A Hybrid Framework for Text Modeling with Convolutional RNN

Chenglong Wang, Feijun Jiang, Hongxia Yang
Alibaba Group
969 West Wenyi Road
Hangzhou, China 310000
{chenglong.cl,feijun.jiangfj,yang.yhx}@alibaba-inc.com

ABSTRACT

In this paper, we introduce a generic inference hybrid framework for **Conv**olutional **R**ecurrent **N**eural **N**etwork (***conv − RNN***) of semantic modeling of text, seamless integrating the merits on extracting different aspects of linguistic information from both convolutional and recurrent neural network structures and thus strengthening the semantic understanding power of the new framework. Besides, based on ***conv − RNN***, we also propose a novel sentence classification model and an attention based answer selection model with strengthening power for the sentence matching and classification respectively. We validate the proposed models on a very wide variety of data sets, including two challenging tasks of answer selection (AS) and five benchmark datasets for sentence classification (SC). To the best of our knowledge, it is by far the most complete comparison results in both AS and SC. We empirically show superior performances of ***conv − RNN*** in these different challenging tasks and benchmark datasets and also summarize insights on the performances of other state-of-the-arts methodologies.

CCS CONCEPTS

•**Computing methodologies** →**Information extraction;** *Supervised learning by classification; Neural networks;* Lexical semantics;

KEYWORDS

Text Modeling, Recurrent Neural Network, Convolution Neural Network, Answer Selection, Sentence Classification, Hybrid Framework

1 INTRODUCTION

Text modeling which represents the natural language as feature vectors, is the critical step for natural language understanding tasks such as question-answering, chat bot and user intent recognition. Traditional natural language processing (NLP) approaches for text modeling, e.g., *n*-gram, suffer from both large memory requirement and data sparsity because of language ambiguity and the limited amount of annotated data available. Distributed representations based on deep neural networks, e.g, recurrent neural network (RNN) or convolution neural network (CNN), are recently widely used to alleviate such sparsity [21, 22, 25]. Most previous neural network approaches focus on mapping each input sentence into a fixed-length vector and then performing sentence level comparisons. [5] develops an encoder-decoder architecture, which utilizes CNN and RNN as the sentence encoder and decoder respectively, to learn distributed sentence representations by reconstructing the input sentence, or predicting the future sentence. [12] trains an encoder-decoder model that tries to reconstruct the surrounding sentence of an encoded sentence using the continuity of text from books. [15] proposes an unsupervised method named as Paragraph Vector to represent input document by a dense vector which is trained to predict words in the document. [29] trains the Long Short-Term Memory (LSTM) in a weakly supervised manner on user click-through data logged by a commercial web search engine and further uses the resulted embedding vectors to perform document retrieval.

There has been much development in the areas of question-answer (QA) matching and sentence classifications (SC). [10] applies CNN on top of pre-trained word vectors for various sentence-level classification tasks. [16] uses a RNN with multi-task learning to jointly learn across multiple related text classification tasks. [13] proposes a CNN with coherent and reusable kernels that can be shared by related tasks. [35] proposes CNN for learning an optimal representation of question and answer sentences where the relational information given by the matches between words from the two members are also encoded as embeddings. [51] compares several methods including traditional lexical semantic methods and CNN based models on a new publicly available AS dataset *WikiQA*, and demonstrates the superior performance of CNN. [4] tests various architectures of CNN on a new non-factoid question answering task *InsuranceQA*.

However, it has been found that using a single vector to encode an entire sequence is not sufficient to capture all the important information from the sequence, and therefore advanced techniques such as attention mechanisms and memory networks have been applied to sequence matching problems. [34, 52] focus on the pairwise attention mechanism for discriminative model training, where it learns how to compute interactions between the input item pairs. They first build an attention matrix for a sentence pair, and then directly take the attention matrix as a new channel of the CNN model. [48] proposes to use the attention matrix in a different manner, where they decompose the original sentence matrix into a similar component matrix and a dissimilar component matrix and then feed these two matrices into a two-channel CNN model. This model focuses on characterizing the interactions between similarities and

KDD'17, August 13–17, 2017, Halifax, NS, Canada..
 ISBN 978-1-4503-4887-4/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3097983.3098140

dissimilarities of a sentence pair. Recurrent neural networks (RNNs) are powerful tools for modeling sequential data, yet training them by back-propagation through time can be difficult. [43] proposes to add the attention before computing the sentence representation for attention based RNN models. [17] shows that dependence models set up from Markov random field can be naturally extended by assigning weights to concepts and demonstrate that the dependence model can be trained using existing learning-to-rank techniques with a relatively small number of training queries. [23] proposes the key-value memory networks which are versatile models for reading documents or knowledge bases and answering questions about them, allowing to encode prior knowledge about the task at hand in the key-value memories.

1.1 Contributions

The major contributions of this paper can be summarized as follows:

(1) We propose the hybrid ***conv − RNN*** framework that can process the text using both convolutional and recurrent neural networks, seamless integrating the merits on extracting different aspects of linguistic information from both structures and thus strengthening the matching and classification power of the framework.
(2) We extend the base ***conv − RNN*** and propose novel frameworks for SC and AS respectively.
(3) We test empirically on a very wide variety of data sets, including *WikiQA*[51], *InsuranceQA*[4] and several benchmark datasets of SC including movie reviewer (MR [31]), Stanford sentiment treebank (SST [38]), IMDB [18] and Subj [30]. For AS, the proposed model outperforms the state-of-the-arts on both testing datasets; for SC, we achieve the best performances in 4 out of the 5 tasks. To the best of our knowledge, it is by far the most complete comparison results in the fields of AS and SC.

The rest of the paper is organized as follows. In Section 2, we briefly review the related work on RNN, CNN and their hybrid framework. Then in Section 3, we introduce the ***conv − RNN*** as well as the SC model and the attention based AS mdoel. Section 4 presents experimental results on extensive datasets and applications. Finally, we conclude the paper in Section 5.

2 RELATED WORK

2.1 Recurrent Neural Network (RNN)

Long short-term memory (LSTM) is a popular RNN model and has been widely applied in various NLP problems. The H dimensional hidden state h_t at the time step t is updated as follows:

$$
\begin{aligned}
i_t &= \sigma(W_i w_t + U_i h_{t-1} + b_i), && (1)\\
f_t &= \sigma(W_f w_t + U_f h_{t-1} + b_f), && (2)\\
o_t &= \sigma(W_o w_t + U_o h_{t-1} + b_o), && (3)\\
\tilde{C}_t &= \tanh(W_c w_t + U_c h_{t-1} + b_c), && (4)\\
C_t &= i_t * \tilde{C}_t + f_t * C_{t-1}, && (5)\\
h_t &= o_t * \tanh(C_t), && (6)
\end{aligned}
$$

where there are three gates, input gate i, forget gate f and output gate o, and a cell memory vector C_t. σ is the sigmoid function, $W \in \mathcal{R}^{H\times d}, U \in \mathcal{R}^{H\times H}$, and $b \in \mathcal{R}^{H\times 1}$ are the network parameters. Single directional LSTMs suffer form the weakness that they cannot utilize the contextual information from the future tokens. BI-LSTMs solve this problem by using both the previous and future context through processing the sequence in two directions, and generate two sequences of output vectors. The output for each token is the concatenation of the two vectors from both directions.

There is another popular RNN unit, namely gated recurrent unit (GRU)[1]. The GRU is capable of capturing dependencies on different time scales adaptively. Similarly to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cell. This procedure of taking a linear sum between the existing state and the newly computed state is similar to the LSTM unit. The GRU, however, does not have any mechanism to control the degree to which its state is exposed, but exposes the whole state each time. Hence it is more appropriate in our situation due to the imbalance length between questions and answers in AS. The hidden state h_t used for learning sentence representations is computed by

$$
\begin{aligned}
h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h_t}, && (7)\\
\tilde{h_t} &= \sigma(W w_t + U\,[r_t \circ h_{t-1}] + b), && (8)\\
z_t &= \sigma(W_z w_t + U_z h_{t-1} + b_z), && (9)\\
r_t &= \sigma(W_r w_t + U_r h_{t-1} + b_r), && (10)
\end{aligned}
$$

where $W, W_z, W_r \in R^{H\times d}; U, U_z, U_r \in R^{H\times H}$ and $b, b_z, b_r \in R^{H\times 1}$ are network parameters.

2.2 Convolutional Neural Network (CNN)

A CNN leverages three important ideas that can help improve a machine learning system: sparse interaction, parameter sharing and equivariant representation. *Sparse interaction* contrasts with traditional neural networks where each output is interactive with each input. In a CNN, the filter size (or kernel size) is usually much smaller than the input size. As a result , the output is only interactive with a narrow window of the input. *Parameter sharing* refers to reusing the filter parameters in the convolution operations, while the element in the weight matrix of traditional neural network will be used only once to calculate the output. *Equivariant representation* is related to the idea of k-MaxPooling which is usually combined with a CNN. So each filter of the CNN represents some feature, and after the convolution operation, the 1-MaxPooling value represents the highest degree that the input contains the feature. The position of that feature in the input is irrelevant due to the convolution. This property is very useful for many NLP applications. Below is an example to demonstrate the CNN implementation.

Assuming that $W \in R^{n\times d}$ is the input sentence matrix, with each word represented by a d-dimensional word embedding vector; $f \in R^{m\times d}$ represents the filter with sliding window size m. Then the convolutional output of the input W and the filter f is a n-dimensional vector o:

$$
o_i = \sum_{k=0}^{m-1} \sum_{j=0}^{d-1} f_{m-k-1,j} W_{i-k,j}. \qquad (11)
$$

After the k-MaxPooling, the maximum of the k values will be kept for the filter f, which indicates the k highest degree that filter f matches the input W.

There are some fundamental differences between CNN and RNN and thus can bring us different benefits. Convoluational networks can be stacked to represent large context sizes and extract hierarchical features over larger contexts with more abstractive features. On the contrary, RNN views the input as a chain structure and therefore requires a linear number $O(N)$ of operations. However, the latter is well designed for sequence modeling. Besides, in RNN, the next output depends on the previous hidden state which is not suitable for parallelization over the elements of a sequence. CNN, on the other side, is very amenable to this computing paradigm since the computation of all input words can be performed simultaneously.

2.3 Hybrid Framework

With recent advances of neural network models in natural language processing, a standard for sequence modeling now is to encode a sequence of text as an embedding vector using models such as CNN or RNN. For example, to match two sequences, a straightforward approach is to encode each sequence as a vector and then to combine the two vectors to make a decision. In a CNN, the filter size (or kernel size) is usually much smaller than the input size. As a result, the output is only interactive with a narrow window of the input and usually emphasizes the local lexical connections of the n-gram. On the other hand, RNN is well designed for sequence modeling. Especially long short-term memory (LSTM) models can successfully keep the useful information from long-range dependency but with a tradeoff of ignoring the local n-gram coherence. Fundamentally, recurrent and convolutional neural networks have their own pros and cons, and it has been found that using a vector from either CNN or RNN to encode an entire sequence is not sufficient to capture all the important information sequence [8, 9].

There have been several trials to design a hybrid framework for coherent combinations of CNNs and RNNs and enjoy the merits from both. [40] developed hybrid models that process the text using both convolutional and recurrent neural networks on extracting linguistic information from both structures to address passage AS. [6] proposed a novel neural network model based on a hybrid of ConvNet and BI-LSTMs for the semantic textual similarity measurement problem. Besides that, their pairwise word interaction model and the similarity focus layer can better capture fine-grained semantic information, compared to previous sentence modeling approaches that attempt to "cram" all sentence information into a fixed-length vector. [44] proposed an efficient hybrid model that tackles the problem which combines a fast deep model with an initial information retrieval model to effectively and efficiently handle AS.

Table 1: Notations used in *conv − RNN*

Notation	Description
w_i	the ith word in the input sentence
$\|s\|$	the length of the input sentence
S	the input sentence
V	vocabulary
$\|V\|$	the size of vocabulary
W	the word embedding matrix
d_w, d_r	the dimension of word embedding and RNN cell respectively
v_i	the word embedding of word w_i
r_t^f, r_t^b, r_t	the output of forward/backward RNN units and BI-RNN layer respectively at time step t
$h_{\|s\|}^f, h_{\|s\|}^b$	the final hidden states of forward/backward RNN units respectively
n	the number of filter vectors
f_i	the ith filter vector used in convolution layer
c_{it}	the output of convolution layer with filter vector i at time step t
A_q	the attention vector based on input question q
X_s	the final semantic representation of input sentence S

Algorithm 1 ***conv − RNN*** Algorithm

Input: The input sentence consists of a series of words: $\left[w_1, ..., w_{|s|}\right]$, where w_i is drawn from a finite-sized vocabulary V

1: Represent w_i with its corresponding word embedding $v_i \in R^{d_w}$ via a lookup table operation $v_i = LT_W(w_i)$. Define $S = \left[v_1, ..., v_{|s|}\right]$ as the input sentence embedding matrix with dimension $R^{d_w \times |s|}$.

2: Apply BI-RNN to process S to get outputs $r_t^f, r_r^b \in R^{d_r}$ and final hidden states $h_{|s|}^f, h_{|s|}^b$ of forward and backward RNN respectively at time step t. Concatenate r_t^f, r_r^b and get $r_t \in R^{2d_r}$, denote $r_t = \left[r_t^f; r_t^b\right]$.

3: Use a set of n filter vectors $f_i \in R^{2d_r}$ to process R and get C where $C_{it} = f_i^T \cdot r_t$

4: Adopt the rectified linear (ReLU) function $max(0, x)$ to process C with outputs A defined as $A_{it} = max(0, C_{it})$

5: Apply max pooling to process A and get $X_s \in R^n$ where $X_s[i] = max(0, A[i, :])$.

Output: Return X_s

3 MODEL FORMULATION

3.1 *conv − RNN*

CNN with convolutional layers and nonlinear layers followed by a pooling layer has been widely used for semantic representations in text modeling in various NLP tasks, and has proven to gain better performances than traditional NLP methods. However, CNN emphasizes the local n-gram features and could not capture long range interactions. On the other hand, RNN could efficiently keep the useful information from long-range dependency while emphasizing the local information at each time step t simultaneously. Fundamentally, recurrent and convolutional neural networks have their own pros and cons, and it has been found that using a vector from either CNN or RNN to encode an entire sequence is not sufficient to capture all the important information from the sequence. Thus, we propose the following hybrid framework for coherent combinations of CNNs and RNNs and enjoy the merits from both, namely ***conv − RNN***. Our model consists of the following four types of layers: word embedding layer, BI-RNN layer, convolutional

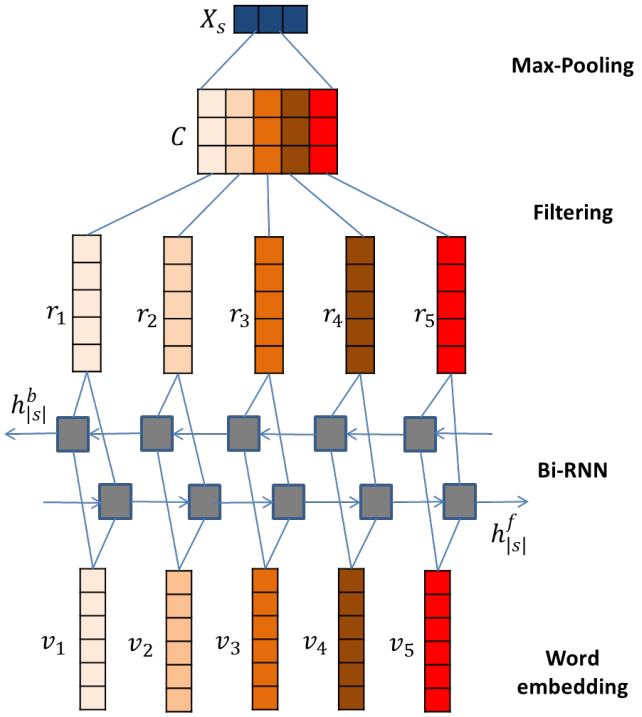

Figure 1: *conv − RNN*

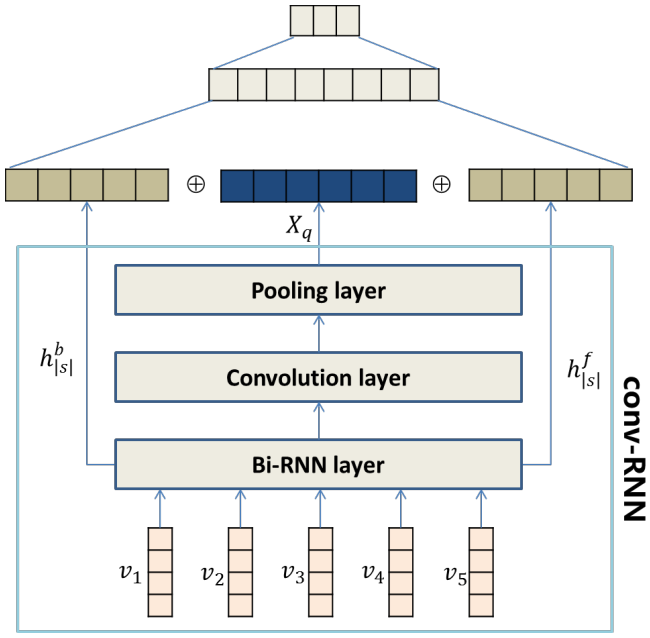

Figure 2: *conv − RNN* based sentence classification.

layer and max-pooling layer. We summarize ***conv − RNN*** in Algorithm 1 (the relevant notations are defined in Table 1) and detail as following:

Word embedding layer The original input is a sentence S consisting of a sequence of words: $[w_1, ..., w_{|s|}]$, where each word is drawn from a finite-sized vocabulary V with size $|V|$. Before fitting into the next layer, each word is transformed into a low-dimensional dense vector via a lookup table operation: $v_i = LT_W(w_i)$, where $W \in R^{d_w \times |V|}$ is the word embedding matrix and d_w is each word embedding dimension. As a result, the input sentence S is represented as a matrix where each column corresponds to a word embedding.

BI-RNN layer The sentence matrix is then fitted into the BI-RNN layer with dimension d_r. Single direction RNN is insufficient and suffers from not utilizing the contextual information from the future words. The BI-directional RNN utilizes both previous and future contexts by processing the sequence on both forward and backward directions. At each time step t, the output r_t is the concatenation of the two output vectors r^f_t and r^b_t from both directions. Besides, the final hidden states $h^f_{|s|}, h^b_{|s|}$ from both directions are usually used for representing the whole sentences. We apply attention mechanism using $h^f_{|s|}, h^b_{|s|}$ for AS.

Convolution layer Given the output of BI-RNN layer, $R \in R^{|s| \times 2d_r}$, the convolution layer uses a set of n filter vectors $f_i \in R^{2md_r}$ with sliding window size m to process it by a linear convolution operation. Formally, let $R_{i:i+j}$ refer to the concatenation of $r_i, r_{i+1}, ..., r_{i+j}$. The linear convolution operation takes the dot product of f_i with each m-gram in the sentence S shown as follows:

$$C_{it} = f_i^T \cdot R_{t-m+1:t} \tag{12}$$

In particular, the width of filter maps m in ***conv − RNN*** is set to 1, as the BI-RNN layer could already capture dependencies on different time scales adaptively. Therefore, we don't need to set a fixed size sliding window size, which might be a bottleneck of CNN [52]. The simplified convolution operation is shown as follows:

$$C_{it} = f_i^T \cdot r_t \tag{13}$$

where $C \in R^{n \times |s|}$. To enable the learning of non-linear decision boundaries, a non-linear activation function is used to process C. In this paper, we used a rectified linear (ReLU) function as the non-linear activation function.

Pooling layer Pooling, including max-pooling, min-pooling and average-pooling, is usually used to extract robust features from the results of convolution operation. In this paper, we propose to apply max-pooling for each filter to capture the most significant signal. Formally, we extract the max value from each row of C, which will generate the final representation vector $X_s \in R^n$ for the input sentence S

3.2 *conv − RNN* for Sentence Classification

In this section, we provide a simple but very effective sentence classification model as illustrated in Figure 2 based on the proposed ***conv − RNN***. The effectiveness of semantic representation

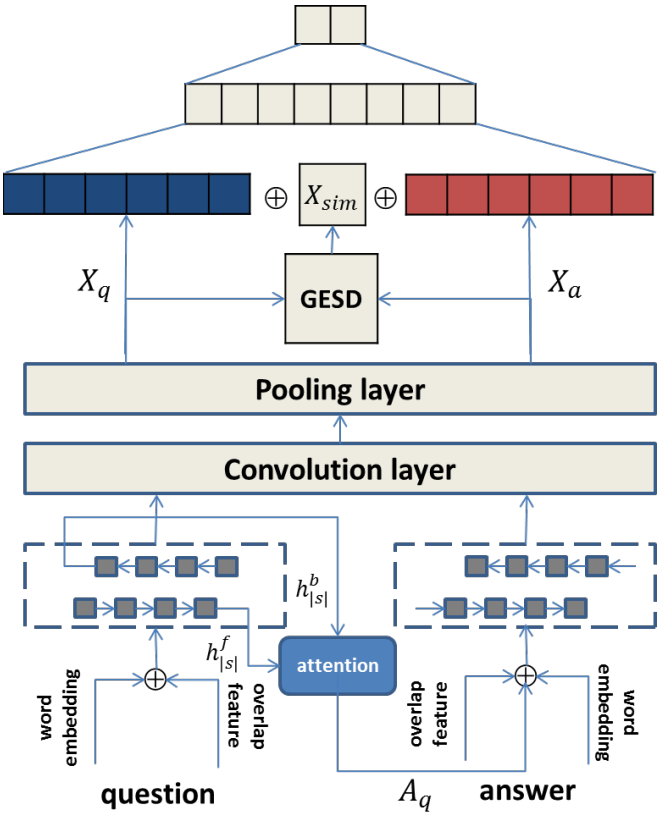

Figure 3: *conv − RNN* based question-answer matching network.

of input sentences is critical to the performance of sentence classification tasks. ***conv − RNN*** is adopted to extract the semantic information of the input texts, which are then used to predict its classes. Specifically, there is a joint layers on top of ***conv − RNN***. This joint layer concatenates the output of ***conv − RNN***, X_q, and two final hidden states from the forward and backward RNN units respectively into $X_{join} = \left[h^f_{|s|}{}', X'_q, h^b_{|s|}{}'\right]'$, which is used as the final representation of input texts. This model includes an additional hidden layer on top of the joint layer to allow for modeling interactions between the components of intermediate representations. On top of the whole model, there is a softmax classification layer, which generates a distribution over the class labels.

3.3 Attention Based *conv − RNN* for Answer Selection

We further proposed an attention based ***conv − RNN*** for AS as illustrated in Figure 3. The problem is formulated as follows: assuming that a question q is associated with a set of candidate answers $\{a_1, ..., a_n\}$ accompanied with their judgements $\{y_1, ..., y_n\}$, where $y_i = 1$ if the answer is correct and $y_i = 0$ otherwise. To better capture QA relationship, we augmented the input word embeddings with additional dimensions to represent the semantic similarity between the words of question and answer sentences. Formally, for each word w^q_i in question q, we augment its word embedding v^q_i with an overlapping score o^q_i which is the maximum inner product of v^q_i with any word embedding in the *answer*; similarly for each word w^a_i in answer a, we augment its word embedding v^a_i with an overlapping score o^a_i which is the maximum inner product of v^a_i with any word embedding in the *question*. This word matching feature was inspired by [45]. Given a word w_i, the final word representation is obtained by concatenating the original word embedding and the corresponding overlapping score.

We use separate BI-RNN layers to process the questions and answers respectively but adopt a shared convolutional layer. This is because questions and answers usually have very different structures, e.g. the lengths of answers are usually much longer than that of questions. It has shown significant improvement on performance and convergence rate by using weight-sharing layers on top of embedding layers [4]. The intuition behind this is that the corresponding elements in q and a are guaranteed to represent the same topic in a shared convolutional layer but there is no such constraint with separate layers.

In addition, we develop a simple but effective attention mechanism to improve the semantic representations for the answers based on the questions. In QA pairs, the answers might be much longer than questions and contain lots of words that are irrelevant to the questions. Hence totally separate encoding of questions and answers might result in answer sentence representations distracted by irrelevant information. We add external information from question BI-RNN encoder to the input of the ***conv − RNN*** for answer sentence encoding. In the recurrent neural networks, the final hidden states $h_{|s|}$ or the average of all hidden states $\frac{1}{|s|}\sum_{t=1}^{|s|} h_t$ is usually adopted as the question representation. In this paper, we add the final hidden states $h^f_{|s|}, h^b_{|s|}$ from forward and backward RNN units respectively to get the attention vector A_q. We apply the gated recurrent unit (GRU)[1] as the RNN unit. Formally, given A_q, the hidden state h_t used for learning answer representations is computed by

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h_t}, \quad (14)$$

$$\tilde{h_t} = \sigma(W v_t + U [r_t \circ h_{t-1}] + C A_q + b), \quad (15)$$

$$z_t = \sigma(W_z v_t + U_z h_{t-1} + C_z A_q + b_z), \quad (16)$$

$$r_t = \sigma(W_r v_t + U_r h_{t-1} + C_r A_q + b_r), \quad (17)$$

where $W, W_z, W_r \in R^{d_r \times d_w}; U, U_z, U_r \in R^{d_r \times d_r}; C, C_z, C_r \in R^{d_r \times d_r}$ and $b, b_z, b_r \in R^{d_r}$ are weight matrices. σ is non-linear activation function. This attention mechanism is designed to focus on the words in the answer sentences that are strongly connected to the questions.

Given the resulting vector representations X_q and X_a, the Geometric mean of Euclidean and Sigmoid Dot (GESD)[4] is used to measure the relatedness between the two representations:

$$X_{sim} = \frac{1}{1 + \|x - y\|} \times \frac{1}{1 + exp(-\gamma(xy^T + c))}. \quad (18)$$

It has been proved that GESD could achieve superior performance than simple cosine similarity. On top of the GESD layer and two blocks, there is a joint layer which concatenates X_q, X_a and X_{sim} into a single vector: $X_{join} = [X'_q, X'_a, X'_{sim}]'$. This vector is then

Table 2: Summary Statistics of SC Datasets.

Data	c	l	N	$\|V\|$	$\|V_{pre}\|$	Test
MR	2	20	106,62	18,765	16,448	CV
SST-1	5	53	11,855	17,836	16,262	2,210
SST-2	2	53	9,613	16,188	14,827	1,821
Subj	2	23	10,000	21,323	17,913	CV
IMDB	2	251	50,000	102,896	58962	25,000

Note: c: Number of classes. l: Average sentence length. N: Size of dataset. $|V|$: Vocabulary size. $|V_{pre}|$: Number of words present in the set of pre-trained word embeddings. Test: Test set size. CV (cross validation): No standard train/test split and thus 10 fold CV was used.

passed through two layers of full-connected neural networks, which generates a distribution over the class labels.

4 EXPERIMENTS AND EVALUATIONS

4.1 Sentence Classification

4.1.1 Experimental Datasets. We tested ***conv − RNN*** for SC (proposed in Section 3.2) on five widely used datasets summarized as following:

- MR: Short movie review dataset with one sentence per review. Each review was labeled with their overall sentiment polarity (positive or negative).
- SST-1: Stanford Sentiment Treebank 1, an extension of movie review dataset. It includes fine-grained labels (very-positive, positive, neutral, negative, very-negative) for 215,154 phrases in the parse trees of 11,855 sentences, which is more convenient for applications of recursive neural network (RecNN).
- SST-2: Similar to SST-1 but with only binary labels (positive or negative, neutral reviews removed).
- Subj: Subjectivity dataset containing sentences labeled with respect to their subjectivity status (subjective or objective).
- IMDB: A large Internet movie database for binary sentiment classification. It includes 50k full-length labeled reviews with provided training and testing splits. Besides, it provides 50k unlabeled reviews for unsupervised learning.

The samples of the first 4 tasks are short snippets with an average length less than 60. IMDB is a much larger dataset containing reviews with an average length more than 250. Summary statistics of these datasets are given in Table 2. We preprocessed the texts so that punctuations are treated as separate tokens and tokenized the text by space. We did not truncate the sentences to specific length. Besides, all the characters are converted to lower case. For comparison with other published results, the standard splits are used when they are available; for MR and Subj, 10-fold cross-validation was used for comparison.

4.1.2 Baseline Competitors. We did a very extensive comparisons with the state-of-the-arts methodologies, which can be broadly divided into the following categories:

Traditional Machine Learning (ML) [3] studied a statistical parsing framework for sentence-level sentiment classification; [46] identified that simple Naive Bayes (NB) and Support Vector Machine (SVM) variants outperform most published results on sentiment analysis datasets; [47] showed how to do fast dropout training by sampling from or integrating a Gaussian approximation, which is justified by the central limit theorem and empirical evidence, and it results in an order of magnitude speedup and more stability; [42] also showed that the dropout regularizer is first-order equivalent to an L_2 regularizer applied after scaling the features by an estimate of the inverse diagonal Fisher information matrix; [19] compared several machine learning approaches in the field of Sentiment analysis, and combined them to achieve better performance.

Deep Learning (DL) [14] extended ***word2vec*** with a new method called Paragraph-Vec, which is an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. [36–38] are various extensions of recursive networks; [26] proposed to incorporate generic and target domain embeddings in CNN for SC; [45] proposed a general "compare-aggregate" framework that performs word-level matching followed by aggregation using CNN; [10] reported on a series of experiments with CNNs trained on top of pre-trained word vectors for sentence-level classification tasks; [13] empirically studied desirable properties such as semantic coherence, attention mechanism and kernel reusability in CNNs for learning SC tasks; [2] presented two approaches that use unlabeled data to improve sequence learning with recurrent networks; [7] leveraged the Combinatory Categorial Grammar (CCG) combinatory operators to guide a non-linear transformation of meaning within a sentence; [27] proposed novel approaches that use both word embeddings created from generic and target domain corpora when it is difficult to find a domain corpus that is large enough for creating effective word embeddings.

Hybrid Framework of ML and DL [28] presented a dependency tree-based method for sentiment classification of Japanese and English subjective sentences using conditional random fields with hidden variables. [39] introduced a generalization of LSTMs to tree-structured network topologies; [16] used the multi-task learning framework to jointly learn across multiple related tasks based on recurrent neural network.

4.1.3 Experimental Setup.

Pre-trained Word Vectors For all of the SC tasks, we use the publicly pretrained ***word2vec*** [21, 22] vectors trained on part of Google News dataset (about 100 billion words). This word embedding model was trained using the continuous bag-of-words architecture and contains 300 dimensional vectors for 3 million words and phrases. For words that are not present in the set of ***word2vec***, uniform distribution was used to generate the vector representations. Preliminary experiments shown that randomly

Table 3: Hyper-parameter Configurations for Grid Search

RNN model	GRU or LSTM
Dimension of RNN unit d_r	100, 150, 200
Number of filters n	150, 200, 300
Dimension of hidden layer	200, 400, 600
weight of L_2-norm	$10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}$

generated vectors are better to have same variance as pre-trained ones. Uniform distribution between $[-0.25, 0.25]$ was used to generate random vectors for words that are not present in ***word2vec***. [35] has shown that it is better to keep the word embeddings static if the dataset is too small to fine-tune the word matrix. On the other hand, fine-tuning the word matrix along with the model may gain an improvement in final results for larger datasets. As a result, we used two sets of word embeddings, static and fine tuned. The two sets of vectors are concatenated to represent each word. During training, gradients are back-propagated only through one word matrix. Hence the model could be able to fine-tune one word matrix while keeping another static. Both word matrix are initialized with ***word2vec*** or by uniform distribution for words that are not present in ***word2vec***.

Training and Hyper-parameter settings Similarly to [10], we used a grid search on the SST-2 dev set to determine the best configurations, but there is no fine tuning for the left tasks. In particular, we tuned the hyper-parameters combinations as shown in Table 3. We carried out experiments on the hyper-parameter combinations over the whole grid. As a result, we used LSTM with 150 dimension as the RNN unit. The number of filters n in convolution layer is set to 200. The dimension of the hidden layer is 200. We also added L_2 regularisation term to the loss function, and the weight of L_2-norm is set to 10^{-3}. For MR and Subj, we also utilize dropout on the word embedding layer, BI-RNN layer and max-pooling layer respectively. The optimal dropout rate is set to 0.2, which is selected from $\{0.2, 0.4, 0.6, 0.8\}$. The dropout rate was optimized along with other hyper-parameters shown above by grid search for MR and Subj. The overall network is trained to minimise the cross-entropy of the predicted and true labels. The model is trained with mini-batches by back-propagation using Adam optimization methods [11]. Batch size is set to 16, which is tested and chosen from $\{16, 32, 64, 128\}$. The learning rate is set to 5×10^{-4}, which is optimized from $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$. Overall, we do not perform any task-specific tuning except dropout.

4.1.4 Results and Discussions. Table 4 lists the test accuracy results of our model compared to other published methods on the 5 benchmark datasets. For MR/SST-2, the best performances are achieved by CNN based models; for SST-1, LSTM based models behave better overall. Interestingly, for Subj and IMDB, simple models such as Naive Bayes/SVM with bag of word features would gain excellent performance, and none of the deep learning models could do significantly better. Specifically, the best result for IMDB

Table 4: Comparison Results of *conv − RNN* on sentence classification tasks.

Model	MR	SST-1	SST-2	Subj	IMDB
Sent-Parser [3]	79.5	-	-	-	-
NBSVM [46]	79.4	-	-	93.2	91.32
MNB [46]	79.0	-	-	93.6	86.59
G-Dropout[47]	79.0	-	-	93.4	91.2
F-Dropout [47]	79.1	-	-	93.6	91.1
Drop-Bi [42]	-	-	-	-	91.98
NB-SVM Trigram [19]	-	-	-	-	91.87
Paragraph-Vec [14]	-	48.7	87.7	-	92.58
RAE [37]	77.7	43.2	82.4	-	-
MV-RNN [36]	79.0	44.4	82.9	-	-
RNTN [38]	-	45.7	85.4	-	-
DCNN [26]	-	48.5	86.8	-	-
CNN-non-static [10]	81.5	48.0	87.2	93.4	-
CNN-multichannel [10]	81.1	47.4	88.1	93.2	-
SA-LSTM [2]	-	-	-	-	**92.76**
WkA + 25% flexible [13] filters (FF)	80.02	46.11	84.29	92.68	90.16
Fully Connected [27] Layer Combination	81.59	-	-	-	-
Tree-CRF [28]	77.3	-	-	-	-
Tree-LSTM [39]	-	50.6	86.9	-	-
Multi-Task [16]	-	49.6	87.9	94.1	91.3
CCAE [7]	77.8	-	-	-	-
conv − RNN	**81.99**	**51.67**	**88.91**	**94.13**	90.39

We use **blue** to highlight wins and use '-' to represent results that are not provided.

was achieved by SA-LSTM with additional unlabeled data. [46] explored this situation with follow up discussions[1]. We argue that IMDB is a such larger dataset containing reviews with average length more than 250 and maximum length 2,635. [24] also suggests that "statistical methods" work well for datasets with hundreds of words in each example but they cannot handle snippets with few sentences. Deep neural networks are limited in representations of large size of text, which is a worthy direction of further study. When there are sufficient contents, simple methods such as bag of words (BOW) are good enough.

As we can see, in 4 out of 5 tasks, our model with little task-specific hyper-parameter tuning exceeds all the other state-of-art methods. Notice that previous state-of-the-arts cover different domains, traditional ML, DL, and the hybrid framework of ML and DL. For MR/Subj, the number of sentences is one order of magnitude smaller than that of parameters in our model, hence regularization has a great effect on the performance. We use L_2-norm and dropout to control overfitting. Dropout has been proved to be such a powerful regularizer that enables us to use a large enough network. Consistently, dropout achieved 2%-3% relative better performances while L_2-norm could only gain slightly better results

[1] https://github.com/sidaw/nbsvm

Table 5: Summary statistics of AS dataset.

	InsuranceQA			*WikiQA*		
	Train	Dev	Test	Train	Dev	Test
#Q	12887	1000	1800*2	832	126	243
#C	50	500	500	10	9	10
#w in Q	7.2	7.2	7.2	6.5	6.5	6.4
#w in A	92.1	92.1	92.1	25.5	24.7	25.1

Note: #Q: Number of questions. #C: Average number of answers per question. #w in Q: Average number of words per question. #w in A: Average number of words per answer

(usually less than 1%). For SST-1/SST-2, the marked labels are provided at the phrase-level with more than 200k samples train models, and regularizations are not very important to avoid overfitting in this situation.

4.2 Answer Selection

4.2.1 Datasets. To test the performances of the proposed attention based $\boldsymbol{conv-RNN}$ in Section 3.3, we focus on the following two widely used benchmark datasets with summary statistics in Table 5.

- *WikiQA*: An open-domain AS dataset containing 3,047 questions originally sampled from Bing query logs. The candidate answers were extracted from the summay paragraphs of associated Wikipedia pages, with labels on whether the sentence is a correct answer to the question provided by crowdsourcing workers. 20.3% of the answers in the *WikiQA* dataset share no content words with questions and is constructed in a natural and realistic manner. We followed the same pre-processing steps as [50] and adopted the standard setup of only considering questions having correct answers for training and evaluation.
- *InsuranceQA*: A large-scale non-factoid QA dataset. All of the pairs are from the insurance domain. It provides a training set, a validation set and two test sets. For each question in test sets and dev set, there is a set of 500 candidate answers, which include the ground-truth answers and randomly selected negative answers.

4.2.2 Baseline Competitors. [51] released the *WikiQA* dataset and compared methods that achieve very competitive results. Other methods can be categorized into Information Retrieval, DNN and the recently popular Attention Based DNN as following:

Information Retrieval [23] introduced a new method, Key-Value Memory Networks, that makes reading documents more viable by utilizing different encodings in the addressing and output stages of the memory read operation; [49] presented an information retrieval approach for chatbot engines that can leverage unstructured documents, instead of Q-R pairs, to respond to utterances. [17] showed that one of the most effective existing term dependence models could be naturally extended by assigning weights to concepts and demonstrated that the weighted dependence model could be trained using existing learning-to-rank techniques, even with a relatively small number of training queries.

DNN [6] presented a hybrid deep learning network to explicitly model pairwise word interactions and present a novel similarity focus mechanism to identify important correspondences for better similarity measurement; [20] introduced a generic variational inference framework for generative and conditional models of text and validated this framework on two very different text modelling applications, generative document modelling and supervised question answering; [48] designed a model to take into account both the similarities and dissimilarities by decomposing and composing lexical semantics over sentences; [35] used the relational information given by the matches between words from the two members of the pair through CNN.

Attention Based DNN [34] proposed Attentive Pooling (AP), a two-way attention mechanism for discriminative model training; [52] presented a similar general Attention Based CNN (ABCNN) for modeling a pair of sentences. [43] analyzed the deficiency of traditional attention based RNN models quantitatively and qualitatively and presented three new RNN models that added attention information before RNN hidden representation.

4.2.3 Experimental Setup. For AS tasks, we utilize the Global Vectors for Word Representation (Glove)[32]. Specifically, We use the provided model *Common Crawl* with 300 dimensional vectors and 2.2M vocabulary to initialize the word matrix. For *InsuranceQA*, we used two sets of word embeddings during training: static and fine-tuned. These two sets of vectors are concatenated to represent the corresponding words. As for *WikiQA*, we only use the static embedding due to the reason that the *WikiQA* is an open-domain dataset and its train/dev/test sets contain separate questions from different domains. Hence there are much fewer overlapping words among train/dev/test sets. Besides, *WikiQA* is much smaller compared to *InsuranceQA*, hence fine-tuning the word matrix during training would easily leads to overfitting and has a negative effect on the final outputs.

Similarly to the set up of SC experiments, a grid search on *WikiQA* is used to determine the best configurations but no tuning for *InsuranceQA*. In particular, we tuned the same hyper-parameters shown in 4.1.3. As a result, we utilize GRU as RNN unit in Bi-RNN to encode questions and answers respectively. Questions and answers share the same convolution layer and max-pooling layer. The dimension of GRU d_r is set to 150 , and the number of filters is set to 200. The parameter of GESD, γ and c, are both set to 1.0. The cross-entropy between the predicted and true distributions is the objective function to be optimized. L_2-norm is also added to loss function for regularization and the regularizer is set to 10^{-4}. We use dropout on the Bi-LSTM layer and joint layer, and the dropout rate is set to 0.8. We use two layers of full-connected neural networks on top of X_{join} to predict the probability distribution over classes, and the hidden size is 200. Training is done via stochastic gradient descent (SGD) over shuffled mini-batches updated through Adam. The learning rate is set to 5×10^{-4} and batch size is set to 64. We use mean average precision (MAP) and mean reciprocal rank (MRR) for

Table 6: Comparison Results of *conv − RNN* on *WikiQA*.

Model	MAP	MRR
Word Cnt [51]	0.4891	0.4924
Wgt Word Cnt [51]	0.5099	0.5132
LCLR [51]	0.5993	0.6086
Key-Value Memory Network [23]	0.7069	0.7265
DocChat+(2) [49]	0.7008	0.7222
Paragraph-Vec [51]	0.5110	0.5160
CNN [51]	0.6190	0.6281
Paragraph-Vec-Cnt [51]	0.5976	0.6058
CNN-Cnt [51]	0.6520	0.6652
CubeCNN [6]	0.7090	0.7234
NASM + Cnt [20]	0.689	0.707
L.D.C [48]	0.7058	0.7226
CNNr [35]	0.6951	0.7107
IARNN-Occam(context) [43]	0.7341	0.7418
PairwiseRank+SentLevel [33]	0.701	0.718
AP-CNN [34]	0.6886	0.6957
ABCNN [52]	0.6914	0.7127
conv − RNN	**0.7427**	**0.7504**

We use **blue** to highlight wins.

the ranked set of answers to measure the performance on *WikiQA*. For *InsuranceQA*, performance is measured using top-one accuracy.

4.2.4 Results and Discussions. Table 6 and 7 summarize the results of the proposed attention based ***conv − RNN***. For *WikiQA*, it is clear that the sentence semantic models based on deep neural networks (CNN or RNN) significantly outperform traditional information retrieval methods, suggesting that semantic understanding beyond lexical semantics is important for AS tasks. Much previous work [20, 51] has demonstrated significant accuracy boosting with the result obtained from the combination of a lexical overlapping feature and the output from the deep semantic model. Results from attention based neural network models verify the effectiveness of attention mechanism for AS tasks. Our attention based ***conv − RNN*** also demonstrates its effectiveness in semantic representation in this task. Notice that the comparisons also revealed that attention is important for semantic matching of questions and answers. The proposed attention mechanism could consistently boost 1.0%-2.0% for MAP measure of *WikiQA* on average.

5 CONCLUSIONS

We propose a generic inference hybrid framework for text modeling, namely ***conv − RNN***, which seamlessly integrates the merits from both CNN and RNN. Besides, based on ***conv − RNN***, we also propose a novel sentence classification model and an attention based answer selection model, both of which utilize the effectiveness of ***conv − RNN*** on semantic understanding to strengthen the sentence classification and matching power respectively. We test empirically on a very wide variety of datasets on sentence classification and answer selection and empirically demonstrate the effectiveness of ***conv − RNN***.

Table 7: Comparison Results of *conv − RNN* on *InsuranceQA*.

Model	dev	test1	test2
IR model [17]	52.7	55.1	50.8
QA-LSTM with attention [41]	68.4	68.1	62.2
CNN with GESD [4]	65.4	65.3	61.0
Attentive LSTM [40]	68.9	69.0	64.8
IARNN-Occam [43]	69.1	68.9	65.1
IARNN-Gate [43]	70.0	70.1	62.8
AP-BILSTM [34]	68.4	**71.7**	66.4
conv − RNN	**71.7**	71.4	**68.3**

REFERENCES

[1] K. Cho, B.V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *arXiv preprint arXiv:1406.1078*.

[2] A M Dai and Q V. Le. 2015. Semi-Supervised Sequence Learning. In *Advances in Neural Information Processing Systems*. 3079–3087.

[3] L. Dong, F. Wei, S. Liu, M. Zhou, and K. Xu. 2015. A Statistical Parsing Framework for Sentiment Classification. *Computational Linguistics* 41, 2 (2015), 293–336.

[4] M. Feng, B. Xiang, M.R. Glass, L. Wang, and B. Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

[5] Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2016. Unsupervised Learning of Sentence Representations using Convolutional Neural Networks. *arXiv preprint arXiv:1611.07897* (2016).

[6] H. He and J. Lin. 2016. Pairwise Word Interaction Modeling with Deep Neural Networks for Semantic Similarity Measurement. In *Proceedings of NAACL-HLT*.

[7] K.M. Hermann and P. Blunsom. 2013. The Role of Syntax in Vector Space Models of Compositional Semantics. In *The 51st Annual Meeting of the Association for Computational Linguistics*.

[8] K.M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 1693–1701.

[9] F. Hill, A. Bordes, S. Chopra, and J. Weston. 2016. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. In *Proceedings of the International Conference on Learning Representations*.

[10] Y. Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1746–1751.

[11] D.P. Kingma and J. Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* (2014).

[12] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. 3294–3302.

[13] M. Lakshmana, S. Sellamanickam, S. Shevade, and K. Selvaraj. 2016. Learning Semantically Coherent and Reusable Kernels in Convolution Neural Nets for Sentence Classification. In *arXiv:1608.00466*.

[14] Q.V. Le and T. Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning*.

[15] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents.. In *ICML*, Vol. 14. 1188–1196.

[16] P. Liu, X. Qiu, and X. Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *arXiv:1605.05101*.

[17] BenderskyTh M., D. Metzler, and B.C. Croft. 2010. Learning concept importance using a weighted dependence model. *Proceedings of the third ACM international conference on Web Search and Data Mining (WSDM)* (2010).

[18] A. L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, and C. Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. 142–150.

[19] M. Mesnil, T. Mikolov, M. Ranzato, and Y. Bengio. 2015. Ensemble of Generative and Discriminative Techniques for Sentiment Analysis of Movie Reviews. In *Accepted as a workshop contribution at ICLR 2015*.

[20] Y. Miao, L. Yu, and P. Blunsom. 2016. Neural Variational Inference for Text Processing. In *Proceedings of the 33rd International Conference on Machine Learning*.

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

[22] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.). Curran Associates, Inc., 3111–3119.

[23] A. Miller, A. Fisch, J. Dodge, A. H. Karimi, A. Bordes, and J. Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *arXiv:1602.03126*.

[24] K. Moilanen and S. Pulman. 2007. Sentiment Composition. In *In Proceedings of RANLP*. 378–382.

[25] R.J. Mooney. 2014. Semantic parsing: Past, present, and future. In *In Association for Computational Linguistics (ACL) Workshop on Semantic Parsing*.

[26] Kalchbrenner N., Grefenstette E., and Blunsom P. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. 655–665.

[27] Limsopatham N. and N. Collier. 2016. Modelling the combination of generic and target domain embeddings in a convolutional neural network for sentence classification. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*. 103–112.

[28] T. Nakagawa, K. Inui, and S. Kurohashi. 2010. Dependency Tree-based Sentiment Classification Using CRFs with Hidden Variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 786–794.

[29] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24, 4 (2016), 694–707.

[30] B. Pang and L. Lee. 2004. A Sentimental Education: Sentiments Analysis using Subjectivity Summarization based on Minimum Cuts. In *Proceedings ACL*.

[31] B. Pang and L. Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of ACL*. 115–124.

[32] J. Pennington, R. Socher, and C.D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[33] J. Rao, H. He, and J. Lin. 2016. Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks. In *Proceedings CIKM' 16*.

[34] C. Santos, M. Tan, B. Xiang, and B. Zhou. 2016. Attentive Pooling Networks. In *arXiv:1602.03609*.

[35] A. Severyn and A. Moschitti. 2016. Modeling Relational Information in Question-Answer Pairs with Convolutional Neural Networks. In *arXiv:1602.01178*.

[36] R. Socher, B. Huval, C.D. Manning, and A.Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-vector Spaces. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 1201–1211.

[37] R. Socher, J. Pennington, E.H. Huang, A.Y. Ng, and C.D. Manning. 2011. Semi-supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 151–161.

[38] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C.D. Manning, A.Y. Ng, and C. Potts. 2013. Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1631–1642.

[39] K.S Tai, R. Socher, and C.D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. 1556fi?!–1566.

[40] M. Tan, C. Santos, B. Xiang, and B. Zhou. 2016. Improved Representation Learning for Question Answer Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 464–473.

[41] M. Tan, B. Xiang, and B. Zhou. 2015. LSTM-based Deep Learning Models for Non-factoid Answer Selection. In *arXiv preprint arXiv:1511.04108*.

[42] S. Wager, S. Wang, and P.S. Liang. 2013. Dropout training as adaptive regularization. In *Advances in neural information processing systems*. 351–359.

[43] B. Wang, K. Liu, and J. Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *The Annual Meeting of the Association for Computational Linguistics*.

[44] L. Wang, M. Tan, and J. Han. 2016. FastHybrid: A Hybrid Model for Efficient Answer Selection. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2378–2388.

[45] S. Wang and J. Jiang. 2016. A Compare-Aggregate Model for Matching Text Sequences. *CoRR* abs/1611.01747 (2016).

[46] S. Wang and C.D. Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. 90–94.

[47] S.I. Wang and C.D. Manning. 2013. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning*.

[48] Z. Wang, H. Mi, and A. Ittycheriah. 2016. Sentence Similarity Learning by Lexical Decomposition and Composition. In *arXiv:1602.07019*.

[49] Z. Yan, N. Duan, J. Bao, P. Chen, M. Zhou, Z. Li, and J. Zhou. 2016. DocChat: an information retrieval approach for chatbot engines using unstructured documents. In *The Annual Meeting of the Association for Computational Linguistics*.

[50] Y. Yang, W. Yih, and Meek C. 2015. A Challenge Dataset for Open-Domain Question Answering.

[51] Y. Yang, W. Yih, and C. Meek. 2015. WikiQA: A Challenge Dataset for Open-domain Question Answering. In *Proceedings of EMNLP*. 2013–2018.

[52] W. Yin, H. Schutze, B. Xiang, and B. Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. In *Transactions of the Association for Computational Linguistics*, Vol. 4. 259–272.