Received February 12, 2018, accepted March 6, 2018, date of publication March 12, 2018, date of current version March 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2814818

Convolutional Recurrent Deep Learning Model for Sentence Classification

ABDALRAOUF HASSAN, (Member, IEEE), AND AUSIF MAHMOOD, (Senior Member, IEEE)
Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT 06604, USA

Corresponding author: Abdalraouf Hassan (abdalrah@my.bridgeport.edu)

ABSTRACT As the amount of unstructured text data that humanity produces overall and on the Internet grows, so does the need to intelligently to process it and extract different types of knowledge from it. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been applied to natural language processing systems with comparative, remarkable results. The CNN is a noble approach to extract higher level features that are invariant to local translation. However, it requires stacking multiple convolutional layers in order to capture long-term dependencies, due to the locality of the convolutional and pooling layers. In this paper, we describe a joint CNN and RNN framework to overcome this problem. Briefly, we use an unsupervised neural language model to train initial word embeddings that are further tuned by our deep learning network, then, the pre-trained parameters of the network are used to initialize the model. At a final stage, the proposed framework combines former information with a set of feature maps learned by a convolutional layer with long-term dependencies learned via long-short-term memory. Empirically, we show that our approach, with slight hyperparameter tuning and static vectors, achieves outstanding results on multiple sentiment analysis benchmarks. Our approach outperforms several existing approaches in term of accuracy; our results are also competitive with the state-of-the-art results on the Stanford Large Movie Review data set with 93.3% accuracy, and the Stanford Sentiment Treebank data set with 48.8% fine-grained and 89.2% binary accuracy, respectively. Our approach has a significant role in reducing the number of parameters and constructing the convolutional layer followed by the recurrent layer as a substitute for the pooling layer. Our results show that we were able to reduce the loss of detailed, local information and capture long-term dependencies with an efficient framework that has fewer parameters and a high level of performance.

INDEX TERMS Convolutional neural network, recurrent neural network, natural language processing, deep learning, sentiment analysis, long-term dependencies.

I. INTRODUCTION

Natural Language Processing (NLP) is a vast area of computer science that is concerned with the interaction between computers and human language. Language modeling is a fundamental task in artificial intelligence and NLP. A language model is formalized as a probability distribution over a sequence of words. Recently, deep learning models have achieved remarkable results in speech recognition [1] and computer vision [2]. Text classification plays an important role in many NLP applications, such as spam filtering, email categorization, information retrieval, web search, and ranking and document classification [3], [4], in which one needs to assign predefined categories to a sequence of text. A popular and common method to represent texts is bag-of-words. However, the bag-of-words method loses the words order and ignores the semantics of words. N-gram models are popular for statistical language modeling and usually perform the best [5]. However, an n-gram model suffers from data sparsity [6].

Neural Networks have become increasingly popular [7]; it has become possible to train more complex models on a much larger dataset. They outperform n-gram models and overcome the data sparsity problem [6]; semantically similar words are close in vector space. The embedding of rare words is poorly estimated, which leads to higher perplexities for rare words. With the progress of machine learning in recent years, it has become possible to train more complex models on much larger data sets [1], [2], [7]–[9]. The distributed representation of words is one of the most successful concepts, and it helps learning algorithms achieve better performance [7].

Convolutional Neural Networks (CNN) [10] recently achieved very successful results in computer vision [2]. A CNN considers feature extraction and classification as one joint task. This idea has been improved by stacking multiple convolutional and pooling layers, which sequentially extract a hierarchical representation of the input [10]–[12].

We investigate Recurrent Neural Networks (RNNs) as an alternative for pooling layers in deep neural network language models to perform a sentiment analysis of a short text. Most of the deep learning architectures for NLP require stacking many layers to capture long-term dependences due to the locality of the convolutional and pooling layers [13]. Our architecture was inspired by the recent success of RNNs in NLP applications and the fact that RNNs can capture long-term dependencies even with one single layer [14]. We were also inspired by the successful work proposed in [9], where a single layer of CNN was applied for sentence classification.

It turns out that providing the network with good initialization parameters can have a significant impact on the accuracy of the trained model and capturing the long-term dependencies more efficiently. In this paper, we present a joint CNN and RNN architecture that takes the local features extracted by a CNN as the input for an RNN for a sentiment analysis of short texts. We propose a new framework that exploits and combines convolutional and recurrent layers into one single model on top of pre-trained word vectors. We utilize long short-term memory (LSTM) as a substitute for pooling layers in order to reduce the loss of detailed, local information and capture long-term dependencies across the input sequence. Our contributions are summarized below:

1. Word embeddings are initialized using a neural language model [7], [8], which is trained on a large, unsupervised collection of words.
2. We use a convolutional neural network to further refine the embeddings on a distance-supervised dataset. We take the word embedding as the input to our model in which windows of different length and various weight matrices are applied to generate a number of feature maps.
3. The word embeddings and other parameters of the network obtained at the previous stage are used to initialize the same framework.
4. The deep learning framework takes advantage of the encoded local features extracted from the CNN model and the long-term dependencies captured by the RNN model. Empirical results demonstrated that our framework achieves competitive results with fewer parameters.

The rest of the paper is organized as follows. Section II presents related works. Section III introduces background. Section IV highlights the research problem and motivation. Section V describes in detail our model architecture. Section VI outlines the experimental setup, and Section VII discusses the empirical results and analysis. Finally, Section VIII presents the conclusion.

II. RELATED WORK

A. TRADITIONAL METHODS

Text classification is significant for NLP systems, where there has been an enormous amount of research on sentence classification tasks, specifically on sentiment analysis. NLP systems classically treat words as discrete, atomic symbols where the model leverages a small amount of information regarding the relationship between the individual symbols.

A simple and efficient baseline method for a sentence structure is to represent the sentence as a bag-of-words and then train a linear classifier (e.g., a logistic regression). However, the bag-of-words approach omits all of the information about the semantics and ordering of words [15], [16]. N-gram models are another popular method to represent a sentence. This method usually performs the best [5]. Words are projected to a high-dimensional space, and then the embedding is combined to obtain a fixed-size representation of the input sentence, which later is used as an input to the classifier. Despite the fact that n-gram models take into account word ordering in short sentences, they do still suffer from data sparsity. Overall, all simple techniques have limitations for certain tasks. Furthermore, linear classifiers do not share parameters among features and classes that might limit their generalization in the context of a large output, where some classes have few examples. A popular solution for this problem is to use multilayer neural networks [13], [16], or to factorize the linear classifier into low-rank matrices [8].

B. DEEP LEARNING METHODS

Deep Neural Networks (DNNs) have achieved significant results in computer vision [2], [17] and speech recognition [1]. Recently, it has become more common to use DNNs in NLP applications, where much of the work involves learning word representations through neural language models [6]–[9] and then performing a composition over the learned word vectors for classification. These approaches have led to new methods for solving the data sparsity problem. Consequently, several neural network-based methods for learning word representations followed these approaches.

DNNs jointly implement feature extraction and classification for text classification. DNN-based approaches usually start with an input text, represented as a sequence of words, where each sequence is represented as one-hot vector; then, each word in the sequence is projected into a continuous vector space. This happens by multiplying it with a weight matrix, which leads to the creation of a sequence of dense, actual, valued vectors. This sequence then feeds into a DNN, which processes the sequence in multiple layers, resulting in prediction probability. This pipeline is tuned jointly to maximize the classification accuracy on the training sets [7]–[9], [12], [13], [17], [18]. However, one-hot vector makes no assumption about the similarity of words, and it is also very high-dimensional [9], [18].

RNNs improve time complexity and analyze texts word-by-word, then preserve the semantic of all of the previous text in a fixed-sized hidden layer [19]. The capability to capture superior, appropriate statistics could be valuable to capture the semantics of a long text in an RNN. However, an RNN is a biased model; recent words are more significant than earlier words. Therefore, they key components could appear anywhere across the document, not only at the end. This might reduce the efficiency when used to capture the semantics of a whole document. Therefore, the long short-term memory (LSTM) model was introduced to overcome the difficulties of the RNN [20].

A standard RNN makes predictions based only on considering the past words for a specific task. This technique is suitable for predicting the next word in context. However, for some tasks, it would be efficient if we could use both past and future words in tagging a task, as part-of-speech tagging, where we need to assign a tag to each word in a sentence [21]. In this case we already know the sequence of the words, and for each word we want to take both words to the left (past) and to the right (future) into consideration when making a prediction. That is exactly what the Bidirectional Neural Network (BNN) does; it consists of two LSTMs. One runs forward from left to right, and the other runs backward from right to left. This technique is successful in tagging tasks and for embedding a sequence into a fixed-length vector [18].

Convolutional Neural Networks (CNNs) were initially designed for computer vision [2], [10]. CNNs exploit layers with convolving filters that are applied to local features. CNNs reached outstanding results in computer vision where handcrafted features were used, e.g. scale-invariant features transform (SIFT) followed by a classifier; the main idea is to consider feature extractors and classifiers as one jointly trained task [9], [17]. The use of neural networks inspired many researchers after the successful approaches in [6], [16], and [17]. This area has been investigated in recent years, especially by using multi-convolutional and pooling layers in CNNs and then sequentially extracting hierarchical representations of the input.

CNN models for NLP achieved excellent results in semantic parsing [22], sentence modeling [11], search query retrieval [23], and other NLP tasks [17]. Recently, the DNN-based model has shown very good results for several tasks in NLP [9], [12], [13], [18]. Despite the good performance of these models, in practice they are relatively slow at training and testing, which restrains them from using a large scale of data, and it requires stacking many convolutional layers in order to capture long-term dependencies.

The combination of both CNNs and RNNs is explored for speech recognition [24], and a similar approach was applied to image classification [2]. [18] Investigated the combination of CNN-RNN to encode character input, and implemented a high-level feature input sequence of character level to capture sub-word information. However, this model performs best when a large number of classes are available. Reference [25] Outlined structured attention networks, which incorporate graphical models to generalize simple attention, describe the technical machinery and computational techniques for backpropagation through models of this form. Referenec [26] aimed to improve representation efficiency, and the model employed Differential State Framework (DSF). DSF models maintain longer-term memory by learning to interpolate between a fast-changing, data-driven representation and a slowly changing, implicitly stable state. Reference [27] Investigated an approach to advance the accuracy of the deep learning method for sentiment analysis by incorporating domain knowledge. This paper combined domain knowledge with deep learning, using sentiment scores learnt by regression to augment the training data. They also utilized weighting across entropy with a penalty matrix as an enhanced loss function.

We observed that the use of a vanilla CNN for text classification has one drawback. In [18] the network must have many layers in order to capture long-term dependencies in an input sentence. Perhaps that might be the motivation behind [12], which utilized a very deep convolutional network with six convolutional layers followed by two fully connected layers.

III. BACKGROUND

A. CONVOLUTIONAL NEURAL NETWORKS

Recently CNNs were applied to NLP systems and accomplished very interesting results [9], [13], [18]; convolutional layers are similar to a sliding window over a matrix. CNNs are numerous layers of convolutions with nonlinear activation functions, such as ReLU or tanh, applied to the results. In a classical, feed-forward neural network, each input of a neuron is attached to each output in the next layer. This is called a fully connected or affine layer. However, CNNs have different approaches where they use convolutions over the input layer to compute the output. Local connections compute the output over the input layer, and then each layer applies different kernels, usually hundreds or thousands of filters, to then combine their results.

During pooling or subsampling layers and during the training stage, CNNs learn the values of their filter size based on the tasks. For instance, in image classification [2] a CNN might learn to detect edges from raw pixels in the first layer, then use the edges to detect simple shapes in the second layer, and then use these shapes to detect higher-level features, such as facial shapes, in higher layers. The layer is then fed to a classifier that uses these high-level features. However, how does this apply to NLP?

As an alternative to image pixels, the input to most NLP tasks consists of sentences and documents represented as a matrix. Additionally, each row of the matrix matches up to one token, usually a word or character. Each row is a vector that represents a word. Typically, this vector is a word-embedded, low-dimensional representation (e.g. word2vec, one-hot vectors) that indexes the word into a vocabulary (e.g. a ten word sentence using a 100-dimensional embedding, 10*100 matrix) as our input. In NLP, a filter slides over full words of the matrix. Therefore, the width of the filters is same as the width of the input matrix. Moreover, the region size may vary, but it is usually a sliding window over two to five words at a time.

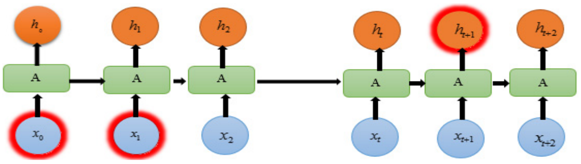

FIGURE 1. Recurrent Neural Network with Long-Term Dependences.

B. RECURRENT NEURAL NETWORKS

The intuition of RNNs is that humans do not start their thinking from scratch every second. The objective of an RNN is to make use of sequential information. The output is based on the previous computation. In traditional Neural Networks, all inputs are independent of each other. While this approach is inefficient for many tasks in NLP (e.g. predicting the next word in a sentence), in this case it is important to know the previous word in order to predict the next word in context. RNNs have shown great success in many NLP tasks [1], [14], [20], [21], [28]. RNNs have a memory which captures information in arbitrary, long sequences.

RNNs are deep neural networks that are deep in temporal dimension and have been used widely in time sequence modeling. The objective behind RNNs for sentence embedding is to find a dense and low-dimensional semantic representation by recurrently and sequentially processing each word in a sentence and mapping it into a low-dimensional vector. The global contextual feature of the whole text will be in the semantic representation of the last word in the sequence [1], [29], [30]. We also can think of RNNs as multiple copies of the same network, where each one is passing a message to the inheritor. What will happen if we unroll the loop as shown in Figure 1?

We can compute the output as follows in a simple RNN:

$$o_t = f(W_o h_t) \tag{1}$$

$$h_t = \sigma(W_h h_{t-1} + W_x x_t) \tag{2}$$

Where W_o, W_h and W_x are the matrices for hidden layer output h_t, past hidden layer activity h_{t-1} and the input x_t. The time recurrence is presented in Eq. (2); the equation conveys the presents hidden layer activity h_t with its past hidden layer activity h_{t-1}. This reliance is nonlinear due to the use of logistic function $\sigma(\cdot)$.

IV. RESEARCH PROBLEM AND MOTIVATION

The objective of using the convolutional layer is for it to learn to extract higher-level features that are invariant to local translation, and, by assembling multiple convolutional layers, the model can extract higher-level translation invariant features from the input sequence. Regardless of this advantage, we observed that most of the existing deep models require multiple layers of convolutional to capture long-term dependencies, and that is because of the locality of the convolutional and pooling layers. This issue becomes more crucial as the length of the input sequence grows. Most of the combination CNN-RNN models applied several types of pooling. We argue that the pooling layer is the reason for lost details in local information, because the pooling layer only captures the most important feature in a sentence and ignore the others; therefore, we attempt to utilize an RNN as an alternative for the pooling layer to capture long-term dependencies more efficiently and also to reduce the number of the parameters in the architecture. Based on these observations, we focused on proposing a simple and efficient combined model that focuses on parameter reduction by excluding the pooling layer from the architecture, while also capturing long-term dependencies more efficiently in terms of accuracy by using the LSTM layer as an alternative to the pooling layer. Only one convolutional layer was applied to extract the most important features in the document; no pooling layers were involved. Further, we fed the feature maps to the recurrent layer to capture long-term dependencies for more efficient classification as shown in Figure 3.

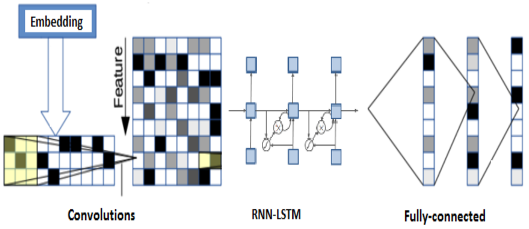

FIGURE 2. The Proposed Convolutional Recurrent Framework.

V. MODEL ARCHITECTURE

In this section, we present the details of the framework model, which consists of convolutional and recurrent neural networks. Our model's architecture uses word embeddings as inputs and takes them to a convolutional neural network to learn to extract high-level features, whose outputs are then given to a long short-term memory recurrent neural network language model, and are finally followed by a classifier layer.

A. THE EMBEDDED LAYER

The first layer of the network transforms words into real-valued feature vectors that capture semantic and syntactic information. Our model's input is a sequence of words $[w_i, \ldots, w_{|s|}]$, with each word being derived from vocabulary V. Words are denoted by distributed vectors $W \in R^{1*d}$ and looked up in a word embedding matrix $W \in R^{1*|v|}$. This is formed by simply concatenating embeddings of all words in V.

B. THE CONVOLUTIONAL LAYER

The model architecture in Figure 2 is a slight alternative of the CNN architecture of [31].

$x_i = R^k$ will be the kdimenstional word vector equivalent to the ithe word in the sentence of length n, which is represented as:

$$x_{1:n} = x_1 \oplus x_2 \oplus \ldots \oplus x_n, \tag{3}$$

Where $\oplus$ is a concatenation operator. Overall, let $x_{i::i+j}$ refer to the concatenation of words $x_i, x_{i+1}, \ldots x_{i+j}$. A convolutional operation includes a filter $w \in R^{hk}$, which is applied to windows of h words to produce new features. For instance, a feature is c_i generated from a window of words $x_{i:i+h-1}$ by:

$$c_i = f(W \cdot X_{i:i+h-1} + b). \tag{4}$$

Here $b \in R$ is a bias term and f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the sentence $\{x_{1:h}, x_{2:h+1}, \ldots, x_{n-h+1:n}\}$ to produce a feature map:

$$c = [c_1, c_2, \ldots, c_{n-h+1}], \tag{5}$$

With $c \in R^{n-h+1}$. Then, we fed the feature maps to a recurrent layer LSTM to capture long-term dependencies. This technique will reduce the number of parameters in the model.

C. THE RECURRENT LAYER

RNN is a type of neural network architecture specially used for sequence modeling. At each time step t, a recurrent layer takes the input vector $x_t \in R^n$ and hidden state h_t by applying the recursive operation:

$$h_t = f(W_{xt} + Uh_{t-1} + b) \tag{6}$$

Where $W \in R^{m*n}, b \in R^{m*m}, b \in R^m$ parameters, and f is an element-wise nonlinearity. Learning long-term dependencies with a vanilla RNN is difficult because of the vanishing and exploding gradient [32]. Long short-term memory LSTM [33] overcomes the deficiencies of an RNN by augmenting the RNN with a memory cell that takes as an input x_t, h_{t-1}, c_{t-1}, and produces h_t, c_t by the following:

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \tag{7}$$

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \tag{8}$$

$$O_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \tag{9}$$

$$g_t = \sigma(W^g x_t + U^g h_{t-1} + b^g) \tag{10}$$

$$c_t = f_t \Theta c_{t-1} + i_t \Theta g_t \tag{11}$$

$$h_t = o_t \Theta \tanh(c_t) \tag{12}$$

Where σ, and tanh are the element-wise sigmoid and hyperbolic tangent function and i_t, f_t, o_t are referred to as input, forget, and output gates. At $t = 1, h_0, c_0$ are initialized to zero vectors. Θ is the element-wise multiplication operator. Parameters of the LSTM are preservative with respect to time. LSTM outperforms vanilla RNNs on many tasks, including language modeling [14]. It is easy to extend LSTM to more than one layer. Having multiple layers is critical for attaining competitive performance on various tasks [29].

D. BACK PROPPAGATION THROUGH TIME

Back propagation through time (BPTT) is the key algorithm that makes training deep models computationally controllable, and it is a way of computing gradients of expression through the recursive application of the chain rule. The core issue we are given is some function $f(x)$ where x is the vector of inputs, and we are interested in computing the gradient of f at $x (i.e \nabla f(x))$. Error can be even backpropagated further [34]. BPTT is a simple extension of the backpropagation algorithm for recurrent neural networks; with BPTT the error is broadcasted via recurrent connection back in time for specific time steps. Therefore, the network absorbs and remembers information for numerous time steps in the hidden layer when it is learned by BPTT. More details about the implementation described can be found in [33].

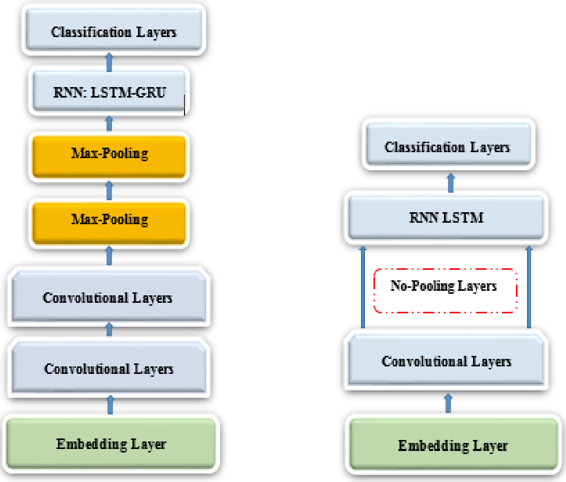

FIGURE 3. The proposed CNN-LSTM architecture compare to traditional CNN-RNN with max-pooling architecture.

E. THE CLASSIFICATION LAYER

The classification layer is, in principle, a logistic regression classifier. It gives a fixed-dimensional input from the lower layer; the classification layer affine transforms it, followed by a softmax activation function to compute the predictive probabilities for all of the categories [35]. This is done by:

$$p(y = k|X) = \frac{\exp(w_k^T x + b_k)}{\sum_{k=1}^{k} \exp(w_k^T x + b_k)} \tag{13}$$

Where w_k and b_k are the weight and bias vectors. We assume there are k categories.

VI. EXPERIMENTAL SETUP

A. SENTIMENT ANALYSIS DATASETS

The performance of the proposed model was evaluated on two benchmark sentiment analysis datasets: the Stanford Large Movie Review dataset (IMDB) and the Stanford Sentiment Treebank dataset (SSTb) [31], derived from Rotten Tomatoes movie reviews [36].

B. STANFORD LARGE MOVIE REVIEW DATASET (IMDB)

The IMDB dataset was first proposed by [32] as a benchmark for sentiment analysis. It consists of 50,000 binary labeled reviews; the reviews are divided into 50:50 training and testing sets. The distribution of labels with each subset of data is balanced. We used 15% of the labeled training documents

TABLE 1. Units for magnetic properties.

Dataset	Set	Sentence	Binary
SSTb	*Train*	*8544*	*2, 5*
	Dev	*1101*	*2,5*
	Test	*2210*	*2, 5*
IMDB	*Train*	*2210*	*2*
	Dev	*4k*	*2*
	Test	*25k*	*2*

as a validation set. One key aspect of this dataset is that each review has several sentences.

C. STANFORD SENTIMENT TREEBANK DATASET (SSTb)

The SSTb dataset was first proposed by [36] and extended by [31] as a benchmark for sentiment analysis. It consists of 11,855 reviews taken from the movie review site Rotten Tomatoes, with one sentence for each review. The SSTb was split into three sets: 8544 sentences for training, 2210 sentences for testing, and 1101 sentences for validation (or development). The SSTb also includes fine-grained sentiment labels. In Table 1, we present additional details about the two benchmark datasets.

D. HYPERPARAMETERS AND TRAINNIG

We used stochastic gradient descent (SGD) to train the network and the back-propagation algorithm to compute the gradient. We believe that by adding a recurrent layer to the model as an alternative to the pooling layer, we can effectively reduce the number of the convolutional layers needed to capture long-term dependencies. Therefore, we consider merging a convolutional and recurrent layer into one single model. Our architecture goal is to reduce the need for stacking multiple convolutional and pooling layers in the network in order to reduce the loss of detailed, local information. Thus, in the proposed model, we consider convolutional layers with only one layer that has $d = 256$ filters and a receptive field size of $r(3, 3, 5)$. For an activation function we use rectifier linear units in the convolutional layer (ReLU). The recurrent layer is fixed to a single layer of LSTM. The hidden state dimension is $d = 128$. For both datasets, the number of training epochs varies between (5, 20). We compared the proposed model with methods using word embedding and convolutional architecture and different deep learning and traditional methods. We also focused on the regularization, the learning rate, and dropout parameters; we then extracted sentence features with the convolutional layer. The recurrent layer provides an indication of the robustness of our approach in multiple domains. In Table 2, we show the selected hyperparameter value for the proposed architecture.

E. REGULARZATION

For regularization we employ dropout as an effective method to regularize deep neural networks and neural networks. Dropout prevents co-adaption of hidden units. We apply it with constraint on the L2-norms of the weight vectors [37]; we insert dropout modules in between CNN and LSTM layers to regularize them.

TABLE 2. Units for magnetic properties.

Parameter	CNN	RNN-LSTTM
Word-Embedding-Dimension	*300*	*300*
Word Context Units	*5*	*5*
Hidden Units	-	-
Learning rate	*0.01*	*0.01*
Dropout	*0.5*	*0.5*

F. UNSUPERVISED LEARNING OF WORD-LEVEL EMBEDDINGS

Initializing word vectors with those obtained from an unsupervised neural language model is a popular method to improve performance in the absence of a large, supervised training set [35], [38]. It has been recently shown that improvements in model accuracy can be obtained by performing unsupervised, pre-trained word embeddings. In our experiments, we utilized the publicly available word2vec vectors that were trained on 100 billion words from Google news. The vectors were trained using a continuous bag-of-words algorithm [7]. While the word embeddings are obtained, the model captures syntactic and semantic aspects of the words they represent; however, they have no notion about their sentiment behavior. Word embeddings play an important role in our neural language model. They are able to capture syntactic and semantic information, which are very significant to sentiment analysis.

VII. EMPIRICAL RESULTS AND ANALYSIS

A. OPTIMIZATION

Training was done through stochastic gradient descent over shuffled mini-batches. For training and validation, we randomly split the full training examples. The size of the validation set is the same as the corresponding test size and is balanced in each class. We trained the model by minimizing the negative log-likelihood or cross entropy loss. Early stopping was utilized to prevent overfitting. In our work, we employed unsupervised learning of word-level embedding using the word2vec, which implemented the continuous bag-of-words and skip-gram architectures for computing vector representations of a word. We validated the proposed model on two datasets, considering the difference in the number of parameters. However, the accuracy of the model does not increase with the number of convolutional layers. More pooling layers typically leads to the loss of long-term dependencies. Therefore, in our model we removed the pooling layer from the convolutional network and replaced it with a recurrent layer to reduce the loss of local information. One recurrent layer is enough to capture long-term dependencies in the input sequence.

TABLE 3. The performance of our approach compared to other approaches on imdb dataset. the accuracy of binary predication.

Parameter	BINARY
MNB-uni [39]	83.5%
MNB-bi [39]	86.6%
SVM-uni [39]	86.9%
SVM-bi [39]	89.2%
NBSVM-uni [39]	88.3%
NBSVM-bi [39]	91.2%
WEEBM +Bow [42]	87.8%
WRBB + Bow (bnc) [42]	89.2%
BoW (bnc) [32]	87.8%
Full +Bow [32]	88.3%
Full +Unlabeled +Bow [32]	88.9%
Paragraph Vector [43]	92.5%
Paragraph Vector (LogReg) [44]	94.4%
Paragraph Vector (2-Layer MLP) [44]	94.5%
Our approach	93.2%

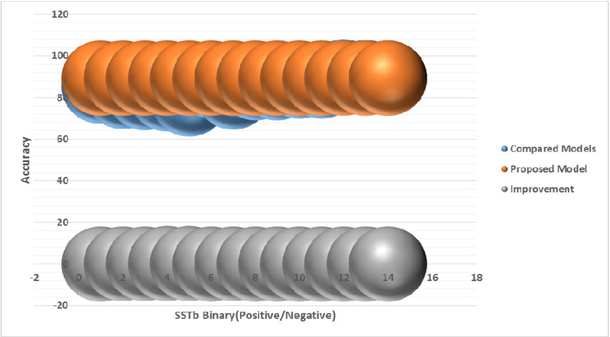

FIGURE 4. Results on SSTb dataset for binary predictions.

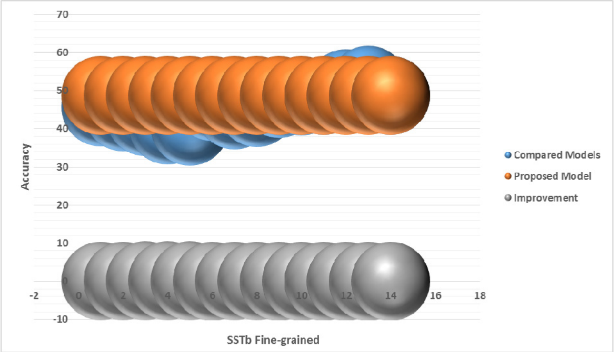

FIGURE 5. Results on SSTb dataset for fine-grained (5-classes).

B. ANALYSIS OF THE STANFORD SENTIMENT TREEBANK DATASET

For the Stanford Sentiment Analysis dataset (SSTb), we performed several experiments to offer a fair comparison with competitive models. We followed the experimental protocols as described in [31]. To make use of the available labeled data, our model treats each sub-phrase as an independent sentence, and we learn the representation for all of the sub-phrases in the training set. We initialized the word vectors with the unsupervised learning of word-level embedding using the word2vec algorithm, which implements continuous bag-of-words and skip-gram architectures for computing vector representations of a word. The Positive/Negative presents results for the binary classification of sentences, and the fine-grained analysis predicts results for the case where five sentiment classes are used (positive, very positive, negative, very negative, and neutral). We report the accuracy of different methods in Table 3. The primary highlight of our result on the SSTb benchmark dataset is that traditional methods (SVV, NB, BiNB) with bag-of-words perform poorly compared to our proposed deep learning language model. We observed 4%-12% absolute improvement in terms of accuracy with the baseline methods proposed in [39]. Initializing word-embeddings using unsupervised, pre-trained vectors gives the model an absolute accuracy that increased around 8% when compared to randomly initializing the vector with a CNN-only architecture [9]. Our model does not require pooling layers, which leads to the more efficient capture of local information compared to the networks proposed in [12] and [18]. The best previous result was reported by [31] and [40] for SSTb. Our approach provides a 4% improvement in accuracy over the RNTN method. We also reported an 8% performance enhancement over the matrix-vector-RNN. In fine-grained classification tasks, our method has an absolute improvement of 7% in terms of accuracy. Figures 4 and 5 show that SSTb (binary and fine-grained), bag-of-n-words model, and (NB, SVM, BiNB) perform poorly on the dataset. A similar model was proposed in [41] and achieved better performance in terms of accuracy; however these models have more hyperparameters and require subsampling layers. On the other hand, our proposed model performed very competitively and came close to matching other state-of-the-art algorithms on both the binary and fine-grained sentiment analyses on the SSTb dataset with fewer parameters.

C. ANALYSIS OF THE IMDB DATASET

Beyond one sentence, each movie review consists of several sentences in the IMDB dataset. The results of our method are reported in Table 4 on the IMDB benchmark dataset compared to other approaches. Reference [31] Applied several methods on the IMDB dataset and found that their Recursive Neural Tensor Network worked much better than a bag-of-words model; however, this model required parsing and took into account the compositionality. Our method performs better than all of the baselines reported in [39]: MNB-uni, MNB-bi, SVM-uni, SVM-bi, NBSVM-uni, and NBSVM-bi, with an approximate improvement of 2-12% in terms of accuracy.

When we compared the proposed model with a combined Restricted Boltzmann Machines model [42], bag-of-words, and WRRMB + BoW (bnc), we achieved 4%-7% relative improvement and 1%-6% compared with Bow (bnc), Full + Unlabeled + BoW, and paragraph vector [43]. The paragraph vectors proposed in [44] achieved a state-of-the-art result on the IMDB dataset; however the model has a reputation for being extremely difficult to tune and requires a downsampling parameter to reduce the feature map dimensionality for computational efficiency. We found that our proposed architecture, with no downsampling layer, achieved competitive

TABLE 4. The performance of our approach compared to other approaches on sstb dataset. the accuracy of fine-grained and binary predications are reported in the table.

Parameter	CNN	BINARY
RNTN [31]	45.7%	85.4%
MV-RNN [40]	44.4%	82.9%
RAE[35]	43.2%	82.4%
NB [31]	43.2%	82.4%
SVM [31]	41.0%	79.4%
CNN-Multi-channel [9]	47.1%	88.1%
CNN-rand [9]	45.0%	82.7%
CNN-static [9]	45.5%	86.8%
CNN-non-static [9]	48.0%	87.2%
DCNN [11]	48.5%	87.8%
Paragraph-Vec [43]	48.7%	87.8%
CNN-GRU-word2vec [41]	50.6%	89.9%
CNN-LSTM-word2vec [41]	51.5%	89.5%
Our approach	48.8%	89.2%

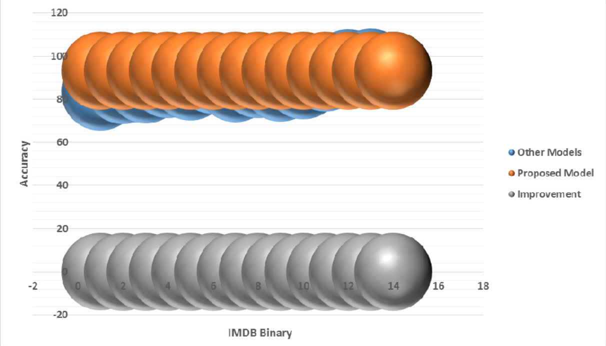

FIGURE 6. Results on IMDB dataset for binary predictions.

results on the IMDB dataset as shown in Figure 6. The CNN-RNN with max-pooling loses detailed, local features due to the pooling layers in the architecture. Compared with the existing methods and experiment results, we found that the approach takes advantage of both CNN and RNN models on the sentiment classification of short texts.

Our experimental results suggest that by using a LSTM layer on top of a CNN architecture, one can effectively reduce the number of convolutional layers needed in order to capture long-term dependencies. Furthermore, we observed that many factors affect the performance of deep learning models, such as: the dataset size, vanishing/exploding of the gradients, and choosing the best feature extractors and classifiers, which are all still open research areas. However, there is no specific model for all types of datasets.

D. OVERVIEW

The challenge in NLP is to develop an architecture that can learn the hierarchal representation of the whole sentence jointly with the task. Convolutional neural networks consider feature extraction and classification as one jointly trained task. The idea of CNNs has been improved upon recently [9], [13], [16]–[18] by using multiple layers of convolutional and pooling to sequentially extract hierarchal representation of input. Reducing the network size has been the interest of several works. More compact layers are also used, likely by replacing the fully connected layers with average pooling [15]. In [16] the weights are constrained by binary, which considerably reduces the memory consumption. To design a simpler network, [15] removed redundant connections and allowed weight sharing. In our work we conducted a series of experiments with both deep learning and traditional methods to offer a fair comparison to competitive models on sentiment analysis benchmark datasets. We did our best to select the architectures that would deliver comparable and competitive results. Despite the fact that the CNN-RNN proposed in [41] has a slightly higher classification accuracy compared to our proposed model, we argue that this result is due to the use of max pooling on adjacent words. However, our proposed architecture is simple and efficient in term of layers. Moreover, our model has significantly fewer parameters, which means less memory consumption. We reported very competitive results in terms of accuracy in comparison to the model proposed in [41]. The reported result shows that, compared to the currently most popular LSTM, CNN, and CNN-LSTM methods, our proposed framework can achieve similar or even better performance on sentiment analysis tasks.

VIII. CONCLUSION

Convolutional neural networks (CNN) learn to extract higher-level features that are invariant to local translation. Despite this advantage, it requires many layers of convolution to capture long-term dependencies, due to the locality of the convolutional and pooling. This becomes more severe as the length of the input sequence grows. Ultimately, this leads to the need for a very deep network with many convolutional layers. In this article, we presented a new framework to overcome this problem. In particular, we aimed to capture the subword information and reduce the number of the parameters in the architecture. Our framework jointly combines CNN and recurrent neural networks (RNN) on top of unsupervised, pre-trained word vectors; recurrent layers are expected to preserve ordering information even with one single layer. Thus, we exploited a recurrent layer as a substitute for the pooling layer to hypothetically reduce the loss of details in local information and capture long-term dependencies more efficiently.

Our approach performed well on two benchmark datasets and achieved a competitive classification accuracy while outperforming several other methods. Our results demonstrated that it is possible to use a much smaller architecture to achieve the same level of classification performance. It will be interesting to see future research on applying the proposed method to other applications such as information retrieval or machine translation.

REFERENCES

[1] A. Graves, A.-R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, p. 391, 1990.

[4] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Found. Trends Inf. Retr.*, vol. 2, nos. 1–2, pp. 1–135, 2008.

[5] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. Eur. Conf. Mach. Learn.*, 1998, pp. 137–142.

[6] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (Jan. 2013). "Efficient estimation of word representations in vector space." [Online]. Available: https://arxiv.org/abs/1301.3781

[9] Y. Kim. (Aug. 2014). "Convolutional neural networks for sentence classification." [Online]. Available: https://arxiv.org/abs/1408.5882

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[11] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. (Apr. 2014). "A convolutional neural network for modelling sentences." [Online]. Available: https://arxiv.org/abs/1404.2188

[12] A. Conneau, H. Schwenk, L. Barrault, and Y. LeCun. (Jun. 2016). "Very deep convolutional networks for text classification." [Online]. Available: https://arxiv.org/abs/1606.01781

[13] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.

[14] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent LSTM neural networks for language modeling," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 23, no. 3, pp. 517–529, Mar. 2015.

[15] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in *Proc. AAAI Workshop Learn. Text Categorization*, 1998, pp. 41–48.

[16] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[17] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12 pp. 2493–2537, Aug. 2011.

[18] Y. Xiao and K. Cho. (Feb. 2016). "Efficient character-level document classification by combining convolution and recurrent layers." [Online]. Available: https://arxiv.org/abs/1602.00367

[19] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] N. M. Mayer, "Echo state condition at the critical point," *Entropy*, vol. 19, no. 1, p. 3, 2016.

[22] W.-T. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *Proc. ACL*, vol. 2. 2014, pp. 643–648.

[23] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "Learning semantic representations using convolutional neural networks for Web search," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 373–374.

[24] T. N. Sainath *et al.*, "Deep convolutional neural networks for large-scale speech tasks," *Neural Netw.*, vol. 64, pp. 39–48, Apr. 2015.

[25] Y. Kim, C. Denton, L. Hoang, and A. M. Rush. (Feb. 2017). "Structured attention networks." [Online]. Available: https://arxiv.org/abs/1702.00887

[26] A. G. Ororbia, II, T. Mikolov, and D. Reitter, "Learning simpler language models with the differential state framework," *Neural Comput.*, vol. 29, no. 12, pp. 3327–3352, 2017.

[27] K. Vo, D. Pham, M. Nguyen, T. Mai, and T. Quan, "Combination of domain knowledge and deep learning for sentiment analysis," in *Proc. Int. Workshop Multi-Disciplinary Trends Artif. Intell.*, 2017, pp. 162–173.

[28] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[29] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. ICML*, 2013, pp. 1310–1318.

[30] A. Graves. (Aug. 2013). "Generating sequences with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1308.0850

[31] R. Socher *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2013, pp. 1631–1642.

[32] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1. 2011, pp. 142–150.

[33] M. Bodén, "A guide to recurrent neural networks and backpropagation," Dallas Project, Dept. Comput. Sci., Univ. Skövde, Skövde, Sweden, Tech. Rep., 2002.

[34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognit. Model.*, vol. 5, no. 3, p. 1, 1988.

[35] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proc. Conf. Empirical Methods Natural Lang. Process*, 2011, pp. 151–161.

[36] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics*, 2005, pp. 115–124.

[37] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. (Jul. 2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: https://arxiv.org/abs/1207.0580

[38] R. Collobert, "Deep learning for efficient discriminative parsing," in *Proc. AISTATS*, 2011, pp. 224–232.

[39] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2. 2012, pp. 90–94.

[40] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, 2012, pp. 1201–1211.

[41] X. Wang, W. Jiang, and Z. Luo, "Combination of convolutional and recurrent neural network for sentiment analysis of short texts," in *Proc. COLING*, 2016, pp. 2428–2437.

[42] G. E. Dahl, R. P. Adams, and H. Larochelle. (Feb. 2012). "Training restricted boltzmann machines on word observations." [Online]. Available: https://arxiv.org/abs/1202.5695

[43] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. ICML*, 2014, pp. 1188–1196.

[44] J. Hong and M. Fang, "Sentiment analysis with deeply learned distributed representations of variable length texts," Stanford Univ., Stanford, CA, USA, Tech. Rep., 2015.

ABDALRAOUF HASSAN (M'14) received the B.S. degree in electrical engineering from the Libyan Aviation Academy, Tripoli, Libya, in 2003, and the M.S. degree in electrical engineering from the University of Bridgeport, CT, USA, in 2012, where he is currently pursuing the Ph.D. degree with the Computer Science and Engineering Department. He was trained as an Electrical Engineer with Schlumberger Oil Service Corporation, Tripoli, from 2004 to 2008. After obtaining his M.S. degree, he joined the Duracell Research and Development Department, Duracell Inc., Bethel, CT, USA, from 2012 to 2013, as a Lab Researcher. He is also an Applications Systems Analyst with the Behavioral Research Lab, Yale School of Management, New Haven, CT, USA. His main areas of research interests include artificial intelligent, deep learning, machine learning, and natural language

AUSIF MAHMOOD received the M.S. and Ph.D. degrees in electrical and computer engineering from Washington State University, USA. He is currently a Professor with the Department of Computer Science and Engineering and the Department of Electrical Engineering, University of Bridgeport, Bridgeport, CT, USA. His research interests include parallel and distributed computing, computer vision, deep learning, and computer architecture.