Artificial Intelligence Review (2020) 53:6155–6195
https://doi.org/10.1007/s10462-020-09845-2

Sentiment analysis with deep neural networks: comparative study and performance assessment

Ramesh Wadawadagi[1] · Veerappa Pagi[1]

Published online: 22 May 2020

Abstract

The current decade has witnessed the remarkable developments in the field of artificial intelligence, and the revolution of deep learning has transformed the whole artificial intelligence industry. Eventually, deep learning techniques have become essential components of any model in today's computational world. Nevertheless, deep learning techniques promise a high degree of automation with generalized rule extraction for both text and sentiment classification tasks. This article aims to provide an empirical study on various deep neural networks (DNN) used for sentiment classification and its applications. In the preliminary step, the research carries out a study on several contemporary DNN models and their underlying theories. Furthermore, the performances of different DNN models discussed in the literature are estimated through the experiments conducted over sentiment datasets. Following this study, the effect of fine-tuning various hyperparameters on each model's performance is also examined. Towards a better comprehension of the empirical results, few simple techniques from data visualization have been employed. This empirical study ensures deep learning practitioners with insights into ways to adapt stable DNN techniques for many sentiment analysis tasks.

Keywords Deep neural networks · Sentiment analysis · Performance evaluation · Aspect-based sentiment analysis · Opinion mining

1 Introduction

The explosive growth of opinion content generated through commercial websites and recent advances in data analytics together have placed new challenges and opportunities (Khan et al. 2017; Chaturvedi et al. 2018; Wadawadagi and Pagi 2019a). Investigating peculiar and potentially useful patterns from a large collection of user-generated content (UGC) is crucial for many sentiment analysis tasks (Medhat et al. 2013; Chen and Lee 2019). Sentiment analysis techniques are specially used to recognize and extract subjective content in source data to assist an enterprise in understanding the social sentiment of their brand, product, or services (Pandarachalil et al. 2015; Krishnakumari et al. 2019).

✉ Ramesh Wadawadagi
rswlib@yahoo.co.in

[1] Basaveshwar Engineering College, Bagalkot 587 102, India

However, identification of sentiments in UGC faces numerous challenges, as they are composed of incomplete, noisy, and unstructured sentences, unusual expressions, ungrammatical phrases, and non-lexical terms (Hassan 2019; Wadawadagi and Pagi 2019a). Besides, it is hard to explore the correlation among opinion sentences due to the diversity of linguistic issues and makes the process of sentiment analysis still more challenging (Medhat et al. 2013; Zhang et al. 2017). Hence, to address these challenges, real-time sentiment analysis systems need to be developed for processing large volumetric opinion data in a reasonable amount of time. Despite numerous challenges, understanding public emotions has potential benefits in many fields, including marketing, politics, online shopping, and many more (Pandarachalil et al. 2015; Cambria et al. 2017; Valdivia et al. 2017; Weichselbraun et al. 2017; Wadawadagi and Pagi 2020a). To improve productivity, many business firms encourage their customers to participate in virtual discussions, asking for their feedback, opinions, and suggestions.

Generally, the problem of sentiment analysis is performed at different levels of granularity, varying from coarse-grained to fine-grained levels. The coarse-grained analysis deals with determining the sentiment of a whole phrase, while fine-grained sentiment analysis is related to attribute-level or phrase-level (Fink et al. 2011; Cambria 2016; Zhang et al. 2017; Lo et al. 2017). However, implementing the right methodology to any key research drives sentiment analysis as a powerful tool that delivers research goals as successful outcomes. In the past, the problem of sentiment classification has been tackled using several traditional natural language processing (NLP) and machine learning techniques. Regardless of the popularity of these techniques, the revolution of deep learning has changed the perspective of computational intelligence by solving the problems using deep architectures (Wang et al. 2016; Liu and Zhang 2017a). In precise, deep neural networks (DNN) have surpassed other methods by a wide margin through accuracy and sometimes even efficiency. Nevertheless, DNN models are far from the maturity achieved by other traditional categories of NLP and machine learning. The constantly evolving state of the field makes initiation challenging, and keeping up with its evolution speed is an incredibly time-consuming task due to the sheer amount of new literature being introduced. This status drives the practitioners to record the updates about sentiment analysis research and interpret their schemes and validate results accordingly.

DNN are basically designed to model the composite non-linear relationships prevalent in data samples (Schmidhuber 2015; Liu and Zhang 2017a). They construct compositional rules with data objects, which are expressed as a layered composition of primitives (Yousefi-Azar and Hamey 2017). Furthermore, the hidden layers of DNN learn feature hierarchies from higher levels of the hierarchy formed by the composition of lower-level features so that each subsequent layer potentially gains a more abstract meaning (Pascanu et al. 2014; Srivastava et al. 2015). Additionally, DNN have evolved with several variants of underlying architectures and successfully employed for specific domains. Nevertheless, the performance of different DNN architectures can be compared adequately, provided the same datasets are employed for evaluation.

The main contribution of this research work is to study empirically various DNN models utilized for sentiment classification. Hence, different networks are studied in the context of their performances evaluated through experiments conducted over benchmark sentiment datasets. Consequently, the effect of tuning hyperparameters on the performance of various networks is also studied. Finally, the empirical results are interpreted with the help of data visualization techniques for better understanding the models. The remainder of this paper is organized according to the following structure. Section 2 presents the framework of DNN architecture and its variants used in the field of NLP and sentiment analysis. A brief introduction to word-embeddings and their usage in sentiment analysis is elaborated in Sect. 3. However, Sect. 4 gives a brief account on different DNN architectures used in sentiment analysis. In Sect. 5, the experimental results are tabulated, and a comparative study of various DNN models is presented. Furthermore, the effect of tuning hyperparameters on network performance is examined in Sect. 6. Moreover, Sect. 7 presents the study on DNN utilized for aspect-based sentiment analysis (ABSA). Finally, Sect. 8 provides a summary of our research contributions, concluding remarks, and future improvements.

2 The framework of deep neural network architectures

DNN are essentially the artificial neural networks (ANN) with several hidden layers embedded between the input and output layers (Schulz and Behnke 2012; Pascanu et al. 2014; Schmidhuber 2015). Hence, they are distinguishable from shallow single-layered networks by their depth that denotes the number of layers through which information navigates (Socher et al. 2013; Irsoy et al. 2014a). The notion of implanting multiple hidden layers in DNN enable learning through the adjustment of connection weights between neurons, resembling the learning process of a biological brain. During its functionality, the lower-order layers that are close to the input layer learn trivial features, while higher-order layers learn more significant features obtained from lower layer features (Irsoy and Cardie 2014b). Then, a hierarchical and comprehensive feature mapping is obtained as a result of the network model. Furthermore, based on the underlying topology, neural networks are generally categorized into conventional neural networks (CNN), recurrent neural networks (RNN), and hybrid neural networks (Schmidhuber 2015; Liu and Zhang 2017a; Zhang et al. 2017b). The general architecture of a deep feedforward neural network (DFNN) is demonstrated in Fig. 1.

The model consists of n layers l_1, l_2, ...l_n, where l_1 is the input layer concerned to the input sequence x=x_1, x_2, ...x_f, w_1, w_2, ...w_i are the weights of the connections on each layer, and b_1, b_2, ...b_i is intercept bias vector. The output layer l_n is associated with output vector y_1, y_2, ...y_m, whereas layers from l_2 to l_{n-1} signify the hidden layers, whose outcomes are invisible in the network. The values in the input layer denote input data supplied to the network, while elements in the hidden and output layers represent the basic computational units called neurons. These neurons are also known as activation functions, which

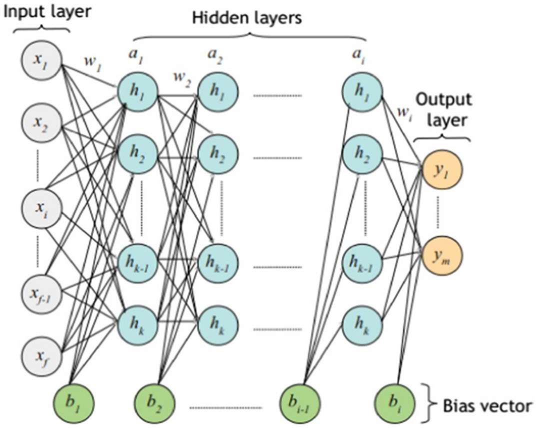


Fig. 1 A generic DFNN architecture

are responsible for non-linear functional mapping between the inputs and a response variable. The flow of information between the neurons is determined through a set of connections using directional edges. Furthermore, each connection in the network is associated with a weight that controls the signal between two neurons. Tuning the weights between neurons allow the network to learn features from a higher level of the hierarchy formed by the composition of lower-level features, where each subsequent layer gains a more abstract representation (Yosinski et al. 2015; Guo et al. 2018). Then, the activation g at each hidden layer i is the weighted sum of inputs from the previous layer and the bias. Mathematically it is denoted as in Eq. 1.

$$h_i(x) = g\left(w_i h_{i-1} + b_i\right) \tag{1}$$

The final output of the network is determined through some activation function applied to the output of the previous hidden layer, as described in Eq. 2.

$$y_m(x) = O\left(h_i(x)\right) \tag{2}$$

The choice of output activation function O depends on the problem at hand (Chaturvedi et al. 2018; Krishnakumari et al. 2019), and the popular selections are *sigmoid* function, hyperbolic tangent function (*tanh*), rectified linear unit (*ReLu*), or a *softmax*. The *sigmoid* is a non-linear activation function of the form $f(x)=\frac{1}{(1+e^{-x})}$, which is continuously differentiable over different values of x and yields a fixed output in the range between 0 and 1. Typically it is used in the output layer of a binary classifier, where the expected output is either 0 or 1. The *tanh* function is an improved version of the *sigmoid* function; however, the only difference is, in the case of *tanh*, the output values range between -1 to 1. Despite their simplicity, *sigmoid* and *tanh* functions are least preferred in DNN due to vanishing gradient problem that seriously affects the performance of the model (Nair and Hinton 2010). Hence, another activation function known as *ReLu* is considered the most preferable activation function used in DNN. It results in the output value 0 if it receives a negative input of x, else for positive inputs, it returns x unchanged similar to a linear function. Mathematically, it is denoted as $f(x) = max(0, x)$. But, *ReLu* hold the limitation of applicability, and hence, it is used only for hidden layers. Then, a *softmax* function similar to *sigmoid* function adequate for handling multi-class problems is utilized. It computes the probability of each input belongs to a predefined class and adjusts the output values for each class to lie in the range of 0 to 1 (Liang et al. 2017). Typically, *softmax* function is employed only for the output layer.

The neural networks are often trained with optimization techniques that need a loss function to estimate the model error. Depending upon the learning task, the loss is measured usually in terms of negative log-likelihood or residual sum of squares. Hence, the network parameters are optimized with the output of the loss function employing different optimization techniques (Le et al. 2011). Stochastic gradient descent (SGD) optimization with back-propagation is the one commonly used technique in training most traditional neural networks (LeCen et al. 2015). In the SGD optimization technique, initial gradients are computed sequentially for a loss function concerning the weights between a last hidden layer and the output layer. Then gradients within higher-order layers are computed through the recursive application of chain rule in a bottom-up fashion. Meanwhile, the gradients collected are utilized to adjust the weights between the layers. This process is repeated in succession to refine the loss function until certain criterion is met. However, SGD often exhibits unusual challenges and may not be suitable for training DNN. Hence, more research has been reported on other optimization techniques that maintain increased flexibility. Recent advances in optimization techniques, including AdaGrad, AdaDelta, Adam, and RMSProp, employ adaptive learning rates that result in smoother and faster convergence (Ruder 2016). Moreover, DNN are loaded densely with hyperparameters, and determining the optimal configuration for parameters in a high dimensional space is challenging. Tuning the hyperparameters such as learning rate, batch size, momentum, and weight decay, becomes essential during the training process. These hyperparameters act as switches that can be tuned finely during the training of any model. For any model to produce good results, it is crucial to obtain the optimal values for these parameters.

3 Word embeddings

Deep learning techniques employed for sentiment analysis and NLP relies heavily on word-embeddings as input features (Collobert et al. 2011; Mikolov et al. 2013a; Pennington et al. 2014; Dragoni and Petrucci 2017; Dragoni et al. 2018; Guo et al. 2019; Pasupa and Ayutthaya 2019). The intuition behind word-embeddings has been that the lexical with similar meaning would take similar representations. They consist of real-valued low-dimensional vectors to model syntactic and semantic information of different words. Furthermore, these vectors can be utilized in many sentiment analysis related tasks as pre-trained feature vectors. Word-embeddings present several advantages over traditional bag-of-words (BoW) representations. For instance, words with semantically identical meanings have close proximity in the word-embedding space. The majority of word-embeddings are based on either Euclidean distance or angle between two word-vectors as a quality criterion of measuring such representations.

The learning of word-embeddings can be achieved through DNN (Mikolov et al. 2013a, b; Mnih and Kavukcuoglu 2013) or with matrix factorization (Huang et al. 2012; Pennington et al. 2014). For example, the *Word2Vec* is one commonly used word-embedding system, which is a shallow two-layer neural network prediction model that learns word-embeddings from a text corpus. It includes a continuous BoW (CBoW) model (Lebret and Collobert 2014) and skip-gram model (Mikolov et al. 2013a). Both techniques have drastically opposite points of views, the former one predicts a word, given a context window, the subsequent predicts the context window for a given word. Most recently, a new word-embedding learning technique called *GloVe* is presented (Pennington et al. 2014). *GloVe* uses word-to-word co-occurrence counts in co-occurrence matrix for training the text corpora, making efficient use of global corpus statistics and preserves the linear substructure of *Word2Vec*. The empirical study reveals that *GloVe* scales well on bigger text corpora and outperforms several other methods when applied for sentiment analysis tasks.

Several improved variants of existing word-embedding techniques have been discussed. For example, Tang et al. (2014) introduce a model to learn continuous word representations for sentiment analysis on the Twitter dataset. Similarly, a model based on *Word2Vec* to learn word-embeddings on a large collection of tweets is presented (Severyn and Moschitti 2015; Ren et al. 2016). Furthermore, some researchers train *Word2Vec* model on Wikipedia corpus consisting of 500M words and employed for various sentiment analysis tasks (Qin et al. 2016; Fu et al. 2017). However, Yu et al. (2017) propose a word-vector refinement model that can be applied to any pre-trained word-vectors such as *Word2Vec* and *GloVe*. The refinement model is based on tuning the word-vector representations so that they can have closer proximity with semantically identical words. Recently, a discriminant document-embeddings based on skip-gram model for generating word-embeddings of clinical texts is proposed (Lauren et al. 2018). Similarly, an improved word-vector that increases the performance of pre-trained word-embeddings in sentiment analysis is also discussed (Rezaeinia et al. 2018). Additionally, Rida-e-Fatima et al. (2019) propose a co-extraction model with refined word-embeddings (distinct vector representations to dissimilar sentiments) to learn the dependency structures of opinion sentences. In particular, the work employ a deep learning-based multilayer dual-attention model to exploit the indirect relation between the aspect and opinion terms. However, for empirical study, we employ an openly available low-dimensional (50, 100, 300) word-embeddings called *GloVe* (Pennington et al. 2014), a global log-bilinear regression model trained on Google News dataset.

4 DNN architectures

Having discussed the essential components of DNN and also about word-embeddings, now it is time to look at the major DNN architectures utilized in sentiment analysis. Based on the underlying structures and hyperparameters involved, most DNN models are categorized into CNN (Kim 2014; Salinca 2017; Krishnakumari et al. 2019), RNN (Schmidhuber 2015; Zhang et al. 2017; Liu et al. 2017b) and hybrid neural networks (Hassan and Mahmood 2018; Rehman et al. 2019; Sadr et al. 2019). The following subsections provide a brief account on the relevant theories underlying these models, distinguishable features, and potential challenges.

4.1 Convolution neural networks

CNN are inherently DFNN, which use convolution filters for specific features mapping (Kim 2014; Salinca 2017; Krishnakumari et al. 2019). They are primarily designed for working with computer vision applications (Litjens 2017). Eventually, they are widely used in other fields such as speech recognition, text mining, NLP, and sentiment analysis. In the following paragraph, the procedure of sentiment classification performed with a basic CNN model is elucidated.

The architecture, described in Fig. 2, is identical to the CNN structure given by Collobert et al. (2011). In this model, each row represents a *k*-dimensional word-embeddings similar to *Word2Vec* (Mikolov et al. 2013b) or *GloVe* (Pennington et al. 2014). Hence, a

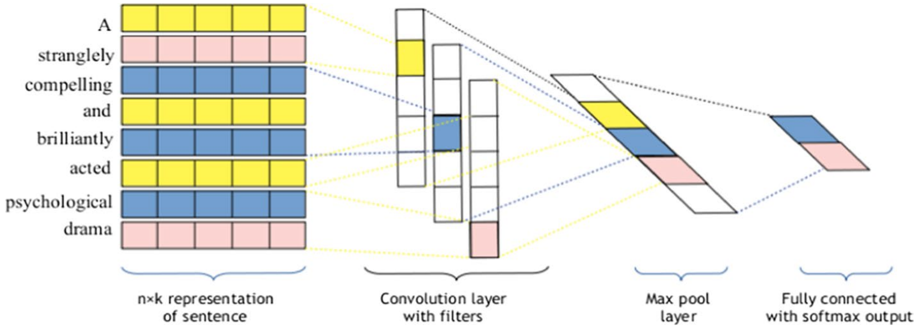

A
strangley
compelling
and
brilliantly
acted
psychological
drama
n×k representation of sentence
Convolution layer with filters
Max pool layer
Fully connected with softmax output

Fig. 2 CNN architecture for sentiment classification

sentence of length n words, each with d-dimensional word-embedding is represented as a $n \times d$ matrix and is provided as input to the network. If x_i is the word-embedding referring to i^{th} word, then a sentence of length n words is represented as the concatenation of all associated word-vectors in the sentence. In the convolution layer, a filter f_s is applied over a window of k terms to generate a convoluted feature vector c_i in a feedforward manner (Santos and Gatti 2014; Kim 2014; Salinca 2017). Thus, the vector c_i is computed as given in Eq. 3.

$$c_i = f\left(x_{i:[i+k-1]}\right) + b \tag{3}$$

where b is a bias vector and $f(\cdot)$ is an activation function such as *ReLu* or *tanh*. Subsequently, the filter is applied to all possible windows of words in the sentence to generate a feature map. These feature vectors are allowed to pass through the pooling process for concatenating them into a single vector of size $d \times 1$. Finally, the vector is submitted to a classifier (e.g., *softmax* layer), which reduces the vector into $m \times 1$ vector resulting in the probability of m classes. In CNN, convolution layers act as feature extractors that capture local features by controlling the region in the input space of the hidden layers to be local. It is due to the ability of CNN handling spatial local correlation among neurons of neighboring layers. This trait is crucial for sentiment classification, where it is required to identify the prominent local features for determining class membership.

4.2 Recurrent neural network

In RNN, the connections among neurons are sequentially interlaced to form a directed cycle (Elman 1990). Hence, this model is well known for sequential data processing, which uses its internal memory for a sequence of inputs (Sutskever 2012; Salehinejad et al. 2017; Majumder et al. 2019). RNN performs the same task on each element of an input sequence recurrently, and while considering the previous computations, it generates the current output. A simple prototype of RNN is demonstrated in Fig. 3. The unfolded network comprising cycles on the left side graph is folded towards the right side graph with a sequence of networks in three-time steps. However, the decision on time steps involved is dependent on the length of the input sequence.

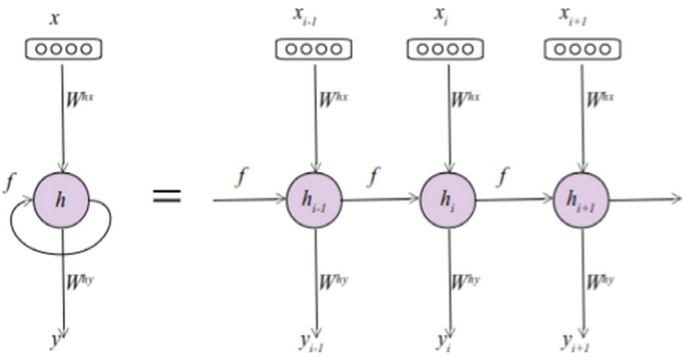


Fig. 3 RNN architecture

In Fig. 3, a hidden state h_t is determined using an input x_i at current time step t, and representations of previous hidden state h_{t-1}. Then, h_t can be mathematically denoted as in Eq. 4.

$$h_t = f\left(W^{hh}h_{t-1} + W^{hx}x_t\right) \tag{4}$$

where, W^{hh} and W^{hx} are the weights on the previous hidden state h_{t-1} and current input x_t respectively. The output y_t is the probability distribution over a given vocabulary at time step t and is provided by Eq. 5.

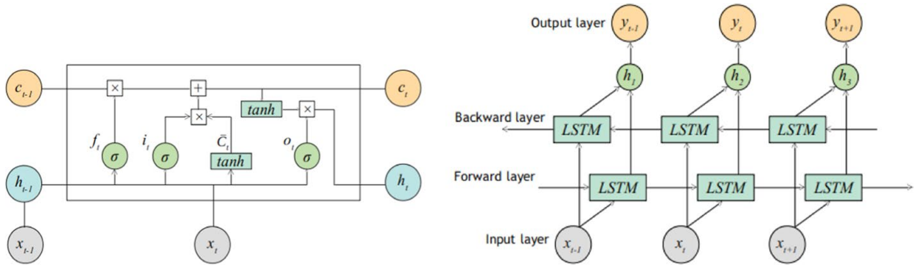
$$y_t = f\left(W^{hy}h_t\right) \tag{5}$$

The activation function $f(\cdot)$ in Eqs. 4 and 5 can be chosen suitably as discussed in Sect. 2. The hidden state h_t is interpreted logically as a memory of the network; hence, the result of y_t is dependent only on the memory h_t at time t, and weight matrix W^{hy}. Unlike feedforward networks that use different weights on different layers, RNN share the same weights across all the iterations (Ma and Ji 1998). However, the above formulation is suitable only for small and fixed-length input sequences, but for arbitrary long sequences, it suffers from vanishing gradient or exploding gradient problems (Nair and Hinton 2010). To overcome the challenges associated with standard RNN, many researchers have proposed more improved versions of RNN.

4.3 Long short-term memory network

The long short-term memory network (LSTM) network is a variant of RNN with the learning ability of long-term dependencies (Zhang et al. 2015; Mousa and Schuller 2017; Schuurmans and Frasincar 2019). In contrast to RNN, LSTM units are composed of four essential components, namely, a cell, an input gate, an output gate, and a forget gate (Nguyen et al. 2012). During a stipulated time interval, the cell records information, and gates control the flow of information from and to the cell. Two states are observed in this process: a hidden state and a cell state. The cell state emulates memory of an LSTM cell, and the hidden state records result of this cell. Thus, a hidden state and a cell input together decide the information stored in the cell, i.e., to discard or to store new information.

A typical LSTM network is illustrated in Fig. 4a. Initially, at time step t, the forget gate that employs a sigmoid function (ρ) makes a decision on information to be recorded in the



(a) LSTM cell

(b) BiLSTM network

Fig. 4 LSTM cell and its variant BiLSTM architecture

cell state. It accepts the input x_t and receives the output of previous hidden state h_{t-1} to compute the probability of storing current information using Eq. 6.

$$f_t = \rho\left(W^f x_t + U^f h_{t-1}\right) \tag{6}$$

Subsequently, LSTM decides over the new information to be stored in the cell state. Moreover, the input gate (*sigmoid*) determines the values to be updated based on Eq. 7. On the other hand, *tanh* function generates a candidate vector $\bar{C}_t$, which is further combined with a cell state to bring an update to the current cell state. Equation 8 is used to generate the vector $\bar{C}_t$.

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

$$\bar{C}_t = \tanh\left(W^n x_t + U^n h_{t-1}\right) \tag{8}$$

Then, Eq. 9 is utilized to update the previous cell state C_{t-1} into a new cell state C_t.

$$C_t = f_t \times C_{t-1} + i_t \times \bar{C}_t \tag{9}$$

In Eq. 9, the forget gate f_t controls the gradient passes through it and allows memory operations that help in alleviating vanishing or exploding gradient problems observed in RNN. Finally, the *sigmoid* function at the output gate decides which portion of a cell state to return using Eq. 10. The resultant cell state C_t is subjected to *tanh* function and gets multiplied with another *sigmoid* function o_t to yeild the hidden state h_t, as illustrated in Eq. 11.

$$o_t = \rho\left(W^o x_t + U^o h_{t-1}\right) \tag{10}$$

$$h_t = o_t \times \tanh\left(C_t\right) \tag{11}$$

It is often beneficial to consider both the future as well as past contexts in modeling sequential tasks. However, the above LSTM network can handle sequences only in temporal order while ignoring the future context (Zhou et al. 2016). To overcome the limitations of standard LSTM, a bidirectional LSTM (Bi-LSTM) is proposed (Schuster and Paliwal 1997). Bi-LSTM network extends the regular LSTM by adding a second layer, where the hidden-to-hidden linkages allow the information to flow in reverse temporal order, as depicted in Fig. 4b. Therefore, the model can be trained with information available from past and future contexts. For example, the output of ith word can be predicted through the element-wise sum of both forward and backward pass outputs as described in Eq. 12.

$$h_i = h_i^f + h_i^b \tag{12}$$

4.4 Long short-term memory network with attention mechanism

Although, LSTM and Bi-LSTM can address the problem of long-range dependencies, in practice, handling long-term dependencies are more challenging (Thireou and Reczko 2007; Plank et al. 2016; Gao et al. 2017; Akhtar et al. 2018; Pergola et al. 2019; Liu et al. 2019a). Hence, a technique called attention mechanism is introduced. The attention mechanism in DNN is inspired by the visual attention mechanism observed in human beings (Liu and Zhang 2017a). In the human visual attention mechanism, our eyes focus on a specific area of an object with higher intensity, while the attention on the surrounding region is generally lesser. Similarly, in natural language text, not all terms necessarily contribute to the context of the sentiment polarity. The attention mechanism allows the model to learn a specific portion of the input based on the present and past contexts, rather than encoding the entire source input into a fixed-length vector (Baziotis et al. 2017; Liu et al. 2019b; Wei et al. 2019). The general idea of applying attention mechanism to an LSTM network is demonstrated in the following paragraph. Figure 5a, b depict the attention mechanism in LSTM and Bi-LSTM, respectively.

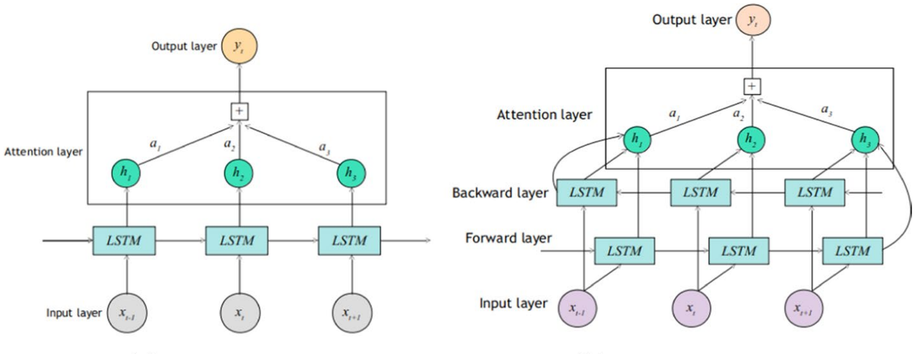
For a given input sequence $x = (x_1, x_2, \dots, x_n)$, and an opinion target $y = (y_1, y_2, \dots, y_m)$ that occurs in the sentence with a subsequence of m terms out of x, then LSTM is used to capture the sequential information, and outputs a sequence of hidden vectors as mentioned in Eq. 13.

$$[h_1, h_2, \dots, h_n] = LSTM(n, \theta_{lstm}) \tag{13}$$

where n is the number of terms in a sentence, and θ_{lstm} is the number of hidden layers in LSTM network. However, the Bi-LSTM utilizes both the forward and reverse order sequential information through concatenating the hidden-layer outputs of each model. It obtains the word representations by summarizing information from both directions of words, and hence the representations incorporate better contextual information. More formally, Bi-LSTM contains the forward-LSTM (denoted as $\overrightarrow{LSTM}$), which reads feature sequences of n terms in forward direction, and backward-LSTM (denoted as $\overleftarrow{LSTM}$), which reads n terms in reverse order, and θ_{lstm} being the number of LSTM layers in both direction. Then, the outputs of Bi-LSTM are stated in Eq. 14, and the combined representation can be obtained using Eq. 12.

$$\begin{aligned} h^f &= \overrightarrow{LSTM}\left(n, \theta_{lstm}\right) \\ h^b &= \overleftarrow{LSTM}\left(n, \theta_{lstm}\right) \end{aligned} \tag{14}$$

Now, the attention layer assigns a weight a_i to each term in the sequence of LSTM/Bi-LSTM to compute the context information r_i for each sequence, as the weighted sum of all terms in the sequence with Eq. 15. Further, the attention weights a_i are computed using Eq. 16.



(a) LSTM with attention

(b) Bi-LSTM with attention

Fig. 5 LSTM and Bi-LSTM architectures with attention

$$r_i = \sum_{i=1}^{n} a_i h_i \tag{15}$$

$$a_i = \frac{exp\left(f_s\left(h_i, t\right)\right)}{\sum_{j=1}^{n} exp\left(f_s\left(h_j, t\right)\right)} \tag{16}$$

In Eq. 16, $f_s(\cdot)$ is context-similarity function that captures the semantic association between each word and the target, where t is the target representation, and both are determined using different strategies (He et al. 2018) as listed in Table 1. Finally, each sentence representation r_i is fed to an output layer for predicting the probability distribution p over sentiment labels on the target t using Eq. 17.

$$p = softmax(W^o r_i + b^o) \tag{17}$$

On the other hand, the self-attention networks or intra-attention networks (Cheng et al. 2016; Lin et al. 2017; Vaswani et al. 2017; Shaw et al. 2018; Shen et al. 2018) have found to be useful for capturing global dependencies. Self-attention networks estimate attention weights among each pair of tokens in a single sequence, thus, they can capture long-range dependencies spontaneously as compared to standard RNN. The attention is applied to each sequence position by generating a query, value, and key representations of the source sequence. Then, the input sequence $x_i = (x_1, x_2, \dots, x_n)$ gets transformed into $y_i = (y_1, y_2, \dots, y_n)$, which includes x_i coupled with relative information, referring to other positions in x. Mathematically,

$$y_i = \sum_{j=1}^{n} a_{ij} x_j W^v \tag{18}$$

$$a_{ij} = \frac{exp(f_s(x_i W^Q, x_j W^K))}{\sum_{k=1}^{n} exp(f_s(x_i W^Q, x_j W^K))} \tag{19}$$

In Eqs. 18 and 19, W^V, W^Q, W^K are the learning parameters, and $f_s(\cdot)$ is a content-based function such as those used in Table 1. For a scaled dot product, the attentions for the source sequence are computed in parallel through the grouping of queries, keys, and values in Q, K, V matrices as in Eq. 20.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{20}$$

Table 1 Content-based functions for attention models

Content-based function	Equation
Concat product (Bahdanau et al. 2014)	$V_a^T \tanh(W^a[h, t])$
Dot product (Luong et al. 2015)	$h^T t$
General (Luong et al. 2015)	$h^T W^a t$
Scaled dot product (Vaswani et al. 2017)	$\frac{h^T t}{\sqrt{d_k}}$

4.5 Gated recurrent unit

In sequence modeling tasks, it is often useful to build the recurrent units that can capture dependencies adaptively at different time scales. Cho et al. (2014) propose a more simplified version of LSTM called gated recurrent unit (GRU) capable of learning dependencies adaptively. GRU consists of gating units for controlling the flow of information inside the unit without having an explicit memory cell. It combines the forget and input gates of LSTM into a single gate called update gate and extended with an additional reset gate (Jabreel and Moreno 2017). Thus GRU network appears to be simpler than the standard LSTM networks. The architecture of a typical GRU is demonstrated in Fig. 6.

The activation h_t of GRU at time t is a linear interpolation between the previous activation h_{t-1} and the candidate activation $\bar{h}_t$. Hence, h_t can be written formally as in Eq. 21.

$$h_t = \left(1 - z_t\right)h_{t-1} + z_t\bar{h}_t \tag{21}$$

The factor by which the unit updates its activation is decided by z_t and is computed using Eq. 22. The candidate activation is then computed using Eq. 23 as follows.

$$z_t = \rho\left(W^z x_t + U^z h_{t-1}\right) \tag{22}$$

$$\bar{h}_t = \tanh\left(W^t x_t + r_t \times h_{t-1}\right) \tag{23}$$

In Eq. 23, r_t refers to a rest gate and $r_t \times h_{t-1}$ is an element-wise product, and thus, r_t is determined using the following Eq. 24.

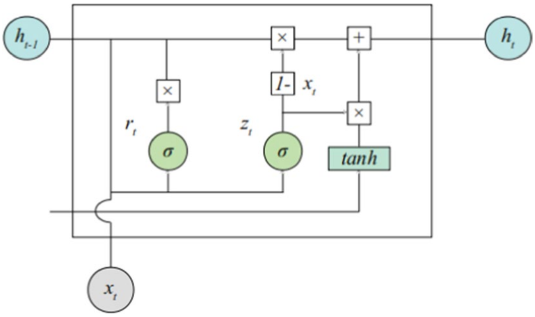
$$r_t = \rho\left(W^r x_t + r \times h_{t-1}\right) \tag{24}$$

The reset gate is set to off state when $r_t == 0$, and allows the unit to forget the previous activation. Since GRU uses fewer parameters they are comparatively faster than LSTM networks (Tian et al. 2018).

4.6 Deep recurrent belief networks

Deep recurrent belief networks (DRBN) are a group of algorithms that use probabilistic models to predict outputs (Zhou et al. 2014; Chaturvedi et al. 2016; Jiang et al. 2018; Rizk et al. 2019). DRBN contains a stack of restricted Boltzmann machines (RBM), which receives low-level representations as input and encodes them using unsupervised learning.

Fig. 6 GRU architecture



These networks are composed of binary latent variables and may take both directed and undirected layers. The first RBM essentially contains undirected links; however, remaining RBM always follow directed connections. Unlike RBM, nodes in a DRBN do not interact simultaneously within their layer. However, a network of symmetrical weights are used to connect different layers. The nodes present in the hidden layer acts as a visible layer to those nodes succeeding it. Furthermore, these nodes discover the correlations among data samples. Mathematically, DRBN learns the joint distribution between an observed vector x and l hidden layers h^k as presented in Eq. 25.

$$P\left(x, h^1, \dots, h^l\right) = \left(\prod_{k=0}^{l-2} P\left(h^k \mid h^{k+1}\right)\right) P\left(h^{l-1} \mid h^l\right) \tag{25}$$

where $x = h^0, P\left(h^k \mid h^{k+1}\right)$ is a conditional distribution for the visible units conditioned on the hidden units of the RBM at level k, and $P\left(h^{l-1} \mid h^l\right)$ is the visible-hidden joint distribution in the top-level RBM. The general architecture of DRBN is demonstrated in Fig. 7.

Moreover, training DRBN is often conducted through greedy techniques. They involve sequential and greedy training to get the optimal choice at each layer, which ultimately finds a global optimum. For instance, Hinton et al. (2006) utilize the weights of unlearned layers to train DRBN by contrastive divergence analysis. Learning of weights happens through the first layer and iterates until all layers are trained. However, learning long-term delays in DRBN is challenging due to the problem of exploding gradients with an increase in delay. To alleviate this problem and increase the accuracy of learning time-delays, Chaturvedi et al. (2016) propose the Gaussian networks with time-delays to initialize the weights of each hidden neuron. In particular, they employ dynamic Gaussian Bayesian networks on training examples with Markov Chain Monte Carlo to learn the initial weights of each hidden layer.

4.7 Memory networks

Memory networks (Graves et al. 2014; Weston et al. 2014; Sukhbaatar et al. 2015; Kumar et al. 2016) are a class of neural networks with a memory component that can be read and written for prediction, inference, and reasoning. These networks typically consist of an array of vectorized objects called memories and several computational modules including an input module that computes the feature maps, a generalization unit to update the

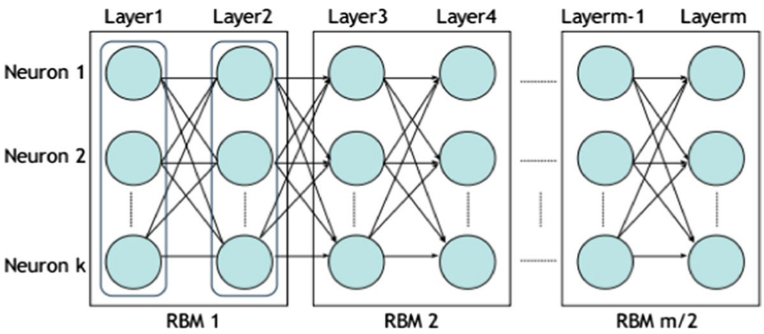


Fig. 7 DRBN architecture (Khanna and Awad 2015)

previous memory, an output module, and a response module to generate the final response (Kumar et al. 2016). The input data on memory networks consist of a history (list of previous inputs), a query (current input) and, a supervised label. Furthermore, the inputs on history and query are represented by word-embedding of fixed-size vector e. Then, each history input vector $x_1, x_2, \dots x_n$ of size e is embedded into memory vectors $m_1, m_2, \dots m_n$ of size d as embedding matrix $W^e \in R^{e\times d}$. Similar to history input embedding, an output embedding matrix $(c_1, c_2, \dots c_n)$ is also computed. But, the query vector p_i, which represents the current input is different from history input and is obtained specially using Eq. 26. Then, C^e is embedded into a vector u of size d.

$$p_i = softmax\left(u^T m_i\right) \tag{26}$$

Furthermore, p_i is used to find an average representation of all the (output) memory embeddings, which considers their relation with the query as given in Eq. 27.

$$o = \sum_i p_i c_i \tag{27}$$

There are many possibilities at this time, but the one selection is to embed the vector o into the space of all possible results and then apply *softmax* according to Eq. 28.

$$\hat{a} = softmax(W^e(o + u)) \tag{28}$$

4.8 Transformer networks

Transformers are a kind of neural architecture designed to work on a problem of sequence transduction, or a neural machine translation (NMT) (Vaswani et al. 2017). They are similar to traditional NMT encoder-decoder systems, as the encoder processes the source language sequence, and the decoder utilizes the output of encoder to construct a sentence in the target language. However, the encoder doesn't generates a fixed-length vector for representing the input sequence as opposed to RNN. But, an output of the same input length is produced, except the one which is transformed by the encoder self-attention layers. The working procedure of a transformer network is elaborated in Fig. 8.

The transformer receives an input word-embedding and concatenates the positional information using different positional encoding schemes. This step is essential as the self-attention mechanism doesn't inherently model the positioning of the input sequence (as opposed to RNN and CNN). Several methods have been proposed to address this problem (Luong et al. 2015). Mainly, there are three important techniques employed, (i) adding sinusoidal positional encodings (ii) learned positional encoding to input embeddings, (iii) using relative positional representations in the self-attention mechanism. Further, the input sequence is allowed to pass through n identical encoder layers, each consisting of two sublayers of multihead self-attention and feedforward layers. The output of multihead self-attention is computed as the weighted sum of all values, where the weight assigned to each value is obtained through the dot product of the query containing all keys. Furthermore, the connector that combines encoder and decoder also ensures the encoder input sequence considering the decoder input sequence up to a given point. Finally, for training, it uses some deep learning concepts such as residual connections (He et al. 2016) or layer normalization (Ba et al. 2016). For example, the residual connections technique permits the positional encoding signal to propagate deeper within the network. These operations wrap

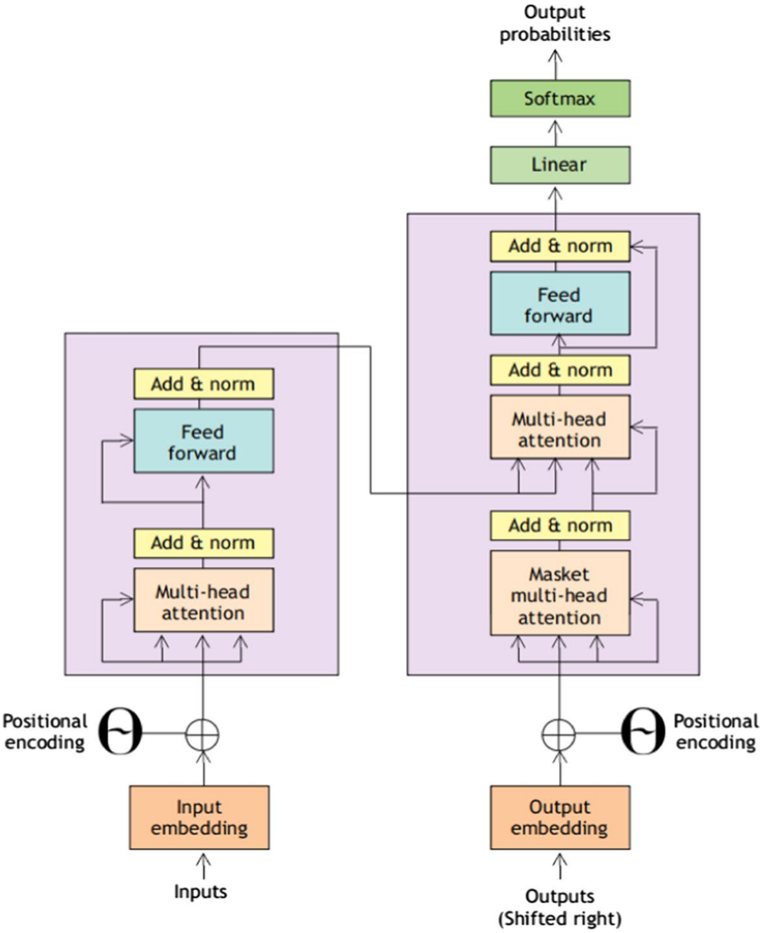


Fig. 8 Transformer architecture (Vaswani et al. 2017)

each sublayer, and the output of each sublayer is denoted as $layernorm(x + sublayer(x))$, where $sublayer(x)$ can be either multihead or feedforward layer.

4.9 Recursive neural network

Recursive neural networks (RcNN) are non-linear adaptive models designed to work on structured inputs (directed positional acyclic graphs) (Socher et al. 2013; Irsoy et al. 2014a). For a given structured input, transformations are applied recursively to obtain the representations for higher-level activation learned from its descendants (Irsoy and Cardie 2014b; Wadawadagi et al. 2019b; Wadawadagi and Pagi 2020b). The above theory can be applied generally to any structured input. However, the usage of RNN is restricted to positional binary trees (e.g., parse trees) as they are simple and efficient in modeling natural language text. Thus, for a given binary parse tree of *n*-gram sentence initialized each leaf node with a word vector, RcNN then computes the parent representations using the Eq. 29.

$$x(v) = f(W^l x(l_v) + W^r x(r_v) + b) \tag{29}$$

where l_v and r_v are left and right nodes of v respectively, W^l and W^r are the synaptic weights that bind left and right children to the parent v, and b is a bias vector. Furthermore, it is noted that representations at the terminal and non-terminal nodes lie in the same space, provided W^l and W^r are square matrices, and $\{l_v, r_v\} \in W^e$ (W^e is $d \times V$ dimensional word-embedding matrix) are not distinguishable from the terminal and non-terminal nodes. Then, the parent representations are obtained through the recursive application of transformations to left and right sub-phrases in a bottom-up manner that are in the same semantic space. Finally, a problem specific output layer is determined according to the Eq. 30.

$$y(v) = g(W^g x(v) + c) \tag{30}$$

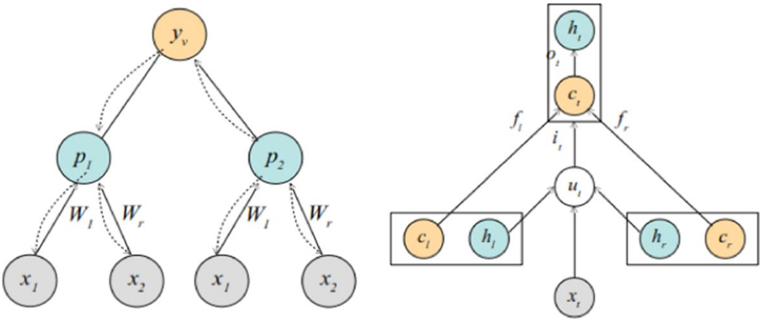
where W^g is the output weight matrix, c is the bias vector to the output layer, and $y(v)$ is a class prediction for the parent node v. Meanwhile, external errors that occurred at y are back-propagated from the root node towards its descendants (Liu et al. 2014).

In the above model, both terminal and non-terminal nodes are assumed to be similar for many classification problems. However, applications based on parse tree need discrimination among terminal and non-terminal nodes, because, terminal and non-terminal nodes have different representations. In order to decide whether the incoming edge emanates from the terminal or non-terminal nodes, a simple parameterization of the weights W is employed. More technically, this idea is illustrated in Eq. 31.

$$z(v) = f(W^l_{l_v} \times z_l(l_v) + W^r_{r_v} \times z_r(r_v) + b) \tag{31}$$

In Eq. 31, if $z(v) = x(v)$, then v is a leaf node, and $z(v) \in \chi$, a vector space of lexical terms, and otherwise $z(v) \in \varphi$, a vector space of phrases. If v is terminal node then $W^v = W^{xz}$ otherwise $W^v = W^{zz}$. The weight matrix W^{xz} is related to transformation matrix that connects word to phrase space and W^{zz} connects phrase space to itself. Additionally, the dimension of W^{xz} is defined to be $|x| \times |z|$ and W^{zz} is $|z| \times |z|$, indicates that even a large pre-trained word vectors with a small number of hidden units can be used to train a network without a quadratic dependence on the word vector dimensionality $|z|$. Figure 9a gives a typical structure of RcNN.

Furthermore, an extension to standard LSTM known as Tree-LSTM for tree-structured network topologies is discussed (Tai et al. 2015). In contrast to LSTM, which constitutes its hidden state from the input sequence at the current time step and the hidden state of previous activation, the Tree-LSTM composes its states from the input vector and the hidden state of arbitrary descendants. The structure of a simple Tree-LSTM model for composing a parent memory cell is depicted in Fig. 9b. Each unit in Tree-LSTM consists of an input gate i_t, an output gate o_t, a memory cell c_t and hidden state h_t. Unlike the LSTM unit, which contains a single forget gate, the Tree-LSTM unit contains one forget gate for each descendant. The updates to gating vectors and memory cells are dependent on the states of possibly many child units. This fact enables the Tree-LSTM units to perform selective information access through the child units. However, the input vector to the Tree-LSTM unit is dependent on the tree-structure used for the network.



(a) Recursive neural networks **(b)** Tree-LSTM

Fig. 9 RcNN and its variant Tree-LSTM

4.10 Convolutional recurrent neural networks

There is an increased interest in developing hybrid models that perform information fusion, thereby overcoming the constraints of employing a single network model. In this direction, numerous techniques have been reported in the literature for sequence modeling using hybrid models. For instance, a framework that combines the features of CNN and RNN into a single model for sentiment classification is proposed (Lai et al. 2015). Here, the model employs LSTM as an alternative pooling technique for capturing long-range dependencies across the input sequence. The fact that CNN can extract local features from the input sequence and LSTM can handle long-term dependencies; it would be beneficial to combine these features for any sequence modeling task (Wang et al. 2016). The architecture of the joint model termed as convolutional recurrent neural networks (CRNN) consists of several layers, including a convolution and pooling layer, concatenation layer, LSTM layer followed by a fully connected layer with softmax output (Hassan and Mahmood 2018). The model architecture is shown in Fig. 10. The role of a convolutional layer is to compute the matrix-vector product of each window of size w over successive windows in the sentence representation. However, to extract local features (n-gram feature vector), the weight matrix is kept constant for all word windows of the sentence. The max and average pooling operations are performed over the output of the convolutional layer that transforms the feature map to its half. The LSTM layer receives representations from a convolutional layer in sequential order and extracts global features to learn the long-term dependencies. The output of LSTM, $h_t \in R^n$ forms the penultimate layer and encodes the entire input sentence. Then, the features obtained from LSTM are transferred to a fully connected *softmax* layer to obtain the probability distribution of all the classes over given vocabulary.

4.11 Quasi recurrent neural networks

RNN and their variants are restricted to handle the issues related to text involving long sequences such as in document classification, where the features at different parts of the document cannot be handled in parallel. Hence, the problem of long sequences can be addressed using a new type of network called quasi recurrent neural networks (QRNN) (Bradbury et al. 2016). Similar to CNN, each layer of a QRNN is composed of two

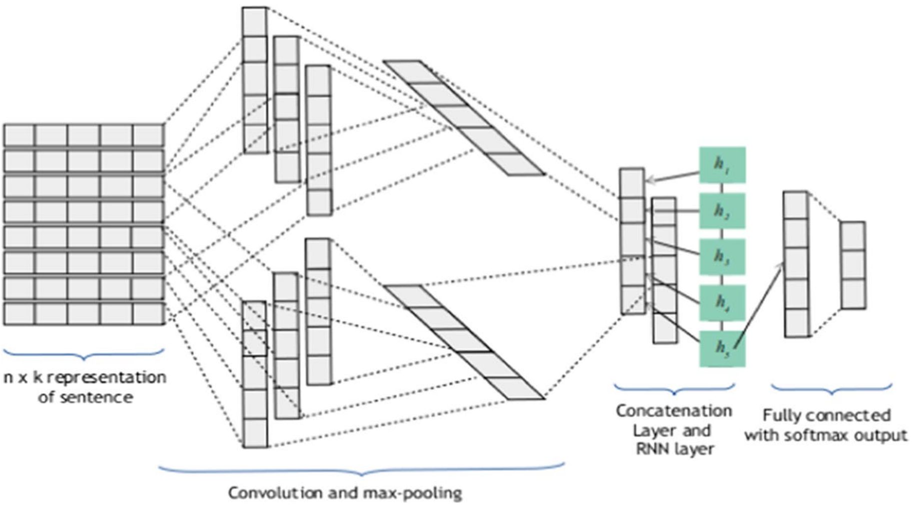

n x k representation of sentence
Convolution and max-pooling
h_1
h_2
h_3
h_4
h_5
Concatenation Layer and RNN layer
Fully connected with softmax output

Fig. 10 CRNN architecture

constituents, a convolutional component and a pooling component. The convolutional component performs fully parallel computation across mini-batches and spatial dimensions while the pooling component performs parallel computation across mini-batch and feature dimensions. For an input sequence x_i of d-dimensional vectors, m filters are employed to perform convolutions in the time-step t that obtains an output sequence z_i of m-dimension. Moreover, separate filters are used to obtain the sequences of vectors for element-wise gates, that are required for the pooling operation. Thus, the element-wise gates of standard LSTM cells can be employed for the pooling operation. Besides, RNN with its distinct capabilities can also be sought in combination with two or more QRNN layers stacked to approximate more complex functions. However, QRNN exhibits similar performance to the RNN, but it is faster than the highly optimized GPU-LSTM implementation.

4.12 Capsule networks

Yet another important neural architecture known as capsule networks (CNet) is receiving a lot of attention due to its performance gains on sequence mining tasks (Sabour et al. 2017; Paik et al. 2019; Patrick et al. 2019; Chen and Lee 2019; Chen and Qian 2019b; Nguyen et al. 2019). In neural networks, it is essential to model the spatial relationships among various parts of the sequence, rather than to model the co-existence disregarding relative positioning. This is accomplished by capturing spatial information using a set of nested layers called capsules and implementing an algorithm called routing-by-agreement. Thus, substituting conventional CNN by CNet can capture the spatial relations among the features and has a higher potential for efficient representation and understanding of the given problem. A simple CNet applied for sentiment classification is demonstrated in Fig. 11.

In CNet, each capsule describes attributes of partial entries, and defines semantics in the larger space by expressing the entities with a vector rather than a scalar. The document $D \in R^{l \times e}$ is given as input to CNet, where l is the length of the document, and e is the word-embedding size. The second layer utilizes convolutions obtained through different filter size f_s and operates iteratively over each row of W^e to generate the feature map

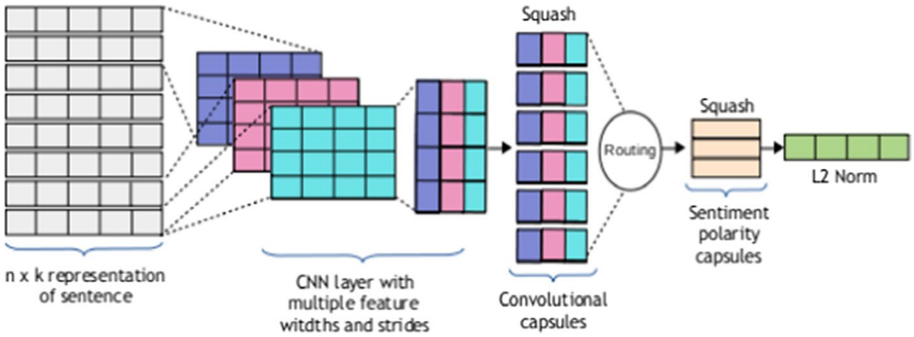

Squash
Routing
Squash
L2 Norm
n x k representation of sentence
CNN layer with multiple feature witdths and strides
Convolutional capsules
Sentiment polarity capsules

Fig. 11 A typical CNet employed for sentiment classification

$q = (q_1, q_2, \dots, q_k)$, in which $q_i = g(f_s \cdot W^e_{i,:} + b)$, where b is a bias term, and g is a non-linear activation function such as *ReLu*. The third layer is a convolutional capsule layer, which is channelized into m-dimensional convolutional capsules. Since the classifier is locally connected to the feature map, it is hard to handle variations in transformation for the classifier. Hence, some routing techniques need to be employed for transferring knowledge from the convolutional capsule layer to the sentiment capsule layer. For instance, Sabour et al. (2017) used an iterative routing process, which updates the weights of coupling coefficients to determine which lower capsules have to be directed to upper capsules. The coupling coefficient, C_{ij}, is presented in Eq. 32.

$$C_{ij} = \frac{exp\left(b_{ij}\right)}{\sum_k exp\left(b_{ik}\right)} \tag{32}$$

where $i \in [1, a], j \in [1, k]$, a is the number of capsules in the first layer, k is the number of classes. The *softmax* output of b_{ij} is updated for every routing iteration and is determined by the degree of similarity between the lower and upper capsules. The predicted vector is expressed by a matrix operation between the weight matrix W_{ij} and h_i. The capsule then performs the non-linear squashing function to produce a vector output $e \in R^{d\times1}$ as in Eq. 33, and, the *softmax* output b_{ij} is determined using Eq. 34.

$$\begin{aligned} e &= squash(s_j) \\ s_j &= \sum_i c_{ij}\hat{h}_{j|i} \end{aligned} \tag{33}$$

$$\begin{aligned} v_j &= \frac{\left\|s_j\right\|^2}{1 + \left\|s_j\right\|^2}\frac{s_j}{\left\|s_j\right\|}, \\ b_{ij} &= b_{ij} + v_j.\hat{h}_{j|i} \end{aligned} \tag{34}$$

Additionally, many variants of CNets have been studied in the literature. For example, Wang et al. (2018) propose a simple CNet based on RNN for sentiment polarity detection. In this network model, each capsule is constructed to carry sentiment category along with probability and reconstruction information. Furthermore, this knowledge is utilized to determine sentiment polarity of an opinion sentence. Similarly, Du et al. (2019a) proposed a hybrid model based on CNet for sentiment analysis of short text messages. The proposed model utilizes both forward and backward information to model the expression ability of the short text messages. Through this, the model attains a simple network structure with improved performance and requires less training time. However, in (Du et al. 2019b), a CNet model to construct vector-based feature representation and features clustering using an expectation-maximization routing algorithm is presented. An interactive attention mechanism is employed in the capsule routing procedure to model the semantic relationship between aspect terms and context. On the other hand, Fentaw and Kim (2019) propose a parallel convolution computational technique to feed the capsule layer for the high dimensional feature reduction and to encode different features with varying kernel sizes. Chen and Lee (2019) propose TransCap, an improved CNet for transferring document-level knowledge to aspect-level sentiment prediction. They devised an aspect routing technique to encapsulate the sentence-level semantics into semantic capsules from both aspect-level and document-level data. Furthermore, it extends the dynamic routing to adaptively couple the semantic capsules with the class capsules under the transfer learning framework. On the contrary, Xu et al. (2020) propose a variant of CNet termed CAPSAR aimed to enhance ABSA by identifying the potential aspect terms of the predicted sentiment polarity when the aspect terms of a test sentence are not defined. Here, sentiment categories are denoted by capsules and aspect term information is injected into sentiment capsules through a sentiment-aspect reconstruction procedure during the network training. Moreover, developing capsule networks for sentiment analysis is limited by scalability, reliability and generalizability issues. For instance, current techniques are inadequate to scale toward larger output spaces and involve less reliable routing processes. To address these issues, Zhao et al. (2019) devise the routing process as a proxy problem, which minimizes a cumulative negative agreement score for estimating the performance of routing processes at the instance level. Furthermore, to increase instance-level convergence and improve the reliability of the routing process, an adaptive optimization algorithm is proposed to self-adjusting the number of iterations for each sample. Additionally, the authors present capsule compression and partial routing to accomplish higher scalability of capsule networks against datasets with large output spaces.

4.13 Ensemble networks

The most established and commonly used category of neural network learning is stacked ensemble networks. In a stacked ensemble, different network models are trained with the same configuration and different weight initialization on the same input dataset (Ekbal and Saha 2014; Akhtar et al. 2017, 2019, 2020). The purpose is to allow each model to make a prediction and the final prediction can be obtained through the average of all predictions or by majority voting. Furthermore, the number of models that can be accommodated in the stack is often less than five or up to ten for pre-trained models. This is due to the computational cost incurred in training various models and declining results in performance by adding more models to ensembles. Ensemble techniques have been studied extensively, and different forms of ensembles have come into existence. However, it is also possible to develop new ensembles by making the choices on combinations of datasets, ensemble models, and network parameters. The literature reveals that the traditional approaches to the ensemble use boosting (Freund et al. 1996; Xiao et al. 2013), bagging (Breiman 1996; Xiao et al. 2013), and voting (Ekbal and Saha 2011). Besides, an ensemble technique based on a multilayer perceptron (MLP) is also studied (Akhtar et al. 2020). This kind of ensemble consists of a stack of DNN models each one trained on different word-embeddings. The goodness of each word-embedding can be estimated through a stacked denoising auto-encoder network. Finally, the outcomes of different DNN models are ensembled through the MLP network. Regarding activation functions, *ReLu* can be utilized effectively between two hidden layers, and for the output prediction layer, either *tanh* or *Softmax* can be used. A variant of the above ensemble network that combines a feature-based supervised model with multiple DNN models is also proposed. In such a network, feature-based learning uses a diverse set of features to train the supervised model and the DNN models are trained with different word-embeddings. The goodness of the ensembled models are evaluated by tuning various network parameters. A typical MLP-based ensemble network is demonstrated in Fig. 12.

Nevertheless, the research on sentiment analysis using DNN models continue to evolve with the advancement of modern architectures to confront new challenges. Table 2 shows a summary of different DNN architecture with their merits and demerits applied to sentiment classification tasks.

5 Experimental setup for empirical study

A series of experiments have been conducted to study the empirical performances of different DNN architectures. From the selection of datasets to the choice of activation units and network training, different combinations have been tested to obtain the optimal hyperparameters of the networks. The following subsections present the arrangement of experiments and steps involved in the process of training and evaluating the DNN.

5.1 Datasets

The performance of various architectures is evaluated on sentiment classification task over benchmark datasets including, IMDB review dataset (IMRD) (Maas et al. 2011), SemEval

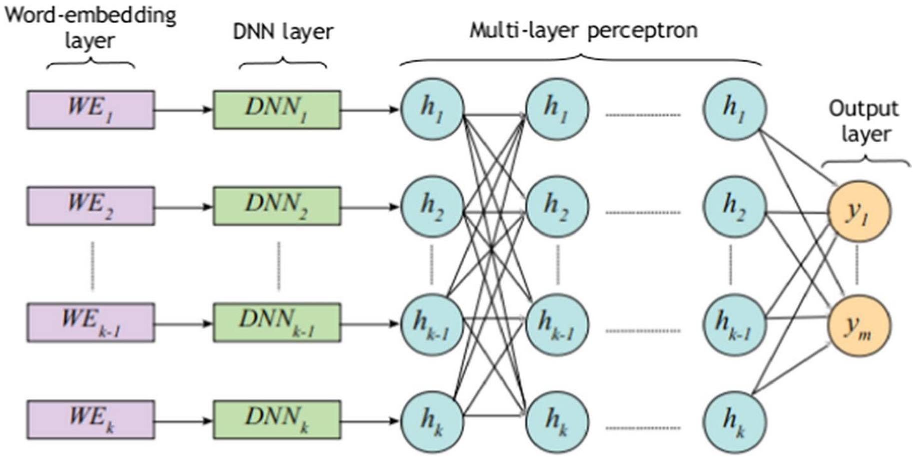


Fig. 12 A typical ensemble network for sentiment classification

Table 2 A summary of DNN models with their merits and demerits

Category	Model	Methodology	Merits	Demerits
Convolutional networks	CNN	A series of layers process the input sentence, and every layer transforms knowledge to another through activation functions	Computationally inexpensive and reduces complexity to a great extent without losing the structure of the data	It cannot encode the position and context of the input sentence
Recurrent networks	RNN	Uses past outputs as input with several hidden states and the same parameters as it performs the same task on hidden layers to produce the output	Capability to process input sequence of any length without an increase in model size, and considers the previous data with shared weights across time	Computationally sluggish, suffer from learning information of a long period, and cannot hold any future input for the current state
	LSTM	The information flows through a mechanism known as cell states, and hence, selectively memorize or forget information	The constant error back-propagation within memory cells results in the ability to connect long time intervals. Handles vanishing and exploding gradient problem and require less parameter fine-tuning	Need more resources to train the model as the number of memory cells is linked to the size of the recurrent weight matrices
	BiLSTM	An extension of traditional LSTM that connects two hidden layers of opposite directions to the same output	Efficiently utilizes past and future representations for a specific time frame	The input sequence must be complete before it can be processed, and hence, makes them unsuitable for real-time applications
	GRU	It is similar to a LSTM with forget gate, but has fewer parameters than LSTM	Computationally more efficient than LSTM, without having a memory unit, it exposes the full hidden content.	Their inherent sequential behavior makes them unfit for increasingly massive datasets and exponentially slower to train
	RNN + Att	Learns a specific portion of the input based on the present and past contexts, rather than encoding the entire source input into a fixed-length vector	Highly parallelizable computation, significantly reduces training time, and flexibility in modeling dependencies	It adds more weight parameters to the model and computes a separate context vector for every step of decoder, which increases training time
	RNN + SelfAtt	Estimates attention weights among each pair of tokens in a single sequence, and captures long-range dependencies	Reduces the training parameters, highly parallelizable computation, and models the words positional information in the input sequence, which helps to distinguish output representation of the same word occurring in different positions in the sentence	Limited by their ability to model context-free languages, nor hierarchical structure, unless the number of layers or heads increases with input size

Table 2 (continued)

Category	Model	Methodology	Merits	Demerits
	DRBN	Includes a stack of restricted Boltzmann machines that receives low-level representations as input and encodes them using unsupervised learning	Require small labeled dataset and training time is reasonably short on GPU enabled machines. Improved performance gain by adding layers compared to a multilayer perceptron	Expensive in terms of hardware utilization for high dimensional datasets
Recursive networks	RcNN	Transformations are applied recursively to a structured input by learning representations for higher-level activation from its descendants	Powerful in learning hierarchical, tree-like structure, efficiently captures syntactic and semantic aspects of a text	Efficiency depends on how best the structure of a tree is; further, the tree structure of every input sample must be known at training time
	Tree-RNN	Constitutes its states from the input vector and the hidden state of arbitrary descendants	Two constitutes are structurally closer to each other even though they are far away in the input sequence. This phenomenon potentially captures long-distance dependencies	Sentences can be wrongly parsed when text is expressed in informal language. Sometimes, parsing is slow and domain-dependent
Hybrid networks	CRNN	Extracts local features from the input sequence using CNN and captures long-term dependencies from RNN	Captures spatio-temporal correlations and has the flexibility to be applied for a variety of vision tasks involving sequence prediction problems	Mapping spatial sequence inputs to static output classes, and mapping static spatial inputs to sequence outputs both are challenging
	QRNN	Consists of a convolutional layer to perform fully parallel computation across mini-batches and spatial dimensions, and a pooling component to perform feature dimensions	Allows for parallel computation across both timestamp and minibatch dimensions, enabling high throughput and good scaling along with sequences	For character-level language modeling, QRNN underperforms other models, because they need much more complex long-term interactions
Advanced networks	TransNet	The encoder generates an output of the same input length, except one which is transformed by the encoder self-attention layers	Maximize the amount of parallelizable computations, measured by the minimum number of sequential operations.	Confront the issue of context fragmentation, i.e., if the input sequence is longer than the predefined segment length, it needs to be separated, and information cannot be captured across segmentation

Table 2 (continued)

Category	Model	Methodology	Merits	Demerits
	CNet	Captures spatial information using a set of nested layers called capsules and implements a routing algorithm to transfer knowledge	Adequately captures the spatial relations among the features and has a higher potential for representation and understanding of given NLP tasks	Routing agreement without a bias term forms the representation of activation vectors a limited and hypothesized, which becomes a problem for deeper capsule networks
	MemNet	Consist of an array of vectorized objects called memories and computational modules such as input module, generalization, output, and response modules	These networks consider the historical data explicitly, with a dedicated vector representation for each history element, and effectively removes the chance to forget	The order in the sequence, as well as the local proximity of sequence elements is lost, which is crucial for most sequential inputs
	EnsemNet	Several network models are trained with the same configuration but different weight initializations on the same input dataset	Joining multiple networks into an ensemble improves the machine learning results compared to a single network	The main drawback is that, they require manual design and also deciding the number of neurons in the hidden layer and the number of DNNs in an ensemble

Challenge 2016 (SEC16) (Pontiki et al. 2014), sentiment labeled sentences (SLS) (Tang et al. 2015) and Stanford sentiment treebank (SST) datasets (Socher et al. 2013). The description of datasets used for experimentation is tabulated in Table 3. The selection of datasets is made with the intention of studying different traits related to the classification task.

5.2 Weight initialization

The initialization of parameter weights is the most crucial step in modeling any neural network (Seifert et al. 2017). However, the selection of proper initialization techniques significantly decreases the time required to converge when trained using some optimization techniques (Li et al. 2017; Yin et al. 2017). To illustrate the benefits of selecting weight initialization on training time, we use a random parameter initialization technique and also a pre-trained model. In random initialization technique, both the compositional matrices W, and word-embedding matrix W^e are initialized with real-valued random numbers, so that, representations of words and phrases are arbitrarily projected over the vector space without any training. In particular, the matrix W^e is initialized with a pre-trained model *GloVe* (Pennington et al. 2014) that gives the word analogies of the associated word categories.

5.3 Regularization of network

DNNs, when trained with a large number of parameters, confront the problem of overfitting. To overcome the problem of overfitting, a simple and powerful technique called dropout regularization (Srivastava et al. 2015) is used. In dropout regularization, certain units from the neural network are selected randomly and ignored during training. Initially, for each hidden layer, the probability of the dropout rate is fixed to 1.0; further, it is tuned between 0.5 and 1.0 over the validation set. Because dropped units are shared within the network, the same units of the hidden layer are dropped at each node. The hidden layers in CNN do not suffer from the problem of overfitting as they are not directly involved in the process of prediction. However, due to the large size of the datasets, keeping dropout on the fully-connected layers is beneficial.

5.4 Activation functions

The common choices of activation functions for empirical study suggest *sigmoid* function, *tanh* function, or *ReLu*. Their mathematical descriptions are as follows:

Sigmoid function: The *sigmoid* function is defined for real-valued inputs and transforms them into the values in the range between 0 and 1, as given in Eq. 35. The non-linearity of *sigmoid* has decreased its popularity due to the activations that quickly get drained at either tail 0 or 1. This sets gradients to nearly zero and interrupts the flow of information in the network. Accordingly, the results of the *sigmoid* function are not zero-centric and manifest undesired zig-zag dynamics in the gradient updates.

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{35}$$

Tanh function: To overcome the limitations of sigmoid, *tanh* function is often preferred in practice as its output range is zero-centered and −1 to 1 instead of [0, 1]. This is given by Eq. 36.

Table 3 Description of datasets

Dataset	Description	Applicability	Train	Dev	Test	Total
IMRD	It contains 50,000 movie reviews along with their associated binary sentiment polarity labels	Binary classification task with positive and negative classes	35,000	10,000	5000	50,000
SEC16	It is a collection of the tweets between the period of 2013–2015	Corse-grained sentiment analysis with three classes positive, negative and neutral	5355	1269	1779	8403
SLS	It contains sentences labelled with positive or negative sentiments, extracted from reviews of products, movies, and restaurants.	Binary classification problem with positive and negative classes	2100	600	300	3000
SST	It includes fine-grained sentiment labels for 215,154 phrases in the form of parse trees of opinion sentences	Fine-grained sentiment analysis with five categories	8544	2210	1101	11,855

$$f(x) = \frac{e^x - e^{-x}}{e^x - e^{-x}} \tag{36}$$

ReLu function: It is advantageous to apply rectifier units for training deep architectures without the pre-training step. Hence, a *ReLu* $f(x) = max\{0, x\}$ is used for hidden layers. *ReLu* avoids vanishing gradients problem as it returns 0 if it receives negative inputs (gradient zero), and restores current value for positive inputs.

Softmax function: However, for a given labeled input, the objective is to learn representations for fine-grained (five-class) sentiment polarities. To attain this, a standard *softmax* activation is used for the output layer that takes a node's vector $z(v)$ as input and produces prediction $y(v)$, according to the Eq. 37. The output of the *softmax* function is equivalent to a categorical probability distribution, i.e., the probability that any of the classes are true.

$$f(x) = \frac{e^x}{\sum_{k=1}^{K} e^x k} \tag{37}$$

5.5 Training the networks

DNN are often trained with optimization techniques that need a loss function to estimate the model error. The network parameters are optimized to minimize the loss in accordance with the output of a loss function using various optimization techniques. The loss is typically measured in terms of negative log-likelihood or residual sum of squares depending upon the learning task (Courbariaux 2016). In the following subsections, we discuss several optimization algorithms and other hyperparameters required for training the neural networks.

*Optimization techniques:*Gradient descent is the most popularly used algorithm to perform the optimization of neural networks. It provides a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \epsilon R^d$ through updating the parameters in opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ with respect to the parameters. The variants of gradient descent have been tried depending upon the size of data to compute the gradient of the objective function. For example, SGD performs parameter updates for each training sample x_i and label y_i as, $\theta = \theta - \eta \times \nabla_\theta J(\theta, x_i, y_i)$. In contrast, the AdaGrad (Duchi et al. 2011) changes the learning rate η at each time-step t for every parameter θ_i based on its past gradients that have been computed for θ_i. The update rule is given as in Eq. 38.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{g_t + \epsilon}} \cdot g_t \tag{38}$$

However, in AdaDelta (Zeiler et al. 2012) the sum of gradients is recursively used as a decaying average of all past squared gradients. Thus, its update rule is given by Eq. 39.

$$\theta_{t+1} = \theta_t - \frac{RMS[\Delta\theta]_{t-1}}{RMS\left[g_t\right]} \cdot g_t. \tag{39}$$

Furthermore, RSMProp is similar to the first update vector of AdaDelta, but the difference is RSMProp divides the learning rate by an exponentially decaying average of squared gradients (Ruder 2016). Yet another technique based on adaptive moment estimation called Adam (Kingma and Lei 2014) computes the adaptive learning rates for each parameter and keeps an exponential decaying average of past gradients m_t. The update rule is defined as in Eq. 40.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(1-\gamma)g_{t-1}^2 + \gamma g_t + \epsilon}} \cdot g_t \tag{40}$$

5.6 Loss functions

Errors on a validation set are measured during training and stopped (early stopping) if validation error does not get improved. It is calculated using loss function by matching the target value with predicted value returned by a neural network (Janocha and Czarnecki 2017). In this research work, several error functions have been examined in combination with different optimization techniques. The most popularly used loss function for binary classification is a binary cross-entropy (BCE). The BCE requires a single output value passed through a *sigmoid* function to classify the data into two classes, and gives the output in the range of 0 and 1. For multi-class problems, the most suitable loss function can be a categorical cross-entropy (CCE). In CCE, there must be the same number of output values as that of classes. The result of the final layer is passed through a *softmax* function to obtain each node's output as a probability between 0 and 1. On the other hand, the hinge loss is used for max-margin classification. For an intended output $t = \pm 1$ and a classifier score y, it is defined as $max(0, 1 - t \cdot y)$. Similarly, a margin ranking loss (MRL) is another popularly used loss function that sets a margin of some value between positive and negative samples.

6 Results and discussions

6.1 Empirical study of binary sentiment classification

Table 4 demonstrates the empirical results of various models and their corresponding hyperparameters applied for binary sentiment classification. The hyperparameters such as hidden size, mini-batch size, number of the epochs, learning rate, filter size, and loss functions are tuned on *dev* partition of the dataset. It is observed that attention-based LSTM & GRU and CNRR perform best on the IMRD dataset, and memory-based, attention-based LSTM and CRNN have shown better performance over the SEC16 dataset, while memory-based models stand favorable for SLS dataset. Generally, the attention-based RNN and hybrid models have shown excellent performance for all datasets as it was intended. However, RNN and their variants are well suited for encoding sequential information and phrases having long-term semantic dependency.

6.2 Empirical study of fine-grained sentiment classification

The models based on RcNN are considered for the fine-grained (five-class) classification as well as for binary classification. In particular, the performances of RcNN (Irsoy and Cardie 2014b) and Tree-LSTM are compared with two baselines, namely, the recursive neural tensor network (RTNN) (Socher et al. 2013) and matrix-vector RNN (MV-RNN) (Socher et al. 2012). However, the results of RTNN and MV-RNN are directly borrowed from the publications mentioned with the same ratio of train, dev, and test splitting. The networks

Table 4 Results of neural models for binary sentiment classification

Dataset	Network model	Hidden units	Mini batch	Hidden layers	Epoch	Optimizer	Loss function	Output activation	lr	F1-score
IMDB	CNN	20	64	$fs = 3$	8	RMSProp	BCE	Tanh	0.20	0.775
	LSTM	32	64	1	10	Adam	BCE	Sigmoid	0.001	0.812
	Bi-LSTM	32	128	2	10	Adam	CCE	Softmax	0.05	0.793
	GRU	32	32	3	20	RMSProp	HL	Sigmoid	0.10	0.816
	LSTM+Att	16	32	1	10	AdaGrad	CCE	Softmax	0.01	0.897
	LSTM+RAM	32	64	2	20	Adam	MRL	Softmax	0.001	0.847
	BiGRU+Att	32	128	2	30	AdaGrad	CCE	Softmax	0.01	0.902
	GRU+RAM	32	64	2	20	Adam	MRL	Softmax	0.01	0.919
	CNN+RNN	64	64	$fs = 4, hl = 2$	10	AdaGrad	BCE	Tanh	0.15	0.876
	QuasiRNN	32	32	$fs = 3, hl = 2$	30	AdaDelta	BCE	Sigmoid	0.001	0.782
SEC16	CNN	32	64	$fs = 3$	10	AdaDelta	BCE	Tanh	0.15	0.834
	LSTM	32	64	1	20	Adam	BCE	Sigmoid	0.005	0.843
	Bi-LSTM	32	128	2	25	AdaDelta	CCE	Softmax	0.10	0.789
	GRU	16	64	2	20	AdaGrad	HL	Sigmoid	0.1	0.829
	LSTM+Att	32	64	1	10	AdaGrad	CCE	Softmax	0.005	0.907
	LSTM+RAM	32	128	2	30	Adam	MRL	Softmax	0.05	0.879
	BiGRU+Att	16	64	2	15	AdaDelta	CCE	Softmax	0.005	0.899
	GRU+RAM	32	64	2	25	Adam	MRL	Softmax	0.01	0.871
	CNN+RNN	50	64	$fs = 4, hl = 3$	6	AdaDelta	BCE	Tanh	0.1	0.813
	QuasiRNN	32	64	$fs = 3, hl = 2$	20	RMSProp	BCE	Sigmoid	0.001	0.772

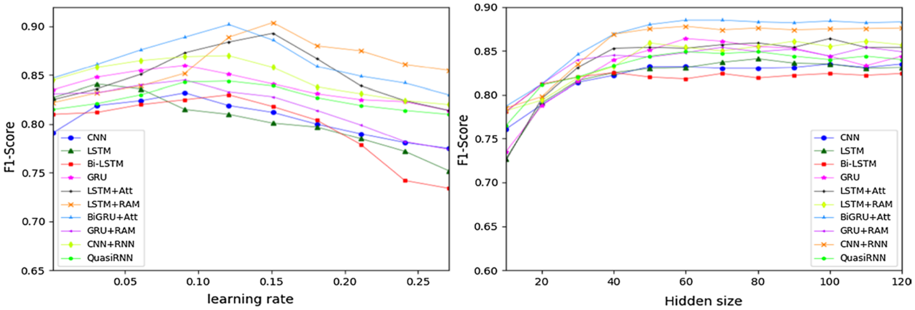
Table 4 (continued)

Dataset	Network model	Hidden units	Mini batch	Hidden layers	Epoch	Optimizer	Loss function	Output activation	lr	F1-score
SLS	CNN	32	32	$fs = 3$	15	RMSProp	BCE	Tanh	0.1	0.816
	LSTM	16	32	1	12	AdaGrad	BCE	Sigmoid	0.01	0.806
	Bi-LSTM	32	64	2	24	AdaGrad	CCE	Softmax	0.05	0.823
	GRU	32	64	3	10	AdaGrad	HL	Sigmoid	0.01	0.797
	LSTM+Att	32	192	1	15	AdaDelta	CCE	Softmax	0.001	0.889
	LSTM+RAM	64	128	2	30	Adam	MRL	Softmax	0.1	0.914
	BiGRU+Att	16	64	2	20	AdaDelta	CCE	Softmax	0.005	0.932
	GRU+RAM	32	64	2	25	Adam	MRL	Softmax	0.01	0.857
	CNN+RNN	32	64	$fs = 4, hl = 2$	12	AdaDelta	BCE	Tanh	0.15	0.863
	QuasiRNN	32	32	$fs = 4, hl = 3$	17	RMSProp	BCE	Sigmoid	0.001	0.805

Hint: *fs-Filter size, hl-Hidden layers*

Table 5 Results of fine-grained sentiment classification

Network model	Fine-grained	Binary
Tree-LSTM	43.49	83.65
MV-RNN	44.43	82.92
RcNN	45.54	84.43
RTNN	45.78	85.4



(a) Learning rate$(hs = 32, bs = 64, dp = 0.5)$ **(b)** Hidden size $(lr = 0.01)$

Fig. 13 Effect of tuning hyperparameters on network performance

are trained for fine-grained sentiment classification on an SST dataset, where a five-dimensional posterior probability output vector of a *softmax* layer is mapped to five-scale integer ranking *(1=very negative, 2=negative, 3=neutral, 4=positive, 5=very positive)*. The same five-dimensional probability vector obtained from the above step is further fused into a two-valued vector for binary prediction. However, for simplicity the hyperparameters are fixed for both the models as follows: *learning rate lr=0.01*, *batch size = 32*, *hidden units = 200* and *epoch = 200*. To update the training parameters, a diagonal variant of AdaGrad (Duchi et al. 2011; Socher et al. 2013; Irsoy and Cardie 2014b) is used. The empirical results are depicted in Table 5.

6.3 Effect of tuning hyperparameters on network performance

The performance of neural networks can be scaled by tuning their hyperparameters. To study this trait, two different hyperparameters, namely, the learning rate and hidden units, have been examined over the IMRD dataset. Figure 13a, b demonstrate the effect of tuning hyperparameters on the network performance. The experiment is being carried out for a wide range of values. The curves on the plot illustrate the rate at which the various models converge at different hyperparameter values.

6.4 Sentiment classification on example sentences

Table 6 depicts the sentiment prediction of several opinion sentences toward understanding the behavior of different neural networks. Here, the class labels of example sentences are unknown to the models and are taken randomly from different categories. In many instances, the models are not consistent; i.e., some predict positive while others predict

Table 6 Sentiment classification on example sentences

Type	Example sentences	CNN	LSTM	Bi-LSTM	GRU	BiGRU + Att	LSTM + Mem	CRNN	QRNN
Neutral	Their computer animated faces are very expressive	*pos*	*neg*	*pos*	*pos*	*pos*	*neg*	*pos*	*neg*
	A romantic comedy that operates by the rules of its own self-contained universe	*pos*	*pos*	*pos*	*pos*	*pos*	*pos*	*pos*	*neg*
Contradictory conjunction	I just loved the resolution and picture quality of the TV, but disappointed that it does not support Bluetooth speakers	*neg*	*neg*	*pos*	*pos*	*neg*	*neg*	*pos*	*neg*
	Admirable certainly, but not much fun to watch	*neg*	*neg*	*neg*	*pos*	*pos*	*neg*	*pos*	*neg*
Negated negative	The film is often filled with a sense of pure wonderment and excitement not often seen in today’s cinema du sarcasm	*pos*	*neg*	*neg*	*neg*	*pos*	*pos*	*pos*	*neg*
	The last 15 minutes of the movie was also not bad as well	*neg*	*neg*	*pos*	*pos*	*pos*	*pos*	*pos*	*neg*

negative or vice versa. However, the study infers that GRU and CRNN models predict identically when sentiment is determined by the whole sentence having a long-range semantic dependency.

7 Deep networks for aspect-based sentiment analysis

As an important subtask of sentiment analysis, ABSA has received much attention from both industry and academic communities (Fan et al. 2018a, b; Li et al. 2018). Sentiment analysis can be studied at different levels of detail. The document-level sentiment analysis arrives with a hypothesis that the whole document contains opinions about one topic. This kind of analysis is not adequate in many cases. The sentence-level sentiment analysis similarly assumes that only one topic can be expressed in one sentence. However, it is often the case that one sentence contains multiple topics (i.e., aspects) or that the opinions are opposite within the same sentence. In both document-level and sentence-level sentiment analysis, the opinion polarities are based on the whole document or sentence rather than the topics present in them. In contrast, ABSA aims to find the sentiment polarity expressed for each aspect being discussed. For instance, an opinion statement, “I just loved the smartphone but disappointed that EMI is not available.” Here, aspect-level sentiment classification should determine the sentiment expressed on the quality of the smartphone as positive and sentiment on the EMI facility as negative.

The following study provides a comprehensive overview of modern deep learning techniques used in ABSA. For each type of ABSA model, a brief description is provided and makes a necessary comparison and summary. The ABSA is a fundamental task in sentiment analysis research (Wang et al. 2016; Ma et al. 2018), which includes several key subtasks such as aspect extraction (Dong et al. 2014; Lakkaraju et al. 2014; Fan et al. 2018a), opinion identification (Chen et al. 2017; Huang et al. 2018), and aspect sentiment classification (Pontiki et al. 2015). The contemporary methods are grouped into five main categories: CNN based ABSA, RNN based ABSA, RcNN based ABSA, attention-based RNN for ABSA, and MemNets based ABSA. Table 7 provides a comparative study of various DNN models used in ABSA. However, the list is not exhaustive. It is only a representative sample of several DNN models explained above.

8 Conclusion and future directions

The fast-paced research in sentiment analysis using DNN techniques urges the researchers to investigate the modeltextquotesingle s performances. Hence, in this research work, the performance evaluation of several DNN architectures for sentiment classification is carried out. Four different and publicly available datasets have been employed to measure the accuracy of both binary and fine-grained sentiment classification models. A low-dimensional pre-trained word representation called *GloVe* is utilized with different DNN, including CNN, RNN, and many advanced techniques. Furthermore, the impact of tuning the hyperparameters such as the learning rate, hidden units, and batch size on various networks is presented. The study further ensures all deep learning practitioners with insights into ways to adapt stable DNN techniques for sentiment classification.

Nevertheless, the research on DNN continues to evolve with ideal solutions to address the current challenges or to bring more applications in the field of sentiment analysis. In

Table 7 Comparative study of DNN models for aspect-based sentiment analysis

DNN	Model	Attention	Task	Dataset	Performance
CNN	Multi-layer CNN (Gu et al. 2017)	No	OTE	Amazon product reviews	F1: 0.727–0.837
	Aspect WE + CNN (Du et al. 2016)	No	SP	Online product reviews	Acc: 0.92–0.944
	Multi-task CNN + WE (Wu et al. 2016)	No	SP	Amazon product reviews	Acc: 0.841
RNN	BiLSTM (Ma et al. 2018)	Target attention	OTI	SentiHood	Acc: 0.893
	LSTM + Local context + Senna WE; BiLSTM + Local Context + Senna WE (Yuan et al. 2017)	No	OTE	SemEval 14	F1: 0.800; F1: 0.748
	GRU (Wang et al. 2017)	Coupled multi-layer attentions	OTE	SemEval 14	F1: 0.778–0.853
	Hierarchical LSTM (Ding et al. 2017)	No	OTE	SemEval 14 & 15	F1: 0.438–0.779
	BiGRU (Cheng et al. 2017)	Aspect attention + sentiment attention	SP	SemEval 14 15 & 16	F1: 0.834–0.913
RcNN	RcNN+WE (Wang et al. 2016a)	No	OTE	SemEval 14	F1: 0.784
	Adaptive RcNN (Dong et al. 2014)	No	SP	Twitter data	Acc: 0.663
	ReTNN (Lakkaraju et al. 2014)	No	OTE	Product reviews	Acc: 0.770–0.810
Memory networks	MemRAM (Chen et al. 2017)	Recurrent attention	SP	Twitter data	Acc: 0.693–0.738
	Tensor DyMemNN (Tay et al. 2017)	Target attention	OTI	SemEval 2016 Tweet task	F1: 0.724
	Feature-based compositing MemNet (Ma et al. 2017)	No	SP	SemEval 14	F1: 0.739–0.820
	Deep MemNet (Tang et al. 2016)	No	SP	Online reviews	F1:0.723–0.809
	Holo DyMemNN (Tay et al. 2017)	Target attention	OTI	Online reviews	F1: 0.731–0.818
Hybrid networks	CNN + LSTM (Li et al. 2018)	Concat attention	SP	SemEval 14	Acc:0.806 F1: 0.712–0.765
	CNN + MemNet (Fan et al. 2018b)	General attention	SP	SemEval 14	Acc:0.782 F1: 0.683–0.763
	MGAN (Fan et al. 2018b)	Dot product attention	SP	SemEval 14	Acc:0.812 F1: 0.719–0.812
	AOA + BiLSTM (Huang et al. 2018)	General attention	OTI & SP	SemEval 14	Acc: 0.812 F1: 0.745
	LSTM+MemNet (Chen et al. 2017)	Concat attention	OTI & SP	SemEval 14	Acc: 0.802 F1: 0.693–0.744
	GRU + MemNet (Majumder et al. 2018)	Concat attention	SP	SemEval 14 & 15	Acc: 0.80 F1: 0.738

Hint: *OTE-Opinion target extraction, OTI-Opinion target identification, SP-Sentiment polarity* the following paragraph, we discuss several issues related to research in this domain as future directions. Firstly, hyperparameters tuning is task-dependent, and often it is hand operated, which cannot be addressed through explicit formulation. It is essential to devise algorithms to automatically optimize the hyperparameters based on the previous results or by using some random search techniques. On the other hand, stability analysis of DNN has gained popularity in the deep learning research community. There has been a significant amount of research reported on the stability issues and synchronization problems for different networks. Ensemble learning is one of the potential research areas in DNN, where several and discrete architectures can be combined to improve generalization on different categories of subjective content. Finally, the applications of DNN can be extended to other sentiment analysis tasks such as relation classification, machine translation, question answering system, topic identification, and also for more fine-grained sentiment analysis tasks, such as polarity detection and characterizing the intensity of opinion expressions.

References

Akhtar MS, Gupta D, Ekbal A, Bhattacharyya P (2017) Feature selection and ensemble construction: a two-step method for aspect based sentiment analysis. Knowl Based Syst 125:116–135

Akhtar MS, Ekbal A, Narayan S, Singh V (2018) No, that never happened!! investigating rumors on twitter. IEEE Intell Syst 33(5):8–15

Akhtar MS, Ghosal D, Ekbal A, Bhattacharyya P, Kurohashi S (2019) All-in-one: emotion, sentiment and intensity prediction using a multi-task ensemble framework. IEEE Trans Affect Comput. https://doi.org/10.1109/TAFFC.2019.2926724

Akhtar MS, Ekbal A, Cambria E (2020) How intense are you? Predicting intensities of emotions and sentiments using stacked ensemble. IEEE Comput Intell Mag 15(1):64–75

Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. arXiv preprint arXiv:1607.06450

Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. Technical report. arXiv preprint arXiv:1409.0473

Baziotis C, Pelekis N, Doulkeridis C (2017) Datastories at semeval-2017 task 4: deep LSTM with attention for message-level and topic-based sentiment analysis. In: Proceedings Of the 11th international workshop on semantic evaluation (SemEval-2017). Vancouver, Canada, August. Association For Computational Linguistics, pp 747–754

Bradbury J, Merity S, Xiong C, Socher R (2016) Quasi-recurrent neural networks. In Arxiv:1611.01576

Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140

Cambria E (2016) Affective computing and sentiment analysis. IEEE Intell Syst 31(2):102–107

Cambria K, Poria S, Gelbukh A, Thelwall M (2017) Sentiment analysis is a big suitcase. IEEE Intell Syst 32(6):74–80

Chaturvedi I, Ong YS, Tsang IW, Welsch RE, Cambria E (2016) Learning word dependencies in text by means of a deep recurrent belief network. Knowl Based Syst 108:144–154

Chaturvedi I, Cambria E, Welsch RE, Herrera F (2018) Distinguishing between facts and opinions for sentiment analysis: survey and challenges. Inf Fusion 44:65–77

Chen LC, Lee CM, Chen MY (2019a) Exploration of social media for sentiment analysis using deep learning. Springer, GmbH Germany, Part of Springer nature, soft computing

Cheng J, Zhao S, Zhang J, King I, Zhang X, Wang H (2017) Aspect-level sentiment classification with HEAT (HiErarchical ATtention) Network. CIKM 2017:97–106

Cheng J, Dong L, Lapata M (2016) Long short-term memory-networks for machine reading. ArXiv:1601.06733

Chen Z, Qian T (2019b) Transfer capsule network for aspect level sentiment classification. In: Proceedings of the 57th annual meeting of the association for computational linguistics, Florence, Italy, July 28–August 2, 2019. Association for Computational Linguistics, pp 547–556

Chen P, Sun Z, Bing L, Yang W (2017) Recurrent attention network on memory for aspect sentiment analysis. In: Proceedings of conference empirical methods natural language processing, pp 452–461

Cho K, Merrienboer BV, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: encoder-decoder approaches. Arxiv:1409.1259

Collobert RJ, Weston L, Bottou M, Karlen K, Kavukcuglu PK (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537

Courbariaux M (2016) Binarized neural networks: training neural networks with weights and activations constrained to +1 or 1. Arxiv:1602.02830v3

Ding Y, Yu J, Jiang J (2017) Recurrent neural networks with auxiliary labels for cross-domain opinion target extraction. AAAI 2017:3436–3242

Dong L, Wei F, Tan C, Tang D, Zhou M, Xu K (2014) Adaptive recursive neural network for target-dependent twitter sentiment classification. ACL 2014:49–54

Dragoni M, Petrucci G (2017) A neural word embeddings approach for multi-domain sentiment analysis. IEEE TransAffect Comput 8(4):457–470

Dragoni M, Poria S, Cambria E (2018) OntoSenticNet: a commonsense ontology for sentiment analysis. IEEE Intell Syst 33(2):77–85

Du Y, Zhao X, He M, Guo W (2019a) A novel capsule based hybrid neural network for sentiment classification. IEEE Access 9:39321–39328

Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12:21212159

Du C, Sun H, Wang J, Qi Q, Liao J, Xu T, Liu M (2019b) Capsule network with interactive attention for aspect-Level sentiment classification. In: Proceedings of international conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing, Hong Kong, China, November 3–7, 2019. Association for Computational Linguistics, pp 5489–5498

Du H, Xu X, Cheng X, Wu D, Liu Y, Yu Z (2016) Aspect-specific sentimental word embedding for sentiment analysis of online reviews. In: International conference companion on world wide web, pp 29–30

Ekbal A, Saha S (2011) Weighted vote-based classifier ensemble for named entity recognition: a genetic algorithm-based approach. ACM Trans Asian Lang Inf Process 10(2):1–9

Elman J (1990) Finding structure in time. Cognit Sci 14:179–211

Fan F, Feng Y, Zhao D (2018b) Multi-grained attention network for aspect-level sentiment classification. In: Proceedings of conference on empirical methods in natural language processing, pp 3433–3442

Fan C, Gao Q, Du J, Gui L, Xu R, Wong KF (2018a) Convolution-based memory network for aspect-based sentiment analysis. In: Proceedings of 41st International ACM SIGIR conference on Research and development in information retrieval, pp 1161–1164

Fentaw HW, Kim TH (2019) Design and investigation of capsule networks for sentence classification. Appl Sci 9:2200. https://doi.org/10.3390/app9112200

Fink CR, Chou DS, Kopecky JJ, Llorens AJ (2011) Coarse- and fine-grained sentiment analysis of social media text. Johns Hopkins Apl Tech Digest 30(1):22–30

Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Proceedings of 13th international conference on machine learning (ICML), Bari, Italy, pp 148–156

Fu X, Liu W, Xu Y, Cui L (2017) Combine HowNet lexicon to train phrase recursive autoencoder for sentence-level sentiment analysis. Neurocomputing 241:18–27

Gao L, Guo Z, Zhang H, Xu X, Shen HT (2017) Video captioning with attention-based LSTM and semantic consistency. IEEE Trans Multimed 19(9):2045–2055

Graves A, Wayne G, Danihelka I (2014) Neural turing machines. ArXiv:1410.5401

Gu X, Gu Y, Wu H (2017) Cascaded convolutional neural networks for aspect-based opinion summary. Neural Process Lett 46(2):581–594

Guo B, Zhang C, Liu J, Ma X (2019) Improving text classification with weighted word embeddings via a multi-channel TextCNN model. Neurocomputing 363:366–374

Guo L, Ye H, Su W, Liu H, Sun K, Xiang H, (2018) Visualizing and understanding deep neural networks in CTR prediction. Sigir, 2018 Ecom, July 2018. Michigan, USA, Ann Arbor

Hassan A, Mahmood A (2018) Convolutional recurrent deep learning model for sentence classification. IEEE Access 6:13949–13957

Hassan J, Shoaib U (2019) Multi-class review rating classification using deep recurrent neural network. Springer, LLC, Part of Springer Nature, Neural Process Letters

He R, Lee WS, Ng HT, Dahlmeier D (2018) Exploiting document knowledge for aspect-level sentiment classification. In: ACL 2018

Hinton GE, Osindero S, Teh Y (2006) A fast learning algorithm for deep belief nets. Neural Comput 18:1527–1554

Huang EH, Socher R, Manning CD, Ng AY (2012) Improving word representations via global context and multiple word prototypes. In: ACL 2012

Huang B, Ou Y, Carley KM (2018) Aspect level sentiment classification with attention-over-attention neural networks. Proc SBP-BRiMS 2018:197–206

Irsoy O, Cardie C (2014a) Opinion mining with deep recurrent neural networks. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), Doha, Qatar, October. Association For Computational Linguistics, pp 720–728

Irsoy O, Cardie C (2014b) Deep recursive neural networks for compositionality in language. In: Advances in neural information processing systems, pp. 2096–2104

Jabreel M, Moreno A (2017) Target-dependent sentiment analysis of tweets using a bi-directional gated recurrent unit. In: Proceedings of the 13th international conference on web information systems and technologies (WEBIST 2017), pp 80–87. ISBN: 978-989-758-246-2

Janocha K, Czarnecki WM (2017) On loss functions for deep neural networks in classification. Arxiv Preprint Arxiv:1702.05659

Jiang M, Liang Y, Feng X, Fan X, Pei Z, Xue Y, Guan R (2018) Text classification based on deep belief network and softmax regression. Neural Comput Appl 29(1):61–70. https://doi.org/10.1007/s00521-016-2401-x

Khan FH, Qamar U, Bashir S (2017) A semi-supervised approach to sentiment analysis using revised sentiment strength based on sentiwordnet. Knowl Inf Syst 51(3):851–872

Khanna, R, Awad, M (2015) Efficient learning machines: theories, concepts, and applications for engineers and system designers, Apress

Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods. EMNLP, pp 1746–1751

Kingma DP, Lei BJ (2014) Adam: a method for stochastic optimization. Arxiv Preprint Arxiv:1412.6980

Krishnakumari K, Sivasankar E, Radhakrishnan S (2019) Hyperparameter tuning in convolutional neural networks for domain adaptation in sentiment classification. Soft Comput 24:3511–3527

Kumar A, Irsoy O, Su J, Bradbury J, English R, Pierce B, Ondruska P, Gulrajani I, Socher R (2016) Ask me anything: dynamic memory networks for natural language processing. In: ICML 2016

Lai S, Xu L, Liu K, Zhao J (2015) Recurrent convolutional neural networks for text classification. In: Twenty-ninth AAAI conference on artificial intelligence

Lakkaraju H, Socher R, Manning CD (2014) Aspect specific sentiment analysis using hierarchical deep learning. In: NIPS WS on deep neural networks and representation learning, pp 1–9

Lauren P, Qu G, Zhang F, Lendasse A (2018) Discriminant document embeddings with an extreme learning machine for classifying clinical narratives. Neurocomputing 277:129–138

Le QV, Ngiam J, Coates A, Lahiri A, Prochnow B, Ng AY (2011) On optimization methods for deep learning. In: ICML

Lebret R, Collobert R (2014) Word embeddings through Hellinger PCA. In: EACL 2014

LeCen Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436444

Li X, Bing L, Lam W, Shi B (2018) Transformation networks for target-oriented sentiment classification. ACL 2018:946–956

Liang X, Wang X, Lei Z, Liao S, Li SZ (2017) Soft-margin softmax for deep classification. In: Liu D, Xie S, Li Y, Zhao D, El-Alfy ES (eds) Neural information processing. ICONIP, 2017 Lecture notes in computer science, 10635. Springer, Cham

Lin Z, Feng M, dos Santos CN, Yu M, Xiang B, Zhou B, Bengio Y (2017) A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130

Litjens G (2017) A survey on deep learning in medical image analysis. Med Image Anal 42:6088

Liu J, Zhang Y (2017a) Attention modeling for targeted sentiment. EACl 2017:572

Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017b) A survey of deep neural network architectures and their applications. Neurocomputing 234:1126

Liu N, Shen B, Zhang Z, Zhang Z, Mi K (2019) Attention-based Sentiment Reasoner for aspect-based sentiment analysis. Hum Cent Comput Inf Sci 9:35

Liu W, Cao G, Yin J (2019b) Bi-level attention model for sentiment analysis of short texts. IEEE Access 7:13–22

Liu S, Yang N, Li M, Zhou M (2014) A recursive recurrent neural network for statistical machine translation. In: ACL

Li H, Xu Z, Taylor G, Goldstein T (2017) Visualizing the loss landscape of neural nets. Arxiv Preprint Arxiv:1712.09913

Lo SL, Cambria E, Chiong R, Cornforth D (2017) Multilingual sentiment analysis: from formal to informal and scarce resource languages. Artif Intell Rev 48:499–527

Luong MT, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. arXiv:1508.04025

Ma S, Ji C (1998) A unified approach on fast training of feedforward and recurrent networks using EM algorithm. IEEE Trans Signal Process 46(8):2270–2274

Ma R, Wang K, Qiu T, Sangaiah AK, Lin D, Bin LH (2017) Feature-based compositing memory networks for aspect-based sentiment classification in social internet of things. Future Gener Comput Syst. https://doi.org/10.1016/j.future.2017.11.036

Ma Y, Peng H, Cambria E (2018) Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. Proc AAAI 2018:5876–5883

Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies , vol 1, pp 142–150

Majumder N, Poria S, Gelbukh A, Akhtar MS, Cambria E, Ekbal A (2018) IARM: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis. Proc Conf Empir Methods Nat Lang Process 2018:3402–3411

Majumder N, Poria S, Peng H (2019) Sentiment and sarcasm classification with multitask learning. IEEE Intell Syst 34(3):38–43

Medhat W, Hassan A, Korashy H (2014) Sentiment analysis algorithms and applications: a survey. Ain Shams Eng J 5(4):1093–1113

Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013b) Distributed representations of words and phrases and their compositionality. NIPS 2013:3111–3119

Mikolov T, Chen K, Corrado G, Dean J (2013a) Efficient estimation of word representations in vector space. In: ICLR 2013

Mnih A, Kavukcuoglu K (2013) Learning word embeddings efficiently with noise-contrastive estimation. In: NIPS 2013

Mousa A, Schuller S (2017) Contextual bidirectional long short-term memory recurrent neural network language models: a generative approach to sentiment analysis. In: Proceedings of the 15th conference of the European chapter of the association for computational linguistics, vol 1, pp 1023–1032

Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: ICML 2010

Nguyen HH, Yamagishi J, Echizen I (2019) Capsule-forensics: using capsule networks to detect forged images and videos. In: 2019 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 2307–2311

Nguyen N, Le A, Pham HT (2012) Deep bi-directional long short-term memory neural networks for sentiment analysis of social data. IUKM 2016:255–268

Paik I, Kwak T, Kim I (2019) Capsule networks need an improved routing algorithm. ArXiv, abs/1907.13327

Pandarachalil R, Sendhilkumar S, Mahalakshmi GS (2015) Twitter sentiment analysis for large-scale data: an unsupervised approach. Cognit Comput 7:254–262

Pascanu R, Gulcehre C, Cho K, Bengio Y (2014) How to construct deep recurrent neural networks. In: Proceedings of the second international conference on learning representations (ICLR 2014)

Pasupa K, Ayutthaya TSN (2019) Thai sentiment analysis with deep learning techniques: a comparative study based on word embedding, POS-tag, and sentic features. Sustain Cities Soc 50:101615

Patrick MK, Adekoya AF, Mighty AA, Edward BY (2019) Capsule networks-a survey. J King Saud Univ Comput Inf Sci (2019)

Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing, pp 1532–1543

Pergola G, Gui L, He Y (2019) TDAM: a topic-dependent attention model for sentiment analysis. Inf Process Manag 56:102084

Plank B, Søgaard A, Goldberg Y (2016) Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In: Annual conference of the association for computational linguistics (ACL), pp 412–418

Pontiki M, Galanis D, Papageorgiou H, Manandhar S, Androutsopoulos I (2015) Semeval-2015 task 12: aspect based sentiment analysis. In: Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015), pp 486–495

Pontiki M, Galanis D, Pavlopoulos J, Papageorgiou H, Androutsopoulos I, Manandhar S (2014) Semeval-2014 task 4: aspect based sentiment analysis. In: Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014), pp 27–35

Qin P, Xu W, Guo J (2016) An empirical convolutional neural network approach for semantic relation classification. Neurocomputing 190:1–9

Rehman AU, Malik AK, Raza B, Ali W (2019) Hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis. Multimed Tools Appl. https://doi.org/10.1007/s11042-019-07788-7

Ren Y, Wang R, Ji D (2016) A topic-enhanced word embedding for Twitter sentiment classification. Inf Sci 369:188–198

Rezaeinia SM, Rahmani R, Ghodsi A, Veisi H (2018) Sentiment Analysis based on improved pre-trained word embeddings. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2018.08.044

Rida-e-Fatima S, Javed A, Banjar A, Dawood AIH, Dawood H, Alamri A (2019) A multi-layer dual attention deep learning model with refined word embeddings for aspect-based sentiment analysis. IEEE Access 7:114795–114807

Rizk Y, Hajj N, Mitri N, Awad M (2019) Deep belief networks and cortical algorithms: a comparative study for supervised classification. Appl Comput Inf 15:81–93

Ruder S (2016) An overview of gradient descent optimization algorithms. Corr Abs/1609.04747

Sabour S, Frosst N, Hinton GE (2017) Dynamic routing between capsules. In: NIPS 2017

Sadr H, Pedram MM, Teshnehlab M (2019) A robust sentiment analysis method based on sequential combination of convolutional and recursive neural networks. Neural Process Lett. https://doi.org/10.1007/s11063-019-10049-1

Salehinejad H, Sankar S, Barfett J, Colak E, Valaee E (2017) Recent advances in recurrent neural networks. Arxiv Preprint Arxiv:1801.01078

Salinca A (2017) Convolutional neural networks for sentiment classification on business reviews. Arxiv Preprint Arxiv:1710.05978

Santos CD, Gatti G (2014) Deep convolutional neural networks for sentiment analysis of short texts. In: Proceedings of coling 2014, the 25th international conference on computational linguistics: technical papers, Dublin, Ireland, August 2014. Dublin City University And Association For Computational Linguistics, pp 69–78

Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85117

Schulz H, Behnke S (2012) Deep learning-layer-wise learning of feature hierarchies. Kunstliche Intelligenz 26(4):357–363

Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681

Schuurmans J, Frasincar F (2019) Intent classification for dialogue utterances. IEEE Intell Syst. https://doi.org/10.1109/MIS.2019.2954966

Seifert C, Aamir A, Balagopalan A, Jain D, Sharma A, Grottel S, Gumhold S (2017) Visualizations of deep neural networks in computer vision: a survey. In: Transparent data mining for big and small data, pp 123–144

Severyn A, Moschitti A (2015) Twitter sentiment analysis with deep convolutional neural networks. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, pp 959–962

Shaw P, Uszkoreit J, Vaswani A (2018) Self-attention with relative position representations. In: Proceedings of NAACL

Shen T, Zhou T, Long G, Jiang J, Zhang C (2018) Bi-directional block self-attention for fast and memory-efficient sequence modeling. In: ICLR 2018

Socher R, Huval B, Manning CD, Ng AY (2012) Semantic compositionality through recursive matrix-vector spaces. EMNLP-CoNLL 12:1201–1211

Socher R, Perelygin A, Wu J, Chuang J, Manning C, Ng A, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: EMNLP 2013

Srivastava RK, Greff K, Schmidhuber J (2015) Training very deep networks. Arxiv Preprint Arxiv:1507.06228

Sukhbaatar S, Szlam A, Weston J, Fergus R (2015) End-to-end memory networks. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) Advances in neural information processing systems, vol 28. Curran Associates Inc, New York, pp 2440–2448

Sutskever I (2012) Training recurrent neural networks. Ph.D. Thesis, University of Toronto.

Tai KS, Socher R, Manning CD (2015) Improved semantic representations from tree-structured long short-term memory networks. In: ACL 2015

Tang D, Qin B, Feng X, Liu T (2015) Effective LSTMs for target dependent sentiment classification. Arxiv Preprint Arxiv:1512.01100

Tang D, Qin B, Liu T (2016) Aspect level sentiment classification with deep memory network. ArXiv Preprint ArXiv:1605:08900

Tang D, Wei F, Yang N, Zhou M, Liu T, Qin b (2014) Learning sentiment-specific word embedding for Twitter sentiment classification. In: Proceedings of the 52nd annual meeting of the association for computational linguistics, vol 1, pp 1555–1565

Tay Y, Tuan LA, Hui SC (2017) Dyadic memory networks for aspect-based sentiment analysis. CIKM 2017:107–116

Thireou T, Reczko M (2007) Bidirectional long short-term memory networks for predicting the subcellular localization of eukaryotic proteins. IEEE/ACM Trans Comput Biol Bioinform 4(3):441–446

Tian Z, Rong W, Shi L, Liu J, Xiong Z (2018) Attention aware bidirectional gated recurrent unit based framework for sentiment analysis. In: Liu W, Giunchiglia F, Yang B (eds) Knowledge science, engineering and management. Ksem, 2018. Lecture notes in computer science, 11061. Springer, Cham

Valdivia A, Luzón MV, Herrera F (2017) Sentiment analysis in tripAdvisor. IEEE Intell Syst 32(4):72–77

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017). Attention is all you need. In: ANIPS, pp 6000–6010

Wadawadagi RS, Pagi VB (2019a) An enterprise perspective of web content analysis research: a strategic roadmap. Int J Knowl Web Intell 6(2):51–88

Wadawadagi RS, Pagi VB (2019b) A deep recursive neural network model for fine-grained opinion classification. In: Santosh K, Hegadi R (eds) Recent trends in image processing and pattern recognition. Rtip2r, 2018. Communications in computer and information science, 1037. Springer, Singapore

Wadawadagi RS, Pagi VB (2020a) Handbook of research on emerging trends and applications of machine learning. In: Arun S, Sandeep K, Anand N (eds) Chapter 24:Handbook of research on emerging trends and applications of machine learning. Part of the advances in computational intelligence and robotics book series. IGI-Gobal Publishers, Hershey, pp 508–527

Wadawadagi RS, Pagi VB (2020b) Fine-grained sentiment rating of online reviews with Deep-RNN. In: Niranjan NC, Takanori F (eds) Advances in artificial intelligence and data engineering. AIDE, 2019. Advances in intelligent systems and computing, 1133. Springer, Singapore

Wang X, Jiang W, Luo Z (2016) Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In: Proceedings of the international conference on computational linguistics (COLING 2016)

Wang W, Pan SJ, Dahlmeier D (2017) Coupled multi-layer attentions for co-extraction of aspect and opinion terms. AAAI, pp 3316–3322. arxiv.org/abs/1702.01776

Wang W, Pan SJ, Dahlmeier D, Xiao X (2016a) Recursive neural conditional random fields for aspect-based sentiment analysis. In: EMNLP 2016

Wang Y, Sun A, Han J, Liu Y, Zhu X (2018) Sentiment analysis by capsules. In: Proceedings of the 2018 world wide web conference. International world wide web conferences steering committee, pp 1165–1174

Weichselbraun A, Gindl S, Fischer F, Vakulenko S, Scharl A (2017) Aspect-based extraction and analysis of affective knowledge from social media streams. IEEE Intell Syst 32(3):80–88

Weston J, Chopra S, Bordes A (2014) Memory networks. CoRR, abs/1410.3916

Wu H, Gu Y, Sun S, Gu X (2016) Aspect-based opinion summarization with convolutional neural networks. In: Proceedings of the international joint conference on neural networks, pp 3157–3163

Xiao T, Zhu J, Liu T (2013) Bagging and boosting statistical machine translation systems. Artif Intell 195:496–527

Xu C, Feng H, Yu G, Yang M, Wang X, Ao X (2020) Discovering protagonist of sentiment with aspect reconstructed capsule network. arXiv:1912.10785

Yin W, Kann K, Yu M, Schutze H (2017) Comparative study of CNN and RNN for natural language processing. Arxiv:1702.01923

Yosinski J, Clune J, Nguyen A, Fuchs T, Lipson H (2015) Understanding neural networks through deep visualization. In: ICML deep learning workshop (2015)

Yousefi-Azar M, Hamey L (2017) Text summarization using unsupervised deep learning. Expert Syst Appl 68:93–105

Yu LC, Wang J, Lai KR, Zhang X (2017) Refining word embeddings for sentiment analysis. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp 534–539 Copenhagen, Denmark, September 7–11, 2017

Yuan J, Zhao Y, Qin B, Liu T (2017) Local contexts are effective for neural aspect extraction. Commun Comput Inf Sci 774:244–255

Zeiler MD (2012) Adadelta: an adaptive learning rate method. Arxiv:1212.5701

Zhang L, Wang S, Liu B (2017) Deep learning for sentiment analysis: a survey. Wires Data Min Knowl Discov 2018:E1253

Zhang S, Zheng D, Hu X, Yang M (2015) Bidirectional long short-term memory networks for relation classification. In: Proceedings of the 29th Pacific Asia conference on language, information and computation, pp 73–78

Zhao W, Peng H, Eger S, Cambria E, Yang M (2019) Towards scalable and reliable capsule networks for challenging NLP applications. In:Proceedings of the 57th annual meeting of the association for computational linguistics, pp 1549–1559

Zhou S, Chen Q, Wang X (2014) Fuzzy deep belief networks for semi-supervised sentiment classification. Neurocomputing 131:312–322

Zhou X, Wan X, Xiao J (2016) Attention-based LSTM network for cross-lingual sentiment classification. In: EMNLP 2016