack of LSTM units, has different order of feature representations at different depth of LSTM unit. The bottom LSTM unit equipped with input and output gates, extracts the first order feature representation from current word. To extract higher order nonlinear feature representation, the LSTM unit at higher position gets input from two parts. First part is the lower LSTM unit's memory cell from previous word. Second part is the lower LSTM unit's hidden output from current word. In this way, the DLSTM captures the nonlinear nonconsecutive interaction within n-grams. Using an architecture that combines a stack of the DLSTM layers with a tradition CNN layer, we have observed new state-of-the-art query classification accuracy on benchmark data sets for query classification.

1 Introduction

Convolutional neural networks (CNNs) have achieved significant improvements for query classification. CNNs capture the correlations of spatial or temporal structures with different resolutions using their temporal convolution operators. A pooling strategy on these local correlations extracts invariant regularities.

However, CNNs use simple linear operations on n-gram vectors that are formed by concatenating word vectors. The linear operation together with the concatenation may not be sufficient to model the nonconsecutive dependency and interaction within the n-grams. For example, in the query "not a total loss", nonconsecutive dependency "not loss" is the key information that is not well addressed by the linear operation with simple concatenation.

In this paper, we propose to use deep long-short-term-memory (DLSTM) based feature mapping to capture high order nonlinear feature representations. LSTM (Hochreiter and Schmidhuber, 1997) is one type of recurrent neural networks (RNNs) that have achieved remarkable performance in natural language processing and speech recognition (Sutskever et al., 2014; Graves et al., 2013).

The DLSTM is a stack of LSTM units where different order of nonlinear feature representation is captured by LSTM units at different depth. The bottom LSTM unit extracts the first order feature representation from current word. The LSTM unit at the higher position captures the higher order feature representation relying on the outputs from LSTM units at lower position, specifically, the memory cell from lower LSTM unit at previous word position and the hidden output from lower LSTM unit at current word position. Using DLSTM, linear feature mapping in traditional CNN can be obviously extended to nonlinear feature mapping. Moreover, the memory cell together with different gates in LSTM unit are able to model the nonconsecutive feature interaction and

Proceedings of NAACL-HLT 2016, pages 1501–1511,
San Diego, California, June 12-17, 2016. ©2016 Association for Computational Linguistics

information decaying based on context. For example, in the query "not so good", the proposed DLSTM is expected to keep the information of "not" and "good" in the memory, and to decay the information about "so" via the forget gates.

Similar to CNNs where multiple convolution operations are used, we propose to stack different DLSTM feature mappings together to model multiple level nonlinear feature representations. The bottom DLSTM layer takes the original word sequence as input. The DLSTM layer at lower position fed its output to the adjacent higher DLSTM layer. In the proposed models, the concatenation of the multiple level feature representations are further reduced by the pooling operation. The prediction output is finally made based on the reduced feature representations.

We evaluated the proposed method on three benchmark data sets: Standford Sentiment Treebank dataset (Socher et al., 2013), TREC (Text Retrieval Conference) question type classification data set (Li and Roth, 2002) and ATIS (Airline Travel Information Systems) dataset (Hemphill et al., 1990). On Standford Sentiment Treebank dataset, our model obtains 51.9% accuracy on fine-grained classification and 88.7% accuracy on binary classification. The SVM based method uses a large amount of engineered features, and it outperforms LSTM and RNN based methods on TREC question type classification dataset. The DLSTM outperforms other neural network based methods without using engineered features. On ATIS data, DLSTM achieves 97.9% F1 score, which is better than the previous best F1 score of 95.6% using the same data settings.

2 Related Work

Deep neural networks (Bengio, 2009; Deng and Yu, 2014; Hinton et al., 2006) dominates natural language processing (Socher, 2012; Collobert et al., 2011; Gao et al., 2014). They have achieved cutting-edge performance in various tasks such as language modeling (Mikolov et al., 2010; Sundermeyer et al., 2012), machine translation (Bahdanau et al., 2014; Cho et al., 2014; Jean et al., 2015), slot filling (Yao et al., 2014a; Shi et al., 2015a) and syntactic parsing (Wang et al., 2015; Collobert et al., 2011). For query classifications, recurrent neural networks (RNNs) and convolutional neural networks (CNNs) have emerged as top performing architectures (Zhang and Wallace, 2016; Kim, 2014; Kalchbrenner et al., 2014; Ravuri and Stolcke, 2015a).

Due to its superior ability to memorize long distance dependencies, LSTMs have been applied to extract the sentence-level continuous representation (Ravuri and Stolcke, 2015a; Tang et al., 2015; Tai et al., 2015). When the LSTM is applied to model a sentence, memory cell from the ending word in the sentence carries the information of the whole sentence. The LSTM hidden vector from the ending word is directly used as sentence feature representation in (Ravuri and Stolcke, 2015a). Alternatively, a sentence is represented by the average of LSTM hidden vectors from its words (Tang et al., 2015). Inspired from recursive neural networks (Socher et al., 2011a), LSTM is further combined with a tree structure to model sentence representation (Tai et al., 2015).

CNNs have been originally developed for image processing (Lecun et al., 1998). They are firstly applied by Collobert et al. (2008; 2011) for natural language processing tasks using max-over-time pooling method to aggregate convolution layer vectors. CNNs have also been applied to spoken language understanding (Shi et al., 2015b), information retrieval (Shen et al., 2014) and semantic parsing (Yih et al., 2015). Kalchbrenner et al. (2014) proposed to extend CNNs max-over-time pooling to k-max pooling for sentence modeling. Remarkable query classification performance on different benchmark datasets have been achieved by integrating CNNs with different feature mapping channels and pre-trained word vectors (Zhang and Wallace, 2015; Kim, 2014). Recently, Mou et al. (2015) proposed to model sentences by tree structured CNNs.

CNNs and LSTMs are complementary in their modeling capabilities; CNNs are good at capturing local invariant regularities and LSTMs are good at modeling temporal features. The combination of CNNs and LSTMs achieves improved performances in speech recognition (Sainath et al., 2015) and query classification (Tang et al., 2015; Zhou et al., 2015). In these models, the basic architecture is the LSTM that models sequence representation from local features captured by CNNs.

Different from the above methods, our method use LSTM units to model the nonlinear and non-

consecutive local features. CNNs are placed on top of these local features for query classification. Our motivation is to use LSTM replace the linear feature mapping in convolution operation where the feature mapping is a multiplication of the word vectors with a filter matrix. So our proposed model is still CNN based model but using DLSTM as feature mapping for convolution operation.

Our work is closely related to tensor product based CNNs (Lei et al., 2015) that expand CNN feature representation capacity with non-consecutive n-grams. They improve the query modeling from two aspects. Firstly, tensor products enable the nonlinear feature vector interactions between adjacent words. Secondly, an exponentially decaying weight is applied to represent non-consecutive n-gram features. Instead of using tensor products as feature mapping, we propose to apply DLSTM to address these two aspects. Nonlinear feature mapping can be achieved by the DLSTM that equipped with nonlinear activation function. The nonconsecutive feature interaction is well addressed by the memory cell and different gates in LSTM unit. In particular, the forget gate is able to decay the information according to the context rather than a fixed decaying weight in tensor product based CNNs.

3 CNN Based Query Classification Using DLSTM Feature Mapping

3.1 Linear Feature Mapping in CNN

Let k-dimensional vector $x_t \in \mathbb{R}^k$ be the continuous feature representation of the tth word in a sentence. A sentence with l words is represented by $x_{0:l-1} = [x_0; x_2; ...; x_{l-1}]$ that is a concatenation of all word vectors. The traditional CNN (Collobert et al., 2011; Kim, 2014) takes such sentence feature vector as input.

Different filters $M_j \in \mathbb{R}^{nd*h}$ are applied in convolution operation to map each n-gram feature vector $x_{t:t+n-1}, t \in (0, l-n)$ to an h-dimensional feature vector $c_{t,j}$.

$$c_{t,j} = M_j^T \cdot x_{t:t+n-1} + b_j, \tag{1}$$

where b_j is the bias in filter j.

The resulting feature vector $c_{t,j}$ are often passed through non-linear element-wise transformations (e.g. the hyperbolic tangent and rectifier linear unit) as well as pooling operations. After aggregation or reduction by different pooling operations such as the max-over-time pooling (Collobert et al., 2011; Kim, 2014) and the average pooling (Lei et al., 2015), a constant dimensional feature vector is generated for sentences with various lengths.

In traditional CNNs, the concatenated word vectors are mapped linearly to feature coordinates as shown in Equation (1). Such linear feature mapping can be improved from the following two aspects, one is to extend linear mapping to nonlinear mapping. The other one is to improve the consecutive feature mapping to nonconsecutive feature mapping. For example, in the query "not a total loss", "not loss" is the key sentiment. By using nonconsecutive feature representation, the information about "not loss" could be addressed. Lei et al. (2015) extends the linear feature mapping to tensor based feature mapping. To model the nonconsecutive n-grams, a decaying weight is applied to control the information carryover. In this paper, we propose to replace the linear feature mapping using DLSTM that captures the nonlinear and nonconsecutive feature interaction within n-grams. Rather than setting a fixed decaying weight, the proposed architecture is able to control the information decaying according to the context information.

3.2 Feature Mapping Based on Deep Long Short Term Memory

Figure 1 gives the basic architecture of a three-order nonlinear feature mapping in DLSTM. The bottom LSTM_0 extract the first order information from word input vector x_t. It is equipped with input gate and output gate. The input gate automatically controls the information saving in memory cell that will be passed to higher order LSTM unit. The output gate modifies the information from the memory cell to represent current word.

$$i_{0,t} = sigmoid(W_i x_t + b_i) \tag{2}$$
$$\tilde{c}_{0,t} = \tanh(W_c x_t + b_c) \tag{3}$$
$$c_{0,t} = i_{0,t} * \tilde{c}_{0,t} \tag{4}$$
$$o_{0,t} = sigmoid(W_o x_t + V_o c_{0,t} + b_o) \tag{5}$$
$$h_{0,t} = o_{0,t} * \tanh(c_{0,t}) \tag{6}$$

On top of the bottom LSTM_0 unit, we analogously

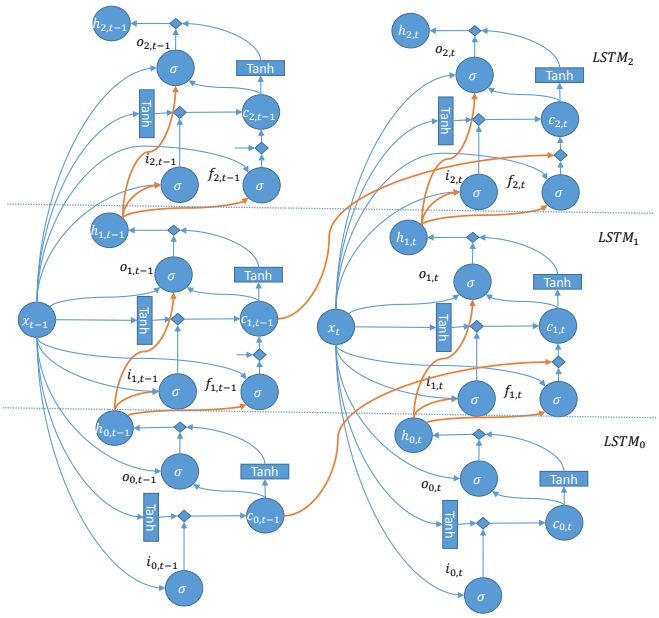

Figure 1: DLSTM based nonlinear feature mapping for bigram "$x_{t-1}x_t$". Three LSTM units are used to extract features from each word position. The bottom LSTM_0 is used for first order feature extraction from the current word. The output from the lower LSTM unit at current word position and the memory cell from lower LSTM at previous word position are fed to the higher LSTM units. Such information propagation is highlighted in the figure by bold orange lines.

stack two LSTM units LSTM_1 and LSTM_2 to extract nonlinear feature representations from bigram and trigram, respectively. The LSTM_j is formulated as follows:

$$i_{j,t} = sigmoid(W_i x_t + U_i h_{j-1,t} + b_i) \quad (7)$$

$$\tilde{c}_{j,t} = \tanh(W_c x_t + U_c h_{j-1,t} + b_c) \quad (8)$$

$$f_{j,t} = sigmoid(W_f x_t + U_f h_{j-1,t} + b_f) \quad (9)$$

$$c_{j,t} = i_{j,t} * \tilde{c}_{j,t} * c_{j-1,t-1} + f_{j,t} * c_{j-1,t} \quad (10)$$

$$o_{j,t} = sigmoid(W_o x_t + U_o h_{j-1,t} + V_o c_{j,t} + b_o) \quad (11)$$

$$h_{j,t} = o_{j,t} * \tanh(c_{j,t}) \quad (12)$$

Due to the effect from different gates that controls the information saving, expressing and decaying, LSTM_1 and LSTM_2 are able to model the non-consecutive interaction in n-grams. Take "not so good" as a example. LSTM_0 extract the nonlinear feature mapping from word "good" as $h_{0,2}$. The LSTM_1 takes $c_{0,1}$ (carries the information from word "so") and $h_{0,2}$ as input. Due to the effect of forget gate, we expect the output $h_{1,2}$ from LSTM_1 to address more on word "good" rather than "so". By further stacking LSTM_2, information about the word "not" and "good" should be emphasized by the proposed DLSTM.

Note the sum of the resulting outputs from these LSTM units is used as the high order feature representation of a n-gram ending with word x_t. So the original sequence input $x_{0:l-1}$ is mapped to a sequence of feature vector $z_{0:l-1} = [z_0, z_1, ..., z_{l-1}]$, where $z_j = h_{0,j} + h_{1,j} + h_{2,j}$.

The proposed DLSTM architecture is characterized by the following two features:

1. Weight Sharing: LSTM_1 and LSTM_2 are identical LSTM units that share the same weights. The bottom LSTM unit LSTM_0 also shares the corresponding weights with other LSTM units such as W_i, W_c and W_o. By sharing weights among different LSTM units, we can effectively reduce the risk of model over-fitting issue. At the same time, LSTM is good at capturing temporal regularities. By sharing weights, the LSTM units can learn the temporal dependencies from being exposed to different order of n-grams during the training.

2. Memory Cell Interaction: To model the nonlinear feature interaction in n-gram vectors, traditional LSTM unit is modified by Equation (10) in which the memory cell stores the interaction of different order memory cells. In this way, the feature interaction in n-grams is characterized by the memory cell interactions.

To stack the LSTM unit deeper, the depth-gated LSTM (Yao et al., 2015) and the highway network (Srivastava et al., 2015; Zhang et al., 2015) also allow the memory cell flow across LSTM units at different depth. There are three basic differences between these architectures with the proposed DLSTM. Firstly, in their architectures, LSTM units at different depth are different LSTMs that have different weight matrices. In our model, the LSTM units in DLSTM share weight matrices with each other. Secondly, in their proposed architecture, the memory cell is carried over to higher LSTM unit for facilitating model training. Because the networking training becomes more difficult with increasing model depth. In our

DLSTM, the LSTM unit at higher position takes the memory cell from lower LSTM unit mainly for feature interaction in n-grams. Finally, an additional "depth" gate is applied in their architecture to control the information flow across different layers. In our model, the input gate in higher LSTM unit controls the interaction between the memory cells extracted from previous word and current word.

3.3 The Architecture

Figure 3 gives the whole architecture of the proposed query classification system. A DLSTM layer first maps the input sequence to a sequence of high order nonlinear feature representations z^0. Instead of being directly used for query classification, the feature representation z^0 is further processed by a stack of DLSTM layers illustrated in previous section. In such stacked DLSTM layers, the output z^i of the ith DLSTM layer, is used as the input for the $i+1$th DLSTM layer parameterized by a different set of weight matrices. As shown in Figure 3, the resulting feature representations $z^0, z^1, ..., z^d$ of all these layers are concatenated. Finally, an average pooling is applied to reduce the sentence feature representation to a fixed dimensional vector that is further fed to a softmax function to obtain the prediction output.

3.4 Learning and Regularization

In the classification layer, the prediction output is obtained by the following softmax function.

$$softmax(y)_j = \frac{\exp(y_j)}{\sum_{i=1}^{i=m} \exp(y_i)}, \qquad (13)$$

where y is a m-dimensional vector. The model is trained by minimizing cross-entropy on the given training data set. To avoid overfitting during training, $L2$ regularization and dropout (Hinton et al., 2012) are used. The $L2$ regularization is applied to constrain all weight matrices using the same regularization weight. The dropout is only applied to the output of each DLSTM layer.

In the training, the model weights are updated using mini-batch stochastic gradient descent (SGD). We adapt a per-feature learning rate control method (AdaGrad) (Duchi et al., 2011) to dynamically tune the learning rate as follows:

$$\alpha_{t,i} = \frac{\alpha}{\sqrt{\sum_{j=1}^{t} g_{j,i}^2 + \varepsilon}}, \qquad (14)$$

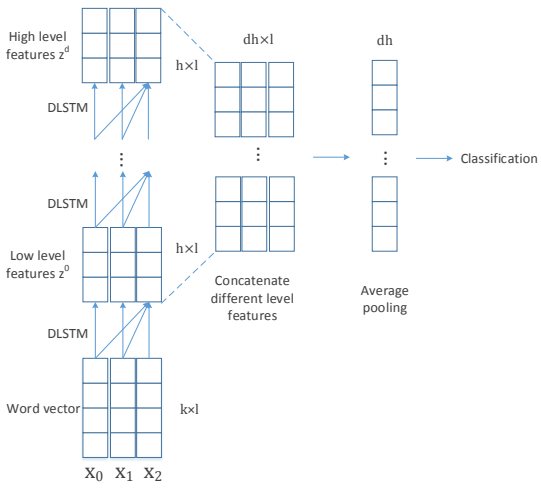

Figure 2: CNN based query classification using DLSTM feature mapping. The input sequence is represented by a $k \times l$ matrix where column t is the word vector for the tth word in the sequence. The word vectors are mapped by a stack of DLSTM layers to multi-level feature representations $z^0, ..., z^d$. As illustrated in Figure 1, each level feature representation is the sum of outputs from different LSTM units. The multi-level features are concatenated and reduced to a dh-dimensional vector where d is the number of DLSTM layers, h is the output size of each LSTM unit. A classification layer gives the prediction output.

where $\alpha_{t,i}$ is the learning rate for weight i at epoch t. $\sum_{j=1}^{t} g_{j,i}$ sums all the historical gradients of weight i. A small positive ε is applied to make the AdaGrad robust. ε is usually set to $1e-5$.

4 Experiments

4.1 Datasets

We evaluate the proposed query classification models on sentence sentiment classification, question type categorization and query intent detection tasks.

For sentence sentiment classification, the Stanford Sentiment Treebank (Socher et al., 2013) is used. In this dataset, 11855 English sentences are annotated at both sentence level and phrases level with fine-grained labels (very positive, positive, neutral, negative and very negative). We use the provided data split, which has 8544 sentences for training, 1101 sentences for developing and 2210 sentences for testing. This dataset also provides a binary classification variant that ignores the neutral

sentences. The binary classification task in this dataset has 6920 sentences for training, 872 sentences for developing and 1821 sentences for testing. There are in total 17835 unique running words for fine-grained dataset and 16185 for binary version dataset.

For query intent detection, ATIS (airline travel information system) dataset (Hemphill et al., 1990; Yao et al., 2014b) is used. This dataset is mainly about the air travel domain with 26 different intents such as "flight", "$ground_service$" and "city". There are 893 utterances for testing (ATIS-III, Nov93 and Dec94), and 4978 utterances for training (rest of ATIS-III and ATIS-II). There are 899 unique running words and 22 intents in the training data.

The question type classification task is to classify a question into a specific type, which is a very important step in question answering system. In TREC (Text Retrieval Conference) data (Li and Roth, 2002), all the questions are divided into 6 categories, including "human", "entity", "location", "description", "abbreviation" and "numeric". The dataset in total has 5952 questions, 5452 of them for training, the rest for testing. The vocabulary size of TREC dataset is 9592.

Following previous work (Iyyer et al., 2015; Tai et al., 2015; Lei et al., 2015), we used word vectors pre-trained on large unannotated corpora to achieve better generalization capability. In this paper, we used a publicly available 300 dimensional GloVe word vectors that are trained using Common Crawl with 840B tokens and 2.2M vocabulary size.

4.2 Settings

We implemented our model based on Theano library (Bastien et al., 2012). All our models are trained on Nvidia Tesla K40m.

We performed extensive hyperparameter selection based on Stanford Sentiment Treebank Binary version of validation data. The selected hyperparameters were directly used for all datasets. To investigate the robustness of the proposed method, we ran each configuration 10 times using different random initialization (random seed ranges from 1 to 10).

For final models, we set the initial learning rate to 0.1, $L2$ regularization weight to $1e-5$, the dropout probability to 0.5 and mini-batch size to 64. We use hidden layer size 256 for all the models described in the experiments. The number of the DLSTM layers and the number of the LSTM units in each DLSTM are both set to 3. So basically there are 9 LSTM units are used for each word position.

For all models, we set maximum iteration number 100 to terminate the training process. For sentiment classification task, during the training, the model with the best classification accuracy on validation data was used as final model for testing. For question type classification and query intent detection, there wasn't validation data. So we simply use the model trained at the 100th iteration as the final model for testing.

model	Fine	Binary
SVM (Lei et al., 2015)	38.3	81.3
Nbow(Lei et al., 2015)	44.5	82.0
Para-vec(Le and Mikolov, 2014)	48.7	87.8
DAN(Iyyer et al., 2015)	48.2	86.8
RAE(Socher et al., 2011b)	43.2	82.4
MVRNN(Socher et al., 2012)	44.4	82.9
RNTN(Socher et al., 2013)	45.7	85.4
DRNN(Irsoy and Cardie, 2014)	49.8	86.8
RLSTM(Tai et al., 2015)	51.0	88.0
CLSTM(Zhou et al., 2015)	49.2	87.8
DCNN(Kalchbrenner et al., 2014)	48.5	86.9
CNN-MC(Kim, 2014)	47.4	88.1
CNN-nostatic(Kim, 2014)	48.0	87.2
TCNN (Lei et al., 2015)	50.6	87.0
TCNN+phrases(Lei et al., 2015)	51.2	88.6
ours	49.2	87.2
ours+phrases	**51.9**	**88.7**

Table 1: Standford Sentiment Treebank Classification accuracy results. "Fine" denotes the accuracy on the fine-grained dataset with 5 labels. "Binary" denotes binary classification results.

4.3 Results on Stanford Sentiment Treebank

model	Acc
discriminative(Tur et al., 2010)	95.5
SVM (Shi et al., 2015b)	95.6
joint-RNN(Shi et al., 2015b)	95.2
ours	**97.9**

Table 2: ATIS intent classification accuracy comparison of different models.

Table 1 lists results for sentiment classification.

There are four blocks in the table. The bottom block gives the results from our model. The third blocks are methods related to CNNs. The second block shows the results from recursive neural network based approaches. The other baseline methods are listed in the top block.

The top block shows that the traditional methods such as SVM using ngram features and neural network using bag-of-words features (**Nbow**) perform much worse than **Para-vec** and **DAN** using word vectors that are pre-trained on large amount of unlabeled data. **Para-vec** builds a logistic regression on top of paragraph vectors. **DAN** is a deep neural network takes the average of word vectors as input.

In addition to pre-trained word vectors, syntactic compositional information can be used to improve the sentiment classification accuracy. **RAE** is a tree structured Antoencoder model based on pre-trained word vectors from Wikipedia. **MVRNN** further improves the recursive neural network by assigning each node with a matrix to learn the meaning change of neighboring words and phrases. To address large amount of different vectors and matrices involved in **MVRNN**, **RNTN** proposed to use one single tensor based function to model all nodes. By making the tree-structured recursive neural networks deeper, significant improvement has been achieved by **DRNN**. According to our knowledge, the best compositional information based model is achieved by **RLSTM** that combines LSTM unit with tree-structure.

By comparing the classification accuracy between second blocks and third blocks, we see that CNN based models in general perform better than recursive neural network based methods. Another advantage of CNN based methods is that they can be generalized to any language without dependency over compositional information. **DCNN** uses a dynamic k-max pooling operator function in CNN. To explore the task specific word vectors and the general word vectors pre-trained on large News dataset, **CNN-MC** equips CNN with two feature mapping channels. **CNN-nostatic** gives the results by only making use of general word vectors. The best published classification results are achieved by **TCNN** that is tensor based CNN.

In this paper, the proposed method is closely related to **TCNN**. Instead of using tensor products to replace linear convolution operation, our method exploits the nonlinear feature mapping through DLSTM. Rather than setting specific decaying weight to model non-consecutive n-gram features in tensor based CNN, the different gates automatically adjust the information storing, removing and outputting according to context.

Following the work of **TCNN**, to leverage the phrases level annotation in Standford Sentiment Treebank, all phrases and their corresponding labels are added to training data as additional sequences. The bottom line of Table 1 shows that our models achieved the state-of-the-art performance on sentiment classification task.

For the best settings described above, we ran each model 10 times with different random initialization. The average and standard deviation for fine-grained classification are 50.7% and 1.04%, for binary classification 88% and 0.41%. Comparing with **TCNN**, our model is more sensitive to the parameter random initialization. In the future, some efforts should be used to analyze and address this issue.

model	Acc
SVM (Silva et al., 2010)	**95.0**
Para-vec(Le and Mikolov, 2014)	91.8
AdaSent(Zhao et al., 2015)	92.4
CNN-MC(Kim, 2014)	92.2
CNN-nostatic(Kim, 2014)	93.6
DCNN(Kalchbrenner et al., 2014)	93.0
LSTM(Zhou et al., 2015)	93.2
BiLSTM(Zhou et al., 2015)	93.0
CLSTM(Zhou et al., 2015)	94.6
ours	94.8

Table 3: TREC Question type Classification accuracy comparison of different models.

4.4 Results on ATIS

ATIS dataset is widely used to test spoken language understanding system. As shown in Table 2, SVM using n-grams performs better than simple RNN and CNN based approach. **joint-RNN** is a query classification and slot filling joint training model where CNN is applied on top of slot tagging RNN for query classification. In this way, **joint-RNN** actually implicitly makes use of slot tag information for query classification. However, **joint-RNN** doesn't take ad-

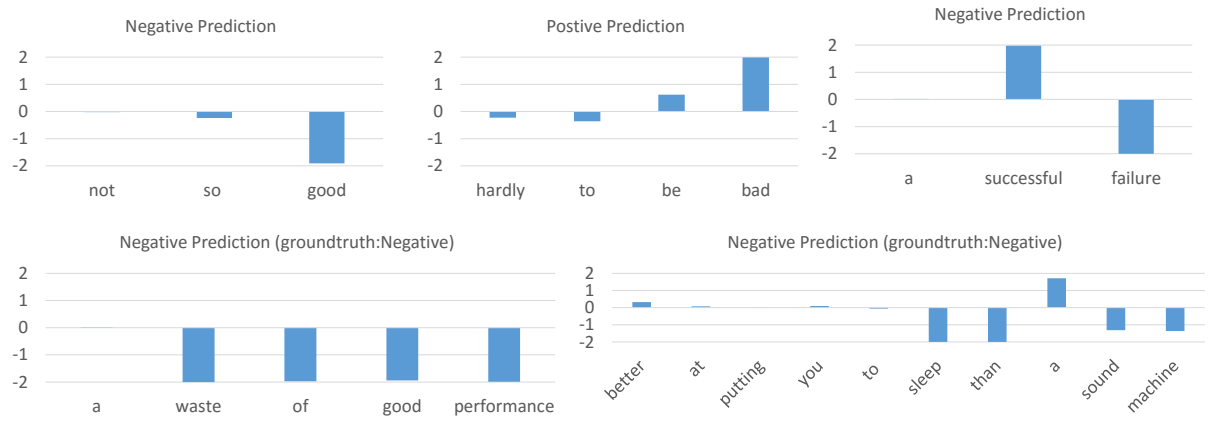

Figure 4: Example predictions given by our model trained on Stanford Sentiment Treebank fine-grained data. The expected sentiment score of each word is plotted in the figure. The score range from -2 to 2, where a score -2 means very "negative", 0 stands for "neutral" and 2 means "very positive".

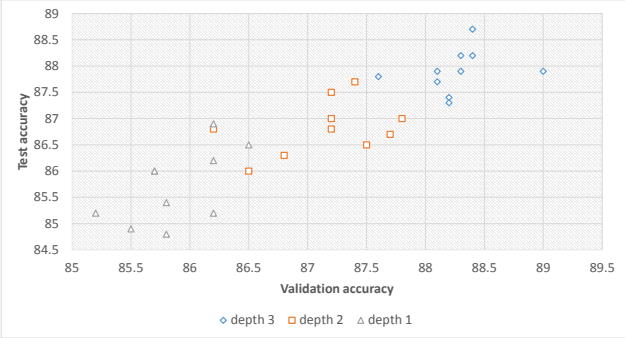

Figure 3: Stanford Sentiment Treebank Binary Classification accuracy comparison among models using the same parameter configuration except the number of DLSTM layers. For each number of DLSTM layers, 10 models are run independently using different random initialization. Horizontal axis gives the validation accuracy. Vertical axis shows the test accuracy.

vantage of word vectors trained on large amount of unlabeled data. Based on pre-trained word vectors, our models obtain more than 2% absolute classification accuracy improvement over the published best model.

In ATIS data, about 70% of queries is categorized to "flight" intent. Recent work using RNN for utterance classification (Ravuri and Stolcke, 2015b; Ravuri and Stolcke, 2015a) simplifies it to a "flight" VS "others" binary classification task. In their paper, using word based LSTM, they achieve 97.55% classification accuracy. By using extra name entity features, word based gated RNN obtains 98.42% classification accuracy.

4.5 Results on TREC Question Type Classification

Table 3 gives the TREC question type classification accuracy of our models with other baseline models. Different from the sentiment classification task, the shallow models using diverse engineered feature performs better than CNN and LSTM based models. Previous best classification results on TREC data is achieved by **SVM** using unigrams, bigrams, wh-word, head word, POS tags, hypernyms, WordNet synsets and a bunch of hand-coded rules.

AdaSent is a self adaptive hierarchical sentence model based on gating networks with level pooling. As shown in Table 3, CNN and LSTM achieve similar performances on question type classification. Recently **CLSTM** achieves substantial improvement over previous neural network based methods. In **CLSTM**, CNN is used to extract high level phrase representation. Such local segment representation is

fed into LSTM to model whole sequence representation. Different with **CLSTM** that is an LSTM based sequence model with CNN for local feature extraction, our model is CNN based model using DLSTM for non-linear feature mapping. Our model outperforms previous neural network based models without relying on task specific feature engineering.

4.6 Deep Architecture

One critical hyperparameter in the proposed method is the number of DLSTM layers. On sentiment binary classification task, we run our model 10 times by keeping all the hyperparameters the same except the number of DLSTM layers using different random initialization. As observed from Figure 3, the better performance is achieved by deeper architecture. Our model achieves the best classification result by stacking 3 DLSTM layers that actually leverages 9 different LSTM units to extract the nonlinear feature from n-grams.

4.7 Examples

Figure 4 demonstrates some examples and their sentiments predicted by our model trained on fine-grained classification data. In order to see how the nonlinear feature mapping captures the sentiment at each word position in the query, we follow the strategy used in (Lei et al., 2015) where the softmax function is directly applied on the concatenated feature mapping without passing through the average pooling layer. So the sentiment distribution p_t at tth word is computed as $p_t = W^T[z_t^0, z_t^1, ..., z_t^d]$. The expected value over the probability distribution $\sum_{s=-2}^{2} s.p_t$ is used as the sentiment score that is plotted in Figure 4. In the figure, the sentiment score ranges from -2 to 2, where -2 means very negative, 2 mean very positive and 0 means neutral.

Five examples are illustrated in the figure where the first row gives the synthetic examples to show that our model is able to model the nonconsecutive interaction within n-grams. For example, in query "hardly to be bad", even though word "hardly" is not directly modifying word "bad", our model still be able to capture such sentiment changes.

The second row of the figure shows the examples from fine-grained classification testing data. Both the example show that our model to some degree can capture sentiment of the satire. Especially the last example, our model actually gives negative prediction, even no word in the query really means negative.

5 Conclusions

We have proposed a deep long-short-term-memory (DLSTM) nonlinear nonconsecutive feature mapping architecture to replace traditional linear mapping in the convolutional neural network based query classification. Each LSTM unit in the DLSTM is responsible for capturing different order feature representation from word segments. The bottom LSTM unit equipped with input gate and output gate, extracts the nonlinear feature from unigram. The higher LSTM unit in the DLSTM takes the outputs from lower LSTM units as input. In such way, the higher LSTM unit is able to capture nonlinear feature representation from higher order n-grams. The sum of different LSTM units is used as the output of the DLSTM layer. The DLSTM output rather than being directly used as input to convolutional neural network for query classification, is passed through a stacked DLSTM layers. The query is finally represented by the concatenation of the outputs from the stacked DLSTM layers.

We evaluated the proposed models on three benchmark datasets–Stanford Sentiment Treebank dataset, TREC dataset and ATIS dataset. On both sentiment classification dataset and ATIS dataset, our model achieved the state-of-the-art performance. On TREC question type classification, SVM based model using extra engineered features still performed better than our model. But we noticed that the proposed method outperformed all the other neural network based approaches.

References

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

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. In *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2:1–127.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *The Proceedings of the International Conference on Machine Learning*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Li Deng and Dong Yu. 2014. Deep learning: Methods and applications. *Found. Trends Signal Process.*, 7:197–387.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of ACL*, pages 699–709.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *The proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The atis spoken language systems pilot corpus. In *The Proceedings of the Workshop on Speech and Natural Language*, pages 96–101.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*.

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

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

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL*, pages 1–10.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, June.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, October.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*, pages 1188–1196.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1565–1575.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*, COLING '02, pages 1–7.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *The Proceedings of Interspeech*, pages 1045–1048.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Tree-based convolution: A new neural architecture for sentence modeling. *CoRR*.

Suman Ravuri and Andreas Stolcke. 2015a. A comparative study of neural network models for lexical intent classification. In *The Proceedings of IEEE Automatic Speech Recogntion and Understanding Workshop*.

Suman Ravuri and Andreas Stolcke. 2015b. Recurrent neural network and lstm models for lexical utterance classification. In *Proceedings of Interspeech*.

Tara N. Sainath, Oriol Vinyals, Andrew W. Senior, and Hasim Sak. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *Proceedings of ICASSP*, pages 4580–4584.

Yelong Shen, Xiaodong he, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of WWW*. WWW 2014, April.

Yangyang Shi, Kaisheng Yao, Hu Chen, Yi-Cheng Pan, and Mei-Yuh Hwang. 2015a. Semi-supervised spoken language understanding using recurrent transductive support vector machines. In *Proceeding of ASRU*.

Yangyang Shi, Kaisheng Yao, Hu Chen, Yi-Cheng Pan, Mei-Yuh Hwang, and Baolin Peng. 2015b. Contextual spoken language understanding using recurrent neural networks. In *The Proceedings of International Conference on Acoustics, Speech and Signal Processing*.

J. Silva, L. Coheur, A. C. Mendes, and Andreas Wichert. 2010. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of ICML*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Richard Socher. 2012. New directions in deep learning: Structured models, tasks, and datasets. *Neural Information Processing Systems (NIPS) Workshop on Deep Learning and Unsupervised Feature Learning*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *INTERSPEECH*, pages 194–197.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566. Association for Computational Linguistics, July.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

G. Tur, D. Hakkani-Tur, and L. Heck. 2010. What is left to be understood in atis? In *Spoken Language Technology Workshop (SLT), 2010 IEEE*, pages 19–24.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional LSTM recurrent neural network with word embedding. *CoRR*.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014a. Spoken language understanding using long short-term memory neural networks. In *The Proceedings of IEEE workshop on Spoken Language Technology*, pages 189–194.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014b. Spoken language understanding using long short-term memory neural networks. In *The Proceedings of IEEE workshop on Spoken Language Technology*, pages 189–194.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated LSTM. *CoRR*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*, pages 1321–1331.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*.

Ye Zhang and Byron Wallace. 2016. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *CoRR*.

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yao, Sanjeev Khudanpur, and James Glass. 2015. Highway long short-term memory rnns for distant speech recognition. *CoRR*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *CoRR*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. 2015. A C-LSTM neural network for text classification. *CoRR*, abs/1511.08630.