Future Generation Computer Systems 115 (2021) 279–294

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis

Mohammad Ehsan Basiri [a], Shahla Nemati [a], Moloud Abdar [b], Erik Cambria [c,*], U. Rajendra Acharya [d]

[a] Department of Computer Engineering, Shahrekord University, Iran
[b] Institute for Intelligent Systems Research and Innovation, Deakin University, Australia
[c] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[d] Department of Electronics and Computer Engineering, Ngee Ann Polytechnic, Singapore

ARTICLE INFO

Article history:
Received 12 March 2020
Received in revised form 25 June 2020
Accepted 3 August 2020
Available online 3 September 2020

Keywords:
Sentiment analysis
Deep learning
Convolutional neural network
Long short-term memory
Attention mechanism

ABSTRACT

Sentiment analysis has been a hot research topic in natural language processing and data mining fields in the last decade. Recently, deep neural network (DNN) models are being applied to sentiment analysis tasks to obtain promising results. Among various neural architectures applied for sentiment analysis, long short-term memory (LSTM) models and its variants such as gated recurrent unit (GRU) have attracted increasing attention. Although these models are capable of processing sequences of arbitrary length, using them in the feature extraction layer of a DNN makes the feature space high dimensional. Another drawback of such models is that they consider different features equally important. To address these problems, we propose an Attention-based Bidirectional CNN-RNN Deep Model (ABCDM). By utilizing two independent bidirectional LSTM and GRU layers, ABCDM will extract both past and future contexts by considering temporal information flow in both directions. Also, the attention mechanism is applied on the outputs of bidirectional layers of ABCDM to put more or less emphasis on different words. To reduce the dimensionality of features and extract position-invariant local features, ABCDM utilizes convolution and pooling mechanisms. The effectiveness of ABCDM is evaluated on sentiment polarity detection which is the most common and essential task of sentiment analysis. Experiments were conducted on five review and three Twitter datasets. The results of comparing ABCDM with six recently proposed DNNs for sentiment analysis show that ABCDM achieves state-of-the-art results on both long review and short tweet polarity classification.

1. Introduction

Sentiment analysis aims at analyzing and extracting knowledge from the subjective information published on the Internet. Due to its vast range of academic and industrial applications as well as exponential growth of Web 2.0, sentiment analysis has been a hot research field in data mining and natural language processing (NLP) recently [1]. Therefore, various methods and tools capable of specifying the polarity of a document have been developed in recent years. Polarity detection is a binary classification task that represents an important dowel in most sentiment analysis applications [2]. Most of earlier methods for sentiment analysis, trained shallow models on carefully designed effective features to obtain satisfactory polarity classification results [3].

These models usually applied traditional classification methods including support vector machines (SVM), latent Dirichlet allocation (LDA), and Naïve Bayes on linguistic features such as n-grams, part-of-speech (POS) tags, and lexical features. There are two main drawbacks to this approach; (i) the feature space on which the model should be trained is sparse and high-dimensional which decreases the performance of the model, (ii) the feature engineering process is a labor- and time-intensive task.

To address the above mentioned drawbacks of traditional classification methods, learning word embedding has been proposed and used by several recent research works [4–6]. Word embedding is a real-valued dense vector created using a neural language model that considers different lexical relationships [7,8]. This makes the use of word embedding as the input to deep neural networks (DNN) very popular in recent NLP studies [7]. DNNs have attracted the attention of many researchers in different fields such as computer vision [9], multimodal sentiment analysis [10], medical informatics [11], and finance [12] in recent years.

DNNs have been proposed for the analysis of textual data mainly focuses on either learning word embedding or performing

* Corresponding author.
E-mail address: cambria@ntu.edu.sg (E. Cambria).

https://doi.org/10.1016/j.future.2020.08.005
0167-739X/© 2020 Elsevier B.V. All rights reserved.

machine learning tasks such as classification and clustering on the learned feature vectors [13]. Among the vast deep network types, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are more common in text processing related studies [13]. The reason for this popularity is the ability of CNNs in learning local patterns and the power of RNNs in sequential modeling. Although RNNs are suitable for many text processing applications, they suffer from vanishing and exploding gradients when there are long-term dependencies in the input data [7]. Such dependencies are common in most NLP applications and specifically in sentiment analysis.

To address this problem, long short-term memory (LSTM) and gated recurrent unit (GRU) networks were introduced. The former address the problem via input, forget, and output gates, while the latter exploits a reset gate and an update gate (see Section 2). Due to their potential to solve the problems of standard RNNs, both LSTM and GRU have attracted the attention of NLP researchers [14]. To consider both the preceding and succeeding contexts, bidirectional LSTM (Bi-LSTM) and bidirectional GRU (Bi-GRU) were proposed. By combining the forward and backward hidden layers, these models can better address the sequential modeling problem.

Although Bi-LSTM and Bi-GRU are extensively used in NLP applications, there are two main drawbacks in them: (1) the high-dimensional input space common in text processing applications increases the complexity of the model and makes it difficult to optimize; (2) the model cannot focus on the important parts of the contextual information of text. To address these problems, several approaches were proposed in the literature. For example, CNNs were used to decrease the dimensionality of the feature space as well as extracting meaningful features from text [15]. Attention mechanism was used to focus on the important parts of context by assigning different weights [7].

However, existing deep models for SA usually address a few problems and neglect others. For example, Chatterjee et al. [16] utilized two pre-trained word embeddings and LSTM to extract both sentiment and semantic for emotion recognition, but their model did not consider the differences between the importance of different parts of sentences. Liu et al. [17] combined bidirectional LSTM with CNN and exploited attention mechanism but did not addressed the problem of co-occurrence of both short and long dependencies. Rezaeinia et al. [18] improved pre-trained word embeddings and employed CNNs but did not consider long dependencies and words with different importances. To fill the existing gap, the current study proposes a new deep model for polarity detection of both short and long user comments. Specifically, our proposed model extracts both long and short intra-sentence relations, considers forward and backward contextual dependencies, selects most important features, and pay more or less attention to different words in comments.

In the proposed model, first, the global vectors for word representation (GloVe) [8] are used as the weights in the embedding layer. Then, the attention mechanism is used at the outputs of Bi-LSTM and Bi-GRU branches to make the model capable of paying more or less attention to different words and sentences. Convolution is then used to extract informative features and decrease the dimensionality of the input data. Also, global and average pooling layers are stacked at the outputs of CNN layers to down sample their feature maps. This makes the resulting feature maps more robust to the positional changes of features. To verify the effectiveness of the proposed model, we conducted the experiments on eight sentiment analysis datasets including five large-scale Amazon review datasets and three Twitter datasets containing more than one million user tweets.

In the experiments, the proposed model is compared with six state-of-the-art DNN-based text classification and sentiment analysis methods. On both long reviews and short tweets, the proposed model outperformed other six methods in terms of common performance measures in sentiment analysis and NLP domains. Our main contributions are the following:

- Building a new deep architecture for sentiment analysis.
- Evaluating our model on two types of social media texts: long reviews and short tweets.
- Comparing the performance of the proposed model with six recent deep architectures for text classification and sentiment analysis.

The remainder of this article is organized as follows: Section 2 proposes a short literature review of neural models for sentiment analysis and text classification; Section 3 presents the theoretical background of the proposed neural model; Section 4 describes the proposed model in detail; experiments and results are presented in Section 5; finally, Section 6 concludes the paper and offers some directions for future research.

2. Related work

2.1. Sentiment analysis

Most traditional sentiment analysis research works used supervised machine learning methods as their core classification or clustering module [19]. These methods usually exploited bag-of-words (BOW) model and n-gram features to present and classify user generated sentiment-bearing texts [20]. These features are proposed to address the problems of simple BOW model including ignoring word order and syntactic structures [21]. The main drawback of using n-gram features, specially when $n >= 3$, is the resulted high dimensional feature space. To address this problem, feature selection methods have been extensively exploited in recent studies [22,23].

Among different classification methods used for detecting users' sentiment from their text, SVM, LDA, Naïve Bayes, and artificial neural networks are more common and achieved higher performance [24–26]. The main problems of these supervised methods are that they need a large amount of training data and are usually slow. To address these problems, unsupervised lexicon-based methods were proposed [20,27]. These methods are simple, fast, and scalable. However, they heavily rely on the lexicon, making them less accurate than their supervised counterparts [27,28]. Domain dependency is another problem of lexicon-based methods which make them less applicable for domains that do not have specific lexicons.

To benefit from the advantages of both supervised and lexicon-based methods, few researchers combined them in different ways [29,30]. For example, Zhang et al. [31] proposed a two-step method for entity-level sentiment analysis of tweets in which the first step is a lexicon-based method with high precision and the second step is a supervised method with high recall. Mudians et al. [32] have also proposed a hybrid of lexicon-based and machine learning methods for concept-based sentiment analysis. Their method outperformed pure lexicon-based methods in both polarity and sentiment strength detection, and offered more accurate explanation and justification compared to purely statistical methods. Recently, Ghiassi and Lee [33] proposed a new hybrid method by identifying and reducing the size of a Twitter specific lexicon set and then, feeding it to a supervised method for sentiment classification. Finally, Chikersal et al. [34] proposed a hybrid of machine learning and lexicon-based methods for sentiment polarity. They showed that the hybrid method outperformed both the statistics- and lexicon-based method in classifying user reviews.

Table 1
Summary of DDN models selected for comparison with our model (ABCDM).

Research	RNN	CNN	Attention	Multi channel	Dataset type	Num of datasets
SS-BED [16]	✓				tweet	1
HAN [35]	✓		✓		review	4
ARC [36]	✓	✓	✓		tweet + review	2
CRNN [37]	✓	✓			review	3
IWV [18]		✓			review	5
AC-BiLSTM [17]	✓	✓	✓		review	7
ABCDM (This study)	✓	✓	✓	✓	tweet + review	8

2.2. Deep models for sentiment analysis

In the field of sentiment analysis, most of recent DNN-based studies has been oriented towards learning word embedding or exploiting different kinds of DNNs for classification or clustering tasks. Word embeddings are created to capture word similarities and their lexical relationships [38]. To create such embeddings, unsupervised methods are usually used. These methods are based on the fact that the words in similar contexts have similar meanings, hence, should have similar vectors. The main drawback of this assumption is that the vector of some semantically different words that usually co-occur in a small neighborhood are similar. For example, sentiment-bearing words with opposite meaning such as good and bad may have similar vectors because they usually appear in similar context. To address this problem, few researchers proposed sentiment-aware word vectors. These vectors are created based on large sentiment lexicons and supervised methods [7,39–41]. Dragoni and Petrucci [42] proposed a new neural word embedding method for multi-domain sentiment analysis. They addressed the main shortcomings of previous methods which did not yield good performance when employed in different domain from the one they were trained on. Their new method performed better by obtained higher performance.

Due to their ability to model long-term dependencies, LSTM and its variants are used extensively in sentiment analysis applications [43]. For example, Ju et al. proposed cached LSTM to learn local and global semantic features in a long text [44]. Lu et al. proposed p-LSTM in which three-words embedding is used instead of one-word embedding. Phrase embedding layer and LSTM are used in p-LSTM for sentiment classification tasks [45]. Recently, Chatterjee et al. [16], proposed a multichannel LSTM model named SS-BED for emotion detection in tweets. In their model, GloVe [8] and Sentiment Specific Word Embedding (SSWE) [46] are used in parallel as pre-trained word embedding, then for each path, three LSTM modules are applied sequentially to model long dependencies in text. Finally, two resulted feature vectors are concatenated to form the input to the fully connected layer. Other variants of LSTM used for sentiment analysis include TD-LSTM [47], SLSTM [48], cBLSTM [49], Tree-LSTM [50], and Sentic LSTM [51].

Recently, the attention mechanism is used to improve DNNs by letting them know where to focus for learning. For example, Zhou et al. [52] proposed a bidirectional LSTM with attention mechanism to select the important features. Yang et al. [35], proposed a new attention-based network named hierarchical attention networks (HAN) for text classification. In their model, they employed two attention modules in word and sentence levels, respectively. They stacked the attention modules on the outputs of GRU-based sequence encoders. Recently, He et al. [53] used two transfer methods besides the attention-based LSTM for document-level sentiment analysis. They also proposed a target-aware and a syntax-based attention mechanism for aspect-level sentiment analysis [54].

CNNs are used in sentiment analysis applications as local feature extractors. In other words, these models are useful when, in a long text, certain local patterns such as n-grams are of importance. For example, Johnson and Zhang [55], used the BOW model in convolution layer and proposed a new model named Seq-CNN to keep words' information. Kalchbrenner et al. [56] proposed a dynamic CNN method named DCNN, for sentence-level sentiment analysis. Dynamic K-Max pooling is used in the DCNN to capture word relations. Recently, Rezaeinia et al. proposed a CNN-based model that exploited improved word embedding for sentiment analysis in document level [18]. In their model, they improved pre-trained Word2Vec [57] and GloVe [8] embedding with lexical, positional, and syntactical features. Then, applied three different CNN modules sequentially to select important features from text. Other variants of CNN used for sentiment analysis applications include charCNN [58], CNN-rand, CNN-static, CNN-multichannel [59], CNN-LSTM [37], Ada-CNN [60], and many more.

Few researchers proposed hybrid DNNs for sentiment analysis [61]. For example, Wang et al. [37] proposed combination of CNN and RNN for sentiment analysis of short texts. They tried different combinations of CNN with LSTM and GRU modules on three different datasets of short texts. Recently, Wen and Li [36], proposed a combination of GRU and CNN with attention mechanism named ARC to classify tweets and reviews. They used bidirectional GRU units and three different CNN modules to extract local n-gram and global features. More recently, Liu and Guo [17] proposed a combination of bidirectional LSTM and CNN networks with attention mechanism named AC-BiLSTM for sentiment analysis and question answering. In AC-BiLSTM, CNN is first applied on the word embedding layer, then BiLSTM is used to extract long dependencies. Finally, attention mechanism is used to focus on important areas of the text.

In addition to the above-mentioned methods, several authors proposed deep models for general text classification which can also be used for sentiment analysis. Minaee et al. [62] reviewed 150 deep learning methods for text classification and they discussed more than 40 well-known datasets used in text classification task. In another research, Liu et al. [63] proposed a pre-training model called RoBERTa, a modified version of BERT, which uses both training data size and key hyper-parameters. Lan et al. [64] introduced two new parameter reduction approaches to increase the training speed and at the same time to reduce memory requirement of BERT. The new techniques named ALBERT and BERT-large obtained results similar to the new state-of-the-art methods (RACE, GLUE, and SQuAD). Attention mechanisms are very popular approaches as they have low training time and use parallel computation. Shen et al. [65] proposed a new model for learning sentence embedding called DiSAN (Directional Self-Attention Network). Their proposed method does not use any CNN/RNN structure and yielded good results for many datasets.

In the current study, we compared the proposed method with six similar DNN models, namely CRNN [37], IWV [18], SS-BED [16], HAN [35], ARC [36], and AC-BiLSTM [17]. The first two models are CNN-based, the SS-BED is a LSTM-based model, and last three ones are hybrid models with attention mechanism. Table 1 summarizes these DNN models. The main differences between these models and our proposed ABCDM model is that our model considers the following important features simultaneously: considering both long and short contextual dependencies

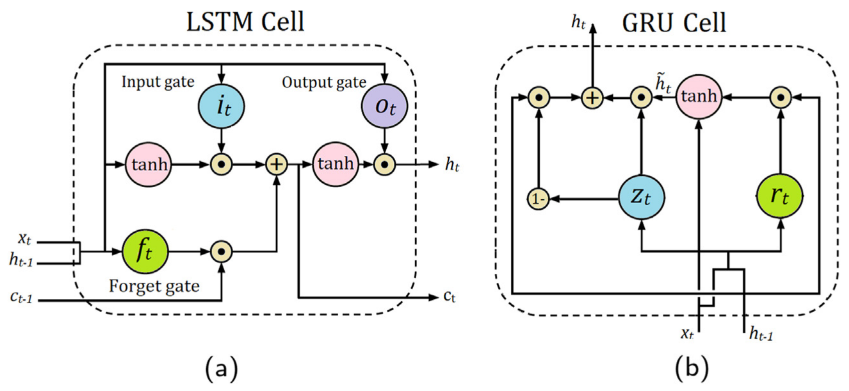


Fig. 1. Comparison of RNN units (a) LSTM and (b) GRU.

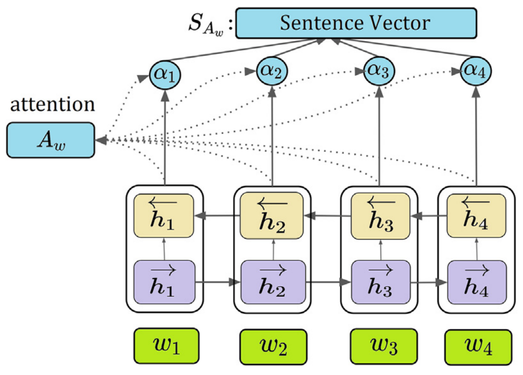


Fig. 2. The attention mechanism in a bidirectional network.
Source: Adopted from [66].

using bidirectional GRU and LSTM, selecting most important features robust to positional changes using CNNs with different filter size, kernels, and pooling mechanism, and paying different attentions to different parts of comments.

3. Preliminaries

In this section, a brief overview of basic building blocks of ABCDM is presented. Specifically, LSTM, GRU and Bidirectional LSTM networks are described in Section 3.1, then attention mechanism and CNN are presented in Sections 3.2 and 3.3, respectively.

3.1. Long short-term memory

LSTM is a special type of RNN which is designed to handle the vanishing/exploding problem faced by RNNs. LSTMs, like other types of RNNs, generate their output based on the input from the current time-step and the output of the previous time-step and send the current output to the next time-step. Every LSTM unit consists of a memory cell c_t, which preserves its state over arbitrary time intervals and three non-linear gates including an input gate i_t, a forget gate f_t, an output gate o_t. These gates are designed to regulate information flow into and out of the memory cell (see Fig. 1(a)) [17].

Suppose $\sigma(.)$, tanh(.), and $\odot$ are the element-wise sigmoid function, hyperbolic tangent function, and product, respectively. $\mathbf{x}_t$ and $\mathbf{h}_t$ are the input vector and the hidden state vector at time t. $\boldsymbol{U}$ and $\boldsymbol{W}$ show the weight matrices of gates or cell for input $\mathbf{x}_t$ and hidden state $\mathbf{h}_t$ and $\boldsymbol{b}$, denote the bias vectors. The forget gate decides what information needs to be forgotten by outputting a number in [0, 1] according to the following equation [17].

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

The input gate decides what new information should be stored by computing i_t and $\tilde{c}_t$ and combining them according to the following equations.

$$\mathbf{i}_t = \sigma(\boldsymbol{W}_i \mathbf{h}_{t-1} + \boldsymbol{U}_i \mathbf{x}_t + \boldsymbol{b}_i) \tag{2}$$

$$\tilde{\mathbf{c}}_t = \tanh(\boldsymbol{W}_c \mathbf{h}_{t-1} + \boldsymbol{U}_c \mathbf{x}_t + \boldsymbol{b}_c) \tag{3}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{4}$$

The output gate decides which parts of the cell state should be outputted according to the following equations.

$$\mathbf{o}_t = \sigma(\boldsymbol{W}_o \mathbf{h}_{t-1} + \boldsymbol{U}_o \mathbf{x}_t + \boldsymbol{b}_o) \tag{5}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{6}$$

To capture the future context in addition to the preceding context, BiLSTM combines forward $\overrightarrow{h}_t$ and backward $\overleftarrow{h}_t$ hidden layers. This results in the temporal information flow in both directions and better learning in the network.

A GRU is simpler variant of LSTM that has two gates, an update gate r that combines forget and input gates, and a reset gate z [13]. Similar to LSTM, the update and reset are computed as [67]:

$$\mathbf{r}_t = \delta(\boldsymbol{W}_r \mathbf{h}_{t-1} + \boldsymbol{U}_r \mathbf{x}_t + \boldsymbol{b}_r) \tag{7}$$

$$\mathbf{z}_t = \delta(\boldsymbol{W}_z \mathbf{h}_{t-1} + \boldsymbol{U}_z \mathbf{x}_t + \boldsymbol{b}_z) \tag{8}$$

where $\delta(.)$ is the logistic sigmoid function and W, U, and b are as before. The reset gate decides when the previous hidden state should be ignored and the update gate decides the amount of information that should be passed to the current state [67]. The hidden state is computed as:

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{9}$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_{\tilde{h}_t}(\mathbf{h}_{t-1} \odot \mathbf{r}_t) + \boldsymbol{U}_{\tilde{h}_t} \mathbf{x}_t) \tag{10}$$

3.2. Attention model

Attention models are used to assign different weights to words contributing differently to the sentiment of a text. A common way of assigning different weights to different words in a sentence is to use a weighted combination (see Fig. 2) of all hidden states, S_{A_W} as follows.

$$\boldsymbol{\alpha}_t = \frac{\exp(\mathbf{v}^{\intercal} \cdot \tilde{\mathbf{h}})}{\sum_t \exp(\mathbf{v} \cdot \tilde{\mathbf{h}})} \tag{11}$$

$$\mathbf{S}_{\mathbf{A_w}} = \sum_t \boldsymbol{\alpha}_t \mathbf{h}_t \tag{12}$$

where $\tilde{h}$ and h are defined as shown in Eqs. (9) and (10) and υ is a trainable parameter [66].

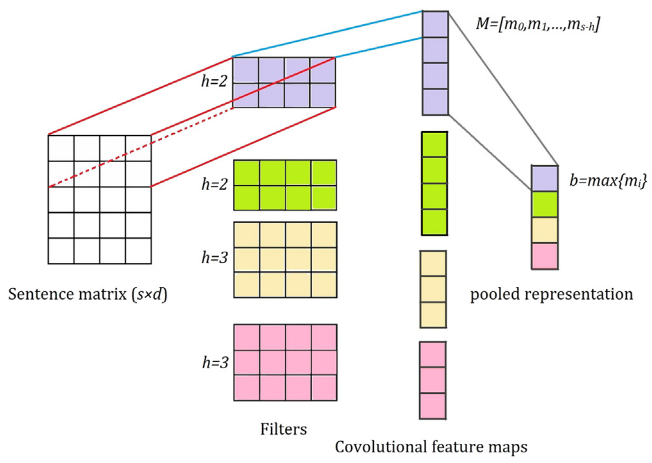


Fig. 3. CNN for feature extraction. The input is a sequence of d-dimension word embeddings ($e = 4$), two filters for $h = 2$ and two filters for $h = 3$ are applied the embeddings and four feature maps are generated. The max-pooling is used to generate a pooled vector of size 4.
Source: Adopted from [68].

3.3. Convolutional neural network

CNNs consist of several convolution layers and are used in the NLP applications for local feature extraction. In these networks, convolution operation is performed on the input features via linear filters. To apply a CNN on a sentence S with s words, first, an embedding vector of size e is created. Then, a filter F of size $e \times h$ is repeatedly applied to the sub-matrices of the input feature matrix. This, produces a feature map $M = [m_0, m_1, \dots, m_{s-h}]$ as follows [68]:

$$m_i = \mathbf{F} \cdot \mathbf{S}_{i:i+h-1} \tag{13}$$

where, $i = 0, 1, \dots, s-h$ and $S_{i:j}$ is a sub-matrix of S from row i to j. it is a common practice to reduce the dimension of feature maps by feeding them to a pooling or sub-sample layer. Max-pooling is a common pooling strategy which select the most important feature b of the feature map as follows:

$$b = \max_{0 \leqslant i \leqslant s-h} \{m_i\} \tag{14}$$

The outputs of pooling layer are concatenated and form a pooled feature vector which may then be used as the input of a fully connected network (see Fig. 3).

4. Proposed system

To address the limitations of the existing deep architectures for sentiment analysis, the current study proposes a new deep model, ABCDM, for polarity detection of both short and long user comments. In ABCDM, GloVe word embedding, bidirectional GRU, bidirectional LSTM, attention mechanism, and CNN are used to better capture both long-term dependencies and local features.

To generate the input comment matrix, a pre-trained GloVe embedding matrix $W_g \in \mathbb{R}^{n \times e}$ with n and e being the total number of words and embedding dimension, is used to embed a comment vector $c \in \mathbb{R}^m$ with m being the padding length or maximum number of words $w_t, t \in [1, m]$ considered in the comment as follows.

$$c_t = W_g w_t, t \in [1, m] \tag{15}$$

Then, two parallel layers of Bi-LSTM and Bi-GRU are applied on the output of embedding layer to process the sequences of arbitrary length and extract long dependencies in both forward and backward directions. We employed both GRU and LSTM to make the proposed model capable of remembering both short and longer sequences.

$$\overrightarrow{\mathbf{h}}_{t_{LSTM}} = \overrightarrow{LSTM}(\mathbf{c}_t), t \in [1, m] \tag{16}$$

$$\overleftarrow{\mathbf{h}}_{t_{LSTM}} = \overleftarrow{LSTM}(\mathbf{c}_t), t \in [m, 1] \tag{17}$$

$$\overrightarrow{\mathbf{h}}_{t_{GRU}} = \overrightarrow{GRU}(\mathbf{c}_t), t \in [1, m] \tag{18}$$

$$\overleftarrow{\mathbf{h}}_{t_{GRU}} = \overleftarrow{GRU}(\mathbf{c}_t), t \in [m, 1] \tag{19}$$

For each word, w_t we can now obtain an annotation by concatenating forward and backward contexts as follows:

$$\mathbf{h}_{t_{LSTM}} = [\overrightarrow{\mathbf{h}}_{t_{LSTM}}, \overleftarrow{\mathbf{h}}_{t_{LSTM}}] \tag{20}$$

$$\mathbf{h}_{t_{GRU}} = [\overrightarrow{\mathbf{h}}_{t_{GRU}}, \overleftarrow{\mathbf{h}}_{t_{GRU}}] \tag{21}$$

The attention mechanism is applied on $h_{t_{LSTM}}$ and $h_{t_{GRU}}$ to make the model capable of paying more or less attention to different words in the comment. To achieve this, we modified the feature vector by extracting informative words in the comment as follows:

$$\mathbf{u}_{t_{LSTM}} = \tanh(\mathbf{W}_{w_{LSTM}} \mathbf{h}_{t_{LSTM}} + \mathbf{b}_{w_{LSTM}}) \tag{22}$$

$$\mathbf{u}_{t_{GRU}} = \tanh(\mathbf{W}_{w_{GRU}} \mathbf{h}_{t_{GRU}} + \mathbf{b}_{w_{GRU}}) \tag{23}$$

$$\boldsymbol{\alpha}_{t_{LSTM}} = \frac{\exp(\mathbf{u}_{t_{LSTM}}^{\top} \mathbf{u}_{w_{LSTM}})}{\sum_t \exp(\mathbf{u}_{t_{LSTM}}^{\top} \mathbf{u}_{w_{LSTM}})} \tag{24}$$

$$\boldsymbol{\alpha}_{t_{GRU}} = \frac{\exp(\mathbf{u}_{t_{GRU}}^{\top} \mathbf{u}_{w_{GRU}})}{\sum_t \exp(\mathbf{u}_{t_{GRU}}^{\top} \mathbf{u}_{w_{GRU}})} \tag{25}$$

$$\mathbf{s}_{LSTM} = \sum_t \boldsymbol{\alpha}_{t_{LSTM}} \mathbf{h}_{t_{LSTM}} \tag{26}$$

$$\mathbf{s}_{GRU} = \sum_t \boldsymbol{\alpha}_{t_{GRU}} \mathbf{h}_{t_{GRU}} \tag{27}$$

where u_t is a hidden representation of h_t and u_w is a context vector which is randomly initialized and jointly learned in the training phase. The importance of a word u_t is calculated using the similarity of u_t with u_w and is normalized as shown in Eqs. (24) and (25). These importance weights α_t are finally aggregated into s by applying a weighted sum on them. s is the comment vector and summarizes all the information of words in the comment.

After obtaining the final comment representation s, convolution operation is used to extract informative local features and decrease the dimensionality of the input data. Moreover, convolution enables the model to acquire position invariance. In ABCDM, two parallel convolution layers with different kernel size for each branch (i.e., the Bi-LSTM and Bi-GRU branches) is employed independently. In this layer, convolution is conducted in one dimension. Specifically, according to Eq. (13), two 1D-CNN with fixed number of filters and different window size are applied to the comment representation (i.e., the outputs of Bi-LSTM and Bi-GRU units) independently.

At this point, we have four outputs from the CNN layer, because two independent CNNs were applied to the outputs of Bi-LSTM and Bi-GRU layers. Now, maximum and average pooling layers are stacked independently on the outputs of CNNs to down sample their feature maps. This makes the resulting feature maps more robust to the positional changes of features. If we consider the number of filters f in the CNN layer, the final feature vector Lc for each pooling operation is $Lc_i = [lc_1, lc_2, \dots, lc_f], i \in [1, 8]$. We obtained 8 local feature maps because for each CNN, maximum and average pooling are applied independently.

These feature vectors are concatenated to form the final document vector. Thus, we have $Lc = [Lc_1, Lc_2, \ldots, Lc_8]$. Having obtained the Lc vector, we applied batch normalization, to accelerate the network training and reduce overfitting [69]. For predicting comments' sentiment polarity, a fully connected dense layer is used to transform the Lc vector into a high-level sentiment representation. The output of this layer is calculated as follows.

$$\mathbf{h}_d = \text{Relu}(\mathbf{W}_d\mathbf{h}_p + \mathbf{b}_d) \tag{28}$$

where h_p is the hidden representation obtained from applying batch normalization on the concatenation of pooling layers, and W_d and b_d are parameters learned in the training process. Finally, the output of the dense layer is fed to an output layer with sigmoid function for binary classification. The pseudo-code of ABCDM is shown in Algorithm 1.

Algorithm 1: Pseudo-code of ABCDM.

```
Data: Comment matrix $\mathbf{C} \in \mathbb{R}^{n\times p}$ and the pre-trained GloVe
      embedding matrix $W_g \in \mathbb{R}^{|V|\times e}$where $n$ is the number
      of comments, $p$ is the sequence padding length, $V$ is
      the vocabulary and $e$ is the embedding dimension.
Result: Comments' polarity vector
        $\mathbf{P} = \{p_i \in \{0, 1\} : i \in [1, n]\}$
1  begin
2      Construct the word embedding matrix $\mathbf{C_e} \in \mathbb{R}^{n\times p\times e}$ using
       $W_g$ according to Eq. (15)
3      $branches$ = {Bi-LSTM, Bi-GRU}
4      for $br \in branches$ do
5          Apply $br$ on $C_e$ to obtain both future and preceding
           contexts $\overrightarrow{h}_{t_{br}}$, $\overleftarrow{h}_{t_{br}}$ using Eqs. (16) to (19).
6          Construct $h_{br}$ using Eqs. (20) and (21), respectively.
7          Construct $u_{t_{br}}$ using Eqs. (22) and (23), respectively.
8          $num = \exp(\mathbf{u}_{t_{br}}^{\intercal}\mathbf{u}_{w_{br}})$
9          $sum \longleftarrow 0$
10         for $t \in [1, m]$ do
11             $sum \longleftarrow sum + \exp(\mathbf{u}_{t_{br}}^{\intercal}\mathbf{u}_{w_{br}})$
12         end
13         $\alpha_{t_{br}} = \frac{num}{sum}$ $s_{br} \longleftarrow 0$
14         for $t \in [1, m]$ do
15             $s_{br} \longleftarrow s_{br} + (\alpha_{t_{br}} h_{t_{br}})$
16         end
17         $Pooling = \{GlobalMax1D, GlobalAvg1D\}$
18         $Lc \longleftarrow \emptyset$
19         for $f \in FilterSize$ do
20             $i \longleftarrow 0$
21             $Lc_{br}^{FilterSize} \longleftarrow \emptyset$
22             while $i \neq NoOfFilters$ do
23                 $lc_i \longleftarrow$ 1D-CNN($s_{br}, FilterSize$)
24                 $Append(Lc_{br}^{FilterSize}, lc)$
25                 $i \longleftarrow i + 1$
26             end
27             for $p \in Pooling$ do
28                 $Append(Lc, Apply(p, Lc_{br}^{FilterSize}))$
29             end
30         end
31     end
32     $Apply(Lc, batchnormalization)$
33     Construct $h_d$ using Eq. (28).
34     Feed $h_d$ into a sigmoid function for binary classification.
35     Update parameters of the model using the binary
       cross-entropy loss function with the Adam method.
36 end
```

Although there are few similar studies in the literature [70,71] that proposed the combination of CNN and LSTM, they have differences with our ABCDM model and them. For example, in [70], several parallel CNN layers are applied on the outputs of the embedding layer to extract n-grams features. These features are then used as the inputs to the RNN layer. There are several differences between this work and ours. First, their model was proposed for multi-label text classification while ours is proposed for single-label multi-class classification. Second, their model does not have any attention layer while ours employed an attention layer to pay more or less attention to different words in a comment. Third, their model uses original RNN while we employed bidirectional LSTM and GRU layers to consider both forward and backward context in the sentence. In another example is [71] have proposed an ensemble method to combine CNN and BiLSTM. Similar to our model their model used a bidirectional LSTM to extract both forward and backward context. However, their model used CNN and BiLSTM in parallel and used average probability scores of these models as final predictions. Our model, in contrast, concatenated the features extracted from bidirectional layers and then applied to the CNN to capture the local information. Moreover, we employed an attention layer to assign various weights to different words in the comment according to their importance.

5. Experiments and results

5.1. Experimental setup

5.1.1. Datasets

In the current study, ABCDM is evaluated on long review and short Twitter datasets for sentiment analysis using the following datasets.

- App: Apps for Android dataset [72]. This dataset contains 752,937 product reviews and metadata from Amazon.
- Kindle: Kindle Store dataset [72]. It contains 982,619 product reviews and metadata from Amazon.
- Movies: Movies and TV dataset [73]. It contains 1,697,533 product reviews and metadata from Amazon.
- Electronics: Electronics dataset [72]. This dataset contains 1,689,188 product reviews and metadata from Amazon.
- CDs: CDs and Vinyl dataset [72]. It contains 1,097,592 product reviews and metadata from Amazon.
- Airline Twitter: Airline Twitter Sentiment dataset [74]. It contains 14,641 tweets about the problems of each major U.S. airline. It was scraped from February of 2015.
- Sentiment140: Sentiment140 dataset was created by computer science graduate students at Stanford University [75]. It is a balanced dataset contains 1,600,000 tweets automatically categorized into negative and positive classes.
- T4SA: Twitter for Sentiment Analysis dataset [76] that contains the sentiment classification of 1,179,957 selected tweets from the multimodal T4SA dataset.

The statistics of the above-mentioned datasets are described in more details in Table 2 and Figs. 5 and 5(b).

5.1.2. Parameter settings

ABCDM and the other baseline models are implemented using the Keras library which is a high-level neural networks API, written in Python. To construct the input comment matrix C, 100,000 words are considered in the Tokenizer method and common preprocessing steps are applied to the datasets during the tokenization process. As shown in Figs. 5 and 5(b), we considered the 100 and 45 first words of comments in the review and tweet datasets by setting the padding size to 100 and 45, respectively.

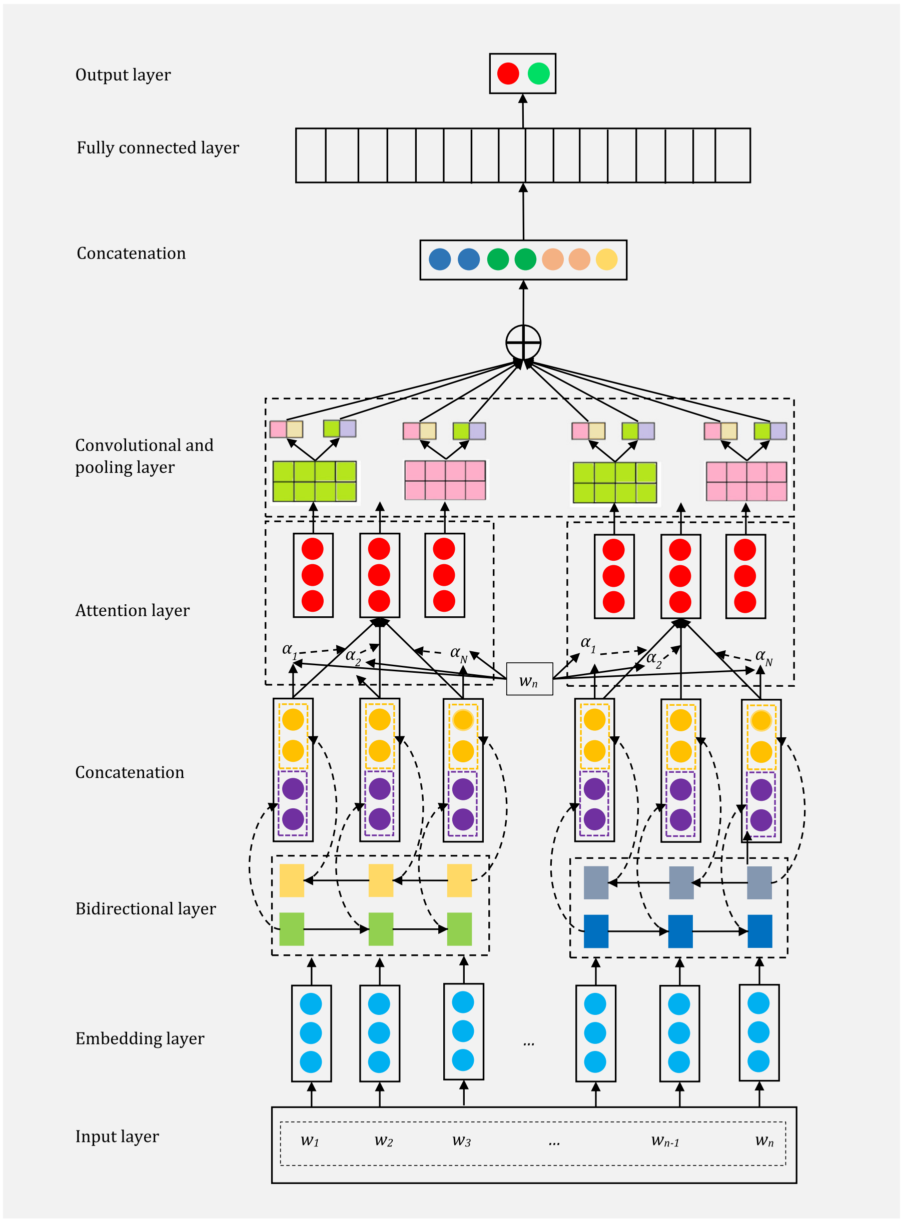


Fig. 4. Proposed architecture of ABCDM.

Table 2
Details of the datasets used in this study.

Type	Unbalanced							Balanced	
	Dataset	Total	5-star	4-star	3-star	2-star	1-star	Positive	Negative
Review	App	752937	386637	158081	85121	44385	78713	123098	123098
	Kindle	982619	575264	254013	96194	34130	23018	57148	57148
	Movies	1697533	906608	382994	201302	102410	104219	206629	206629
	Electronics	1689188	1009026	347041	142257	82139	108725	190864	190864
	CDs	1097592	656676	246326	101824	46571	46195	92766	92766
Tweet	Airline Twitter	14641	–	–	–	–	–	2363	2363
	Sentiment140	1600000	–	–	–	–	–	800000	800000
	T4SA	1179957	–	–	–	–	–	179050	179050

Publicly available pre-trained GloVe word vectors are used as the weights in the embedding layer. The "Wikipedia 2014 + Gigaword 5" version of GloVe [8] that contains 6 billion tokens with vocabulary size of 400000 is used in the current study. The embedding size of 100 is considered in the embedding layer.

Bidirectional CuDNNGRU and CuDNNLSTM, each with 128 memory units are used in the sequential layer. In the convolutional layer, 32 filters with kernel size of 4 and 6 are used as shown in Fig. 4. The size of the dense fully connected layer is 64 and the sigmoid activation function is used as the binary classifier.

Table 3
Parameter settings.

BiLSTM Memory Units	CNN Filters	CNN Kernel Size	Dense Size	Avg. Accuracy
64	16	5, 7	128	0.890
64	32	4, 6	32	0.859
64	64	3, 5	128	0.882
128	16	3, 5	128	0.892
128	32	4, 6	64	0.905
128	64	4, 6	32	0.893
256	16	3, 5	32	0.879
256	32	5, 7	64	0.887
256	64	4, 6	32	0.881

The training batch size for is set as 512 and the dropout rate is 0.2. The Adam stochastic optimizer with the learning and decay rate of 10^{-3} and 10^{-10} are used to train the network using the back-propagation algorithm. Binary cross-entropy is used as the loss function and accuracy metric is calculated to detect the convergence. To prevent overfitting, 5-fold cross validation and early stopping with monitoring validation loss in max mode with patience of 3 is used in the training process.

To obtain reasonable values for number of BiLSTM memory units, number of CNN filters, kernel size, and neurons in the dense layer, we performed grid search technique on three values with each having four parameters as shown in Table 3. In this table for three tested values of BiLSTM memory units and three values of CNN filters, the best values of CNN kernel size and dense layer from their three tested values and corresponding average accuracy using all 8 datasets as shown. Other values of CNN Kernel Size and Dense layer size and their combination with BiLSTM Memory Units and three values of CNN filters are not shown due to space limitation.

5.1.3. Evaluation criteria

Five evaluation criteria namely Precision (Pr), Recall (Re), F1-measure ($F1$), and Accuracy (Acc) are used to assess the performance of the models. These criteria are extensively used in text classification and sentiment analysis tasks [20,77]. These criteria are calculated as follows:

$$Pr = \frac{TP}{TP+FP},$$

$$Re = \frac{TP}{TP+FN},$$

$$F1 = \frac{2 \times Pr \times Re}{Pr + Re},$$

$$Acc = \frac{TP+TN}{TP+FP+TN+FN},$$

where TP, TN, FP, and FN are true positive, true negative, false positive, and false negative, respectively [20].

5.2. Baseline methods

This study, benchmarks the following six similar DNN models developed for sentiment polarity classification, as they achieved good results. This six DNN models are given below:

- CRNN [37]: In this model, each sentence is considered as a region and a regional CNN is applied to the input word vectors. Then, max pooling is used to reduce the dimensionality of the local features. Finally, an LSTM layer is used to capture long dependencies and a linear decoder is used to predict continuous valence and arousal scores.
- IWV [18]: This model consists of three convolution layers, a max pooling layer and a fully connected layer designed for sentiment polarity classification.

Table 4
Comparison of the results obtained on the Kindle dataset.

Method	Class	Recall	Precision	F1	Accuracy
SS-BED	Pos	0.8308	0.9461	0.8827	0.8910
	Neg	0.9514	0.8521	0.8979	
HAN	Pos	0.8762	0.9352	0.9043	0.9075
	Neg	0.9388	0.8843	0.9104	
ARC	Pos	0.8718	0.9422	0.9054	0.9091
	Neg	0.9463	0.8811	0.9124	
CRNN	Pos	0.8833	0.9424	0.9116	0.9145
	Neg	0.9457	0.8908	0.9172	
IWV	Pos	0.8779	0.9354	0.9046	0.9080
	Neg	0.9380	0.8870	0.9109	
AC-BiLSTM	Pos	0.8553	0.9555	0.9018	0.9074
	Neg	**0.9595**	0.8705	0.9122	
ABCDM	Pos	**0.9088**	**0.9570**	**0.9322**	**0.9340**
	Neg	0.9591	**0.9134**	**0.9356**	

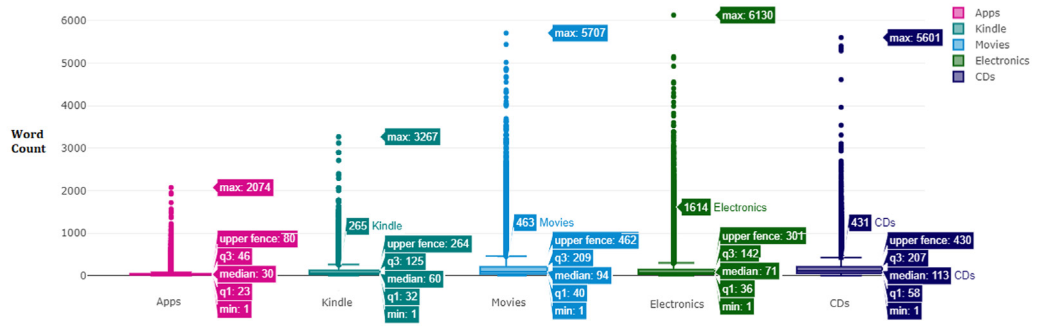
- SS-BED [16]: This model applies two parallel LSTM layers on two different word embedding matrices to learn semantic and sentiment feature representations. The output of the LSTM layers are then fed into a fully connected network with one hidden layer to predict emotion categories.
- HAN [35]: This model consists of four main parts: word sequence encoder which is a bidirectional GRU, a word-level attention layer which is used to form a weighted sentence vector, a sentence encoder which is another bidirectional GRU, and a sentence-level attention layer that rewards sentences correctly classify a document.
- ARC [36]: In this model, a one-layer bidirectional GRU is applied on the word vectors and the results are fed into an attention layer. The output of the attention mechanism is then fed into a CNN layer followed by a max-over time pooling operation, and a fully connected layer.
- AC-BiLSTM [17]: This model starts with a one dimension CNN layer with different filter sizes that is used for local feature extraction. The output of the CNN layer is fed into a bidirectional LSTM network which is followed by an attention mechanism. The output layer in this model consists of a dropout layer and a softmax layer.

5.3. Results

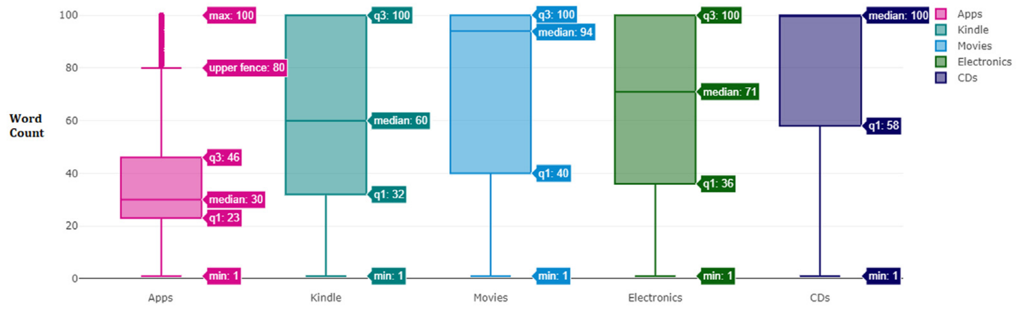
In this section, baseline comparisons are provided. Firstly, ABCDM is compared with six above-mentioned neural methods for sentiment analysis with long reviews. Secondly, a similar comparison is presented with short tweets. Finally, the performance of ABCDM is compared with a stacking method that aggregated the results obtained by all algorithms described in Table 1.

5.3.1. Analysis of the results on long reviews

The results obtained on five long review datasets are shown in Tables 4–8.

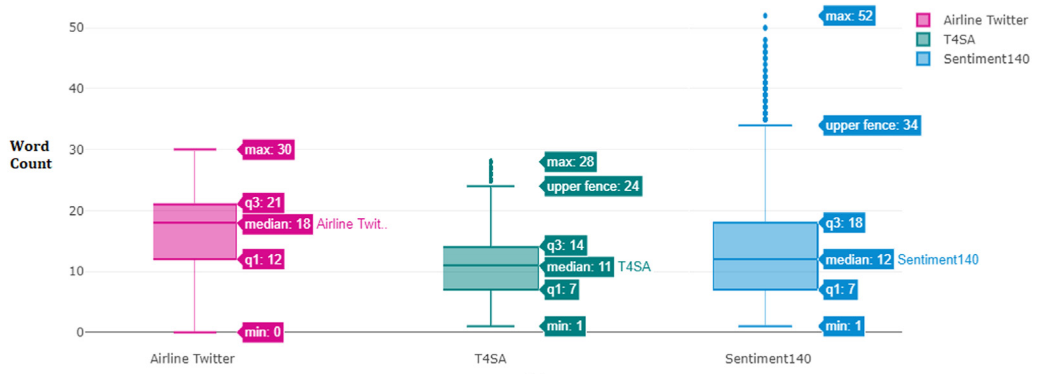


(a)



(b)

Fig. 5. Comparison of word count distributions of 5 review datasets (a) before and (b) after padding review length to 100 words.



(a)

Fig. 6. Comparison of word count distributions for 3 tweet datasets before padding review length to 45 words.

ABCDM achieved accuracy improvements as much as 1.95%, 1.85%, 3.0%, 2.61%, and 3.63% on Kindle, App, Movie, Electronics, and CD datasets as shown in Tables 4–8. For the F1 measure, these improvements for the positive class are 2.06%, 1.70%, 3.52%, 2.93%, and 4.39% and for the negative class are 1.84%, 1.48%, 2.58%, 2.33%, and 3.04%. It can be noted that our ABCDM outperformed the other six models in terms of both accuracy and F1 measures. These improvements are mainly benefited from (1) considering long dependencies existing in text using bidirectional LSTM and GRU layers, (2) using varying length local features by applying different sized CNN layers, and (3) assigning different weights to different words in the review according to their importance achieved by the attention layer. In order to better interpret the predictions of models, we plotted the Receiver Operating Characteristic (ROC) curves of the models in Fig. 7. ROC curve is used when the classes are balanced which is the case in our study. This helps to compare the models using different thresholds. Moreover, the Area Under the Curve (AUC) is compared to evaluate the performance of the model. A notable point in the results is that the performance improvement of ABCDM is lower for the Kindle and App datasets. This may be due to the average length of the reviews in these datasets. As shown in Fig. 5, the Kindle and App datasets have shorter reviews in average. As described in Algorithm 1, ABCDM applied two RNN-based layers on the embedding matrices to extract long-dependencies in text. Such long dependencies are more evident in longer reviews and this can justify the better performance of ABCDM on longer reviews.

Another point in the results is that the improvements for the positive class are higher in comparison to those for the negative class. This may be the result of the fact that local relations such as negations and comparisons are more prevalent in the negative reviews in comparison to the positive reviews and capturing such semantically negative relations is more difficult than positive relations.

Table 5
Comparison of the results obtained on the App dataset.

Method	Class	Recall	Precision	F1	Accuracy
SS-BED	Pos	0.8739	0.9273	0.8994	0.9024
	Neg	0.9310	0.8814	0.9052	
HAN	Pos	0.8811	0.9210	0.9005	0.9027
	Neg	0.9243	0.8862	0.9048	
ARC	Pos	0.8618	0.9372	0.8977	0.9019
	Neg	0.9420	0.8724	0.9057	
CRNN	Pos	0.8724	0.9354	0.9026	0.9060
	Neg	0.9396	0.8808	0.9091	
IWV	Pos	0.8720	0.9254	0.8977	0.9007
	Neg	0.9294	0.8793	0.9035	
AC-BiLSTM	Pos	0.8558	**0.9463**	0.8983	0.9033
	Neg	**0.9509**	0.8692	0.9079	
ABCDM	Pos	**0.8945**	0.9461	**0.9196**	**0.9218**
	Neg	0.9491	**0.9000**	**0.9239**	

Table 6
Comparison of the results obtained on the Movie dataset.

Method	Class	Recall	Precision	F1	Accuracy
SS-BED	Pos	0.8000	0.9096	0.8471	0.8586
	Neg	0.9173	0.8271	0.8675	
HAN	Pos	0.8220	0.9106	0.8632	0.8702
	Neg	0.9184	0.8388	0.8762	
ARC	Pos	0.8282	0.9019	0.8627	0.8687
	Neg	0.9091	0.8424	0.8739	
CRNN	Pos	0.7973	0.9142	0.8511	0.8611
	Neg	0.9249	0.8214	0.8697	
IWV	Pos	0.8301	0.8991	0.8627	0.8680
	Neg	0.9061	0.8428	0.8729	
AC-BiLSTM	Pos	0.8224	0.9211	0.8678	0.8755
	Neg	0.9286	0.8414	0.8821	
ABCDM	Pos	**0.8801**	**0.9274**	**0.9030**	**0.9055**
	Neg	**0.9310**	**0.8860**	**0.9079**	

Table 7
Comparison of the results obtained on the Electronics dataset.

Method	Class	Recall	Precision	F1	Accuracy
SS-BED	Pos	0.8351	0.8964	0.8633	0.8684
	Neg	0.9017	0.8476	0.8728	
HAN	Pos	0.8297	0.9135	0.8694	0.8755
	Neg	0.9212	0.8442	0.8809	
ARC	Pos	0.8184	0.9115	0.8615	0.8689
	Neg	0.9194	0.8365	0.8754	
CRNN	Pos	0.8295	0.9181	0.8708	0.8774
	Neg	0.9254	0.8456	0.8832	
IWV	Pos	0.8292	0.9092	0.8664	0.8725
	Neg	0.9159	0.8443	0.8779	
AC-BiLSTM	Pos	0.8280	0.9253	0.8736	0.8804
	Neg	0.9327	0.8449	0.8864	
ABCDM	Pos	**0.8701**	**0.9387**	**0.9029**	**0.9065**
	Neg	**0.9428**	**0.8791**	**0.9097**	

Table 8
Comparison of the results obtained on the CD dataset.

Method	Class	Recall	Precision	F1	Accuracy
SS-BED	Pos	0.6997	0.8937	0.7847	0.8082
	Neg	0.9165	0.7535	0.8269	
HAN	Pos	0.7800	0.9086	0.8392	0.8507
	Neg	0.9213	0.8076	0.8605	
ARC	Pos	0.7699	0.8994	0.8288	0.8416
	Neg	0.9133	0.7999	0.8524	
CRNN	Pos	0.7818	0.9036	0.8370	0.8487
	Neg	0.9155	0.8094	0.8585	
IWV	Pos	0.8021	0.8756	0.8362	0.8434
	Neg	0.8846	0.8189	0.8497	
AC-BiLSTM	Pos	0.7524	**0.9171**	0.8254	0.8419
	Neg	**0.9314**	0.7917	0.8553	
ABCDM	Pos	**0.8522**	0.9162	**0.8829**	**0.8870**
	Neg	0.9218	**0.8622**	**0.8909**	

Table 9
Comparison of the results obtained on the Airline Twitter dataset.

Method	Class	Recall	Precision	F1	Accuracy
SS-BED	Pos	0.9470	0.9403	0.9436	0.9100
	Neg	0.7658	0.7913	0.7772	
HAN	Pos	0.9349	0.9434	0.9390	0.9035
	Neg	0.7816	0.7574	0.7681	
ARC	Pos	0.9578	0.9460	0.9518	0.9229
	Neg	0.7870	0.8309	0.8070	
CRNN	Pos	0.9561	0.9448	0.9503	0.9205
	Neg	0.7824	0.8234	0.8012	
IWV	Pos	0.9369	0.9367	0.9355	0.8985
	Neg	0.7489	0.7861	0.7542	
AC-BiLSTM	Pos	0.9503	0.9459	0.9480	0.9172
	Neg	0.7888	0.8061	0.7963	
ABCDM	Pos	**0.9574**	**0.9520**	**0.9545**	**0.9275**
	Neg	**0.8112**	**0.8369**	**0.8209**	

Table 10
Comparison of the results obtained on the Sentiment140 dataset.

Method	Class	Recall	Precision	F1	Accuracy
SS-BED	Pos	0.8883	0.7601	0.8191	0.8038
	Neg	0.7192	0.8657	0.7855	
HAN	Pos	0.8674	0.7617	0.8111	0.7979
	Neg	0.7284	0.8461	0.7828	
ARC	Pos	**0.9085**	0.7314	0.8103	0.7873
	Neg	0.6660	0.8795	0.7577	
CRNN	Pos	0.9039	0.7470	0.8180	0.7987
	Neg	0.6936	0.8782	0.7750	
IWV	Pos	0.8954	0.7588	0.8213	0.8052
	Neg	0.7149	0.8727	0.7857	
AC-BiLSTM	Pos	0.8871	**0.7766**	0.8280	0.8157
	Neg	0.7443	0.8686	0.8014	
ABCDM	Pos	0.9019	0.7729	**0.8323**	**0.8182**
	Neg	**0.7444**	**0.8825**	**0.8076**	

As an example consider the following two sentences. "I feel the foods in the restaurant are *quite good*" and "I *don't* feel the foods in the restaurant are *good*". In the first sentence, there is no distance between the word "quite" which is an intensifier and the word "good" which has a positive sense. However, in the second sentence, there is a 7-words distance between the word "don't" as a negation word and the word "good" as the sentiment-bearing word of the sentence. Such long distances between the sentiment polarity changer words in negative reviews may decrease the performance of the model.

5.3.2. Analysis of the results on short tweets

The results obtained on three short tweet datasets are shown in Tables 9–11 and Fig. 8. ABCDM achieved accuracy improvements as much as 0.46%, 0.25%, and 0.54%, on Airline Twitter, Sentiment140, and T4SA datasets as shown in Tables 9–11. For the F1 measure, these improvements for the positive class are 0.27%, 0.43%, and 0.28% and 2.39%, 0.61%, and 0.28% for the negative classes using the same three datasets.

From the results, it can be noted that ABCDM outperformed the other six models in terms of both accuracy and F1 measures with Twitter datasets. However, the amount of improvements is less than using the review datasets.

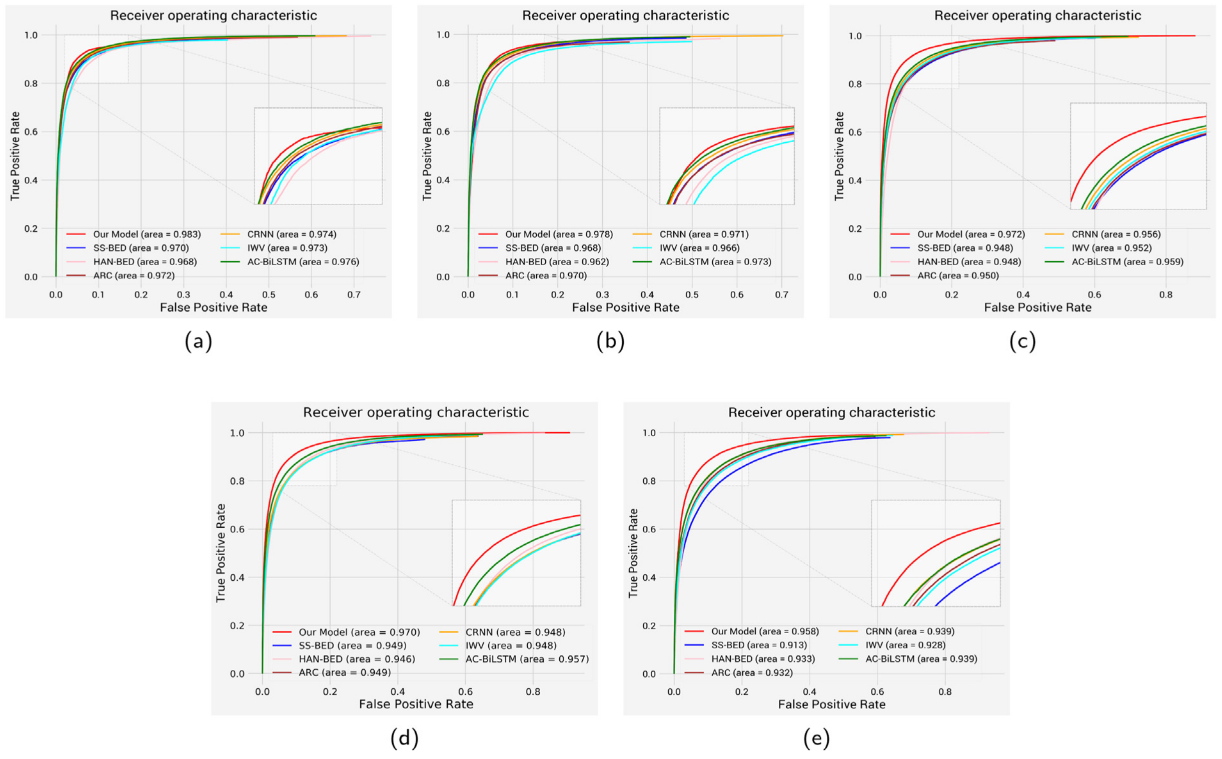


Fig. 7. Comparison of the results obtained using our proposed model on: (a) Kindle, (b) Apps, (c) Electronics, (d) Movies, and (e) CDs datasets.

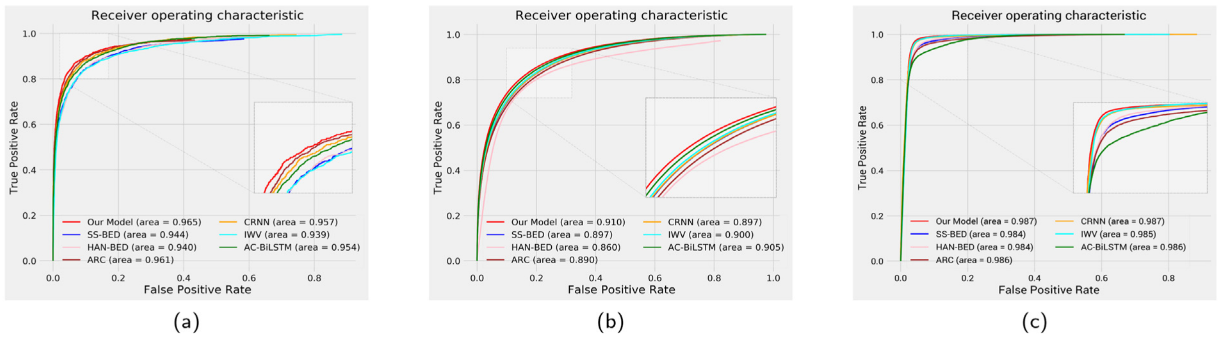


Fig. 8. Comparison of results obtained using our proposed model on: (a) Airline, (b) Sentiment140, and (c) T4SA datasets.

The key reason for this is Twitter datasets contains small number of words as shown in Fig. 6. As mentioned in the previous section, ABCDM does not yield significant improvement using short comments, because the first feature extraction layer in this model is an RNN-based network which is designed to capture long dependencies.

The rationale behind the higher improvement for positive reviews as compared to the negative ones does not hold, because as shown in Fig. 6, every tweet contains about 15 words in average which may be more than one sentences. This, results in having very short sentences in which there is not long distances between negation and other sentiment polarity changers. Therefore, there is no much differences between the structure of positive and negative tweets.

To show the performance of the proposed ABCDM model on positive and negative classes, we have shown the obtained true positive (TP), false positive (FP), true negative (TN), false negative (FN) for all eight datasets in Table 12.

To understand the significant differences between the proposed ABCDM and other six methods, we performed a Nemenyi post-hoc statistical test [78]. The results of 8 datasets is shown in Fig. 9. In this figure, the critical difference (CD) is shown on the top and the average ranks of the methods based on their accuracy is shown in the axis. The best performing algorithm is shown on the left side of the figure. A black horizontal line connects the methods that has no significant difference. As shown in Fig. 9, the proposed ABCDM method has significant difference as it has not connected with other methods.

5.4. Improving ABCDM through stack generalization

Stacked generalization is an ensemble method that trains a new model (level-1 model or meta-learner) to aggregate the outputs of models (level-0 models or base learners) which are already trained on the dataset [79]. Unlike simple ensemble methods such as voting, averaging, and weighted averaging, stack

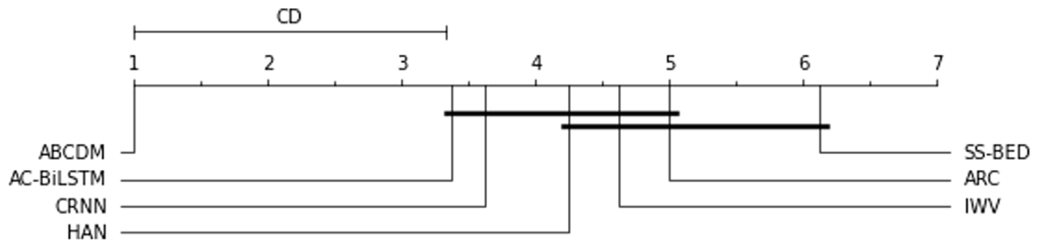


Fig. 9. The critical difference (CD) of the Nemenyi statistical test for comparing the mean-ranking of methods based on their accuracy on 8 datasets.

Table 11
Comparison of the results obtained on the T4SA dataset.

Method	Class	Recall	Precision	F1	Accuracy
SS-BED	Pos	0.9059	0.9756	0.9382	0.9412
	Neg	0.9765	0.9149	0.9438	
HAN	Pos	0.9469	0.9755	0.9610	0.9615
	Neg	0.9762	0.9484	0.9621	
ARC	Pos	0.9265	0.9881	0.9563	0.9576
	Neg	0.9888	0.9309	0.9590	
CRNN	Pos	0.9263	0.9889	0.9565	0.9579
	Neg	0.9895	0.9308	0.9592	
IWV	Pos	**0.9473**	0.9840	0.9652	0.9659
	Neg	0.9845	**0.9494**	0.9666	
AC-BiLSTM	Pos	0.9383	0.9878	0.9623	0.9633
	Neg	0.9883	0.9416	0.9643	
ABCDM	Pos	0.9466	**0.9904**	**0.9680**	**0.9687**
	Neg	**0.9908**	0.9489	**0.9694**	

Table 12
The obtained true positive (TP), false positive (FP), true negative (TN), false negative (FN) of the ABCDM method on the eight datasets.

Dataset	TP	FP	TN	FN
Kindle	51936	2332	54816	5212
APP	110111	6266	116832	12987
Movies	181854	14278	192350	24774
Electronics	166071	10898	179966	24793
CD	162654	14926	175938	28209
Airline Twitter	2262	242	2121	101
Sentiment140	721520	210800	589200	78479
T4SA	169489	1647	177403	9561

Algorithm 2: Pseudo-code for the stack generalization algorithm.

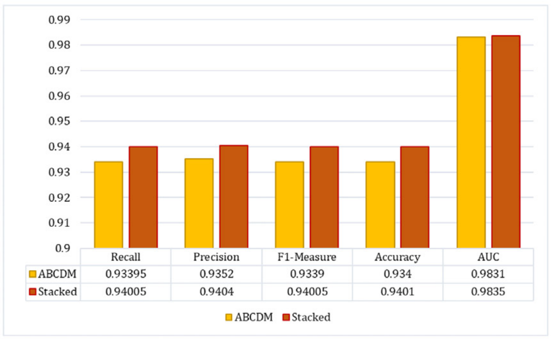
```
Data:   Training dataset
        $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$
        Level-0 learning algorithms $\mathcal{L}_1, \cdots, \mathcal{L}_M$
        Level-1 learning algorithms $\mathcal{L}$
        Test dataset $\mathcal{X}' = \{x'_1, x'_2, \cdots, x'_T\}$
Result: Prediction vector $\mathcal{Y}' = \{y'_1, y'_2, \cdots, y'_T\}$
begin
    Randomly split $\mathcal{D}$ into $I$ almost equal folds: $\mathcal{D}_1, \cdots, \mathcal{D}_I$
    $\mathcal{D}' = \varnothing$
    for $i = 1, \cdots, I$ do
        $\mathcal{D}^{-i} = \mathcal{D} - \mathcal{D}_i$
        $h = \varnothing$
        for $m = 1, \cdots, M$ do
            $h_m = \mathcal{L}_m(\mathcal{D}^{-i})$
        end
        $z = \varnothing$
        for $k = 1, \cdots, |\mathcal{D}^i|$ do
            $d = \varnothing$
            for $m = 1, \cdots, M$ do
                $d_m = h_m(\mathcal{D}^i_k[x])$
            end
            $z_k = (d, \mathcal{D}^i_k[y])$
        end
        $\mathcal{D}' = \mathcal{D}' \cup z$
    end
    $h' = \mathcal{L}(\mathcal{D}')$
    $\mathcal{Y}' = \varnothing$
    for $k = 1, \cdots, T$ do
        $z = \varnothing$
        for $m = 1, \cdots, M$ do
            $z_m = \mathcal{L}_m(x'_k)$
        end
        $\mathcal{Y}'_k = h'(z)$
    end
    return $\mathcal{Y}'$
end
```

generalization conditionally assign different weights to the inputs [80]. In the current study, the algorithm used for stack generalization is shown in Algorithm 2. In the current study, we considered the set of neural models shown in Table 1 as the algorithms and logistic regression as the level-1 meta learner. It is necessary for the level-0 learners to be accurate and diverse [79]. In the current study, we have the first condition for the level-0 models to have significantly lower classification error than random classifier. Due to their different structures, our level-0 models are also expected to be diverse, making errors at various instances. The results of applying stack generalization on our model is shown in Fig. 10.

As shown in Fig. 10, the stacked model outperformed the original ABCDM in all cases. This, besides the superiority of our ABCDM over other level-0 models, validates our hypothesis about the diversity of the models. A notable point is that in terms of accuracy and F1 measures, the improvement is significant while in terms of the AUC, the stacked model is slightly better or equal to the original ABCDM. This may be due to the higher AUCs for level-0 models as compared to their accuracy and F1 measures.

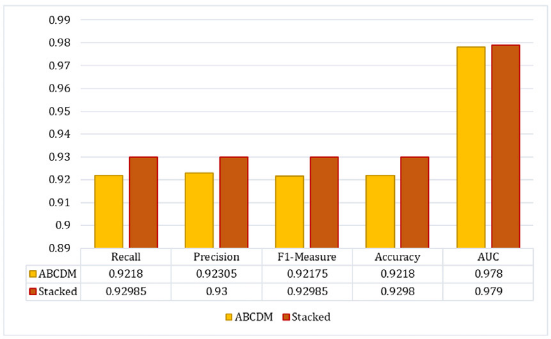
6. Conclusion

Nowadays, deep learning models in general and RNN and CNN models in specific have been widely used in the field of sentiment analysis These existing models have some drawbacks and the classification accuracy can be improved. In this study, a novel Attention-based Bidirectional CNN-RNN Deep Model (ABCDM) is proposed for sentiment analysis. ABCDM exploits publicly available pre-trained GloVe word embedding vectors as the initial weights of the embedding layer. On top of the embedding layer, two bidirectional LSTM and GRU networks are used to extract both past and future contexts as semantic representations of the input text. In order to pay more or less emphasis on different words in a comment, an attention layer is applied to the outputs of LSTM and GRU branches. This makes the semantic representations more informative.



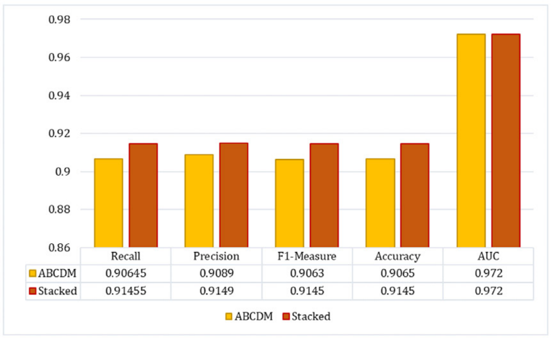
	Recall	Precision	F1-Measure	Accuracy	AUC
🟨 ABCDM	0.93395	0.9352	0.9339	0.934	0.9831
🟥 Stacked	0.94005	0.9404	0.94005	0.9401	0.9835

(a)



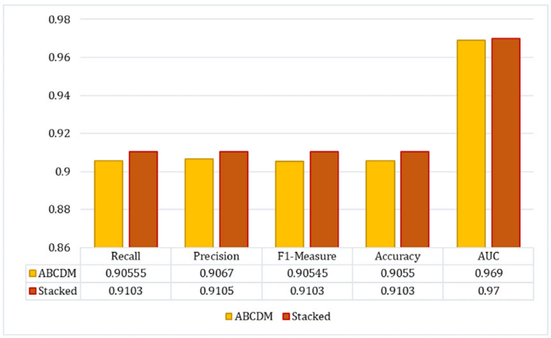
	Recall	Precision	F1-Measure	Accuracy	AUC
🟨 ABCDM	0.9218	0.92305	0.92175	0.9218	0.978
🟥 Stacked	0.92985	0.93	0.92985	0.9298	0.979

(b)



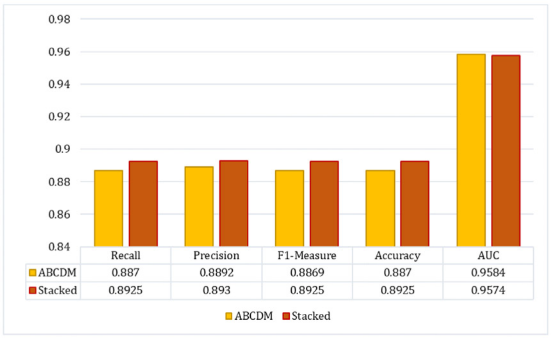
	Recall	Precision	F1-Measure	Accuracy	AUC
🟨 ABCDM	0.90645	0.9089	0.9063	0.9065	0.972
🟥 Stacked	0.91455	0.9149	0.9145	0.9145	0.972

(c)



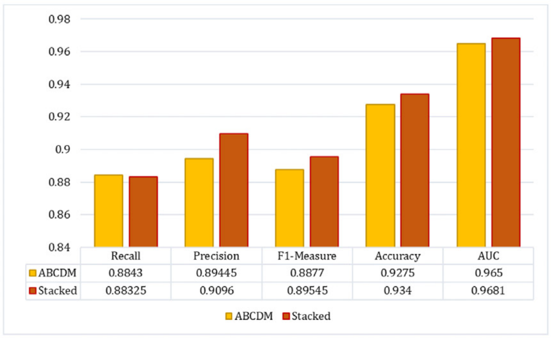
	Recall	Precision	F1-Measure	Accuracy	AUC
🟨 ABCDM	0.90555	0.9067	0.90545	0.9055	0.969
🟥 Stacked	0.9103	0.9105	0.9103	0.9103	0.97

(d)



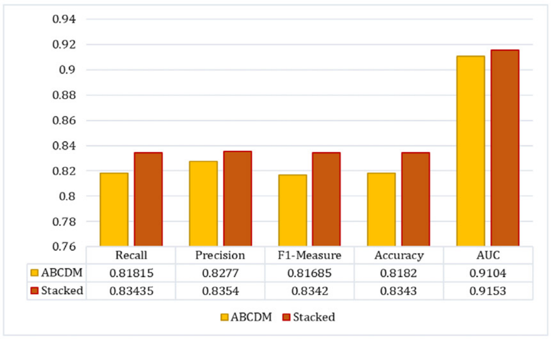
	Recall	Precision	F1-Measure	Accuracy	AUC
🟨 ABCDM	0.887	0.8892	0.8869	0.887	0.9584
🟥 Stacked	0.8925	0.893	0.8925	0.8925	0.9574

(e)



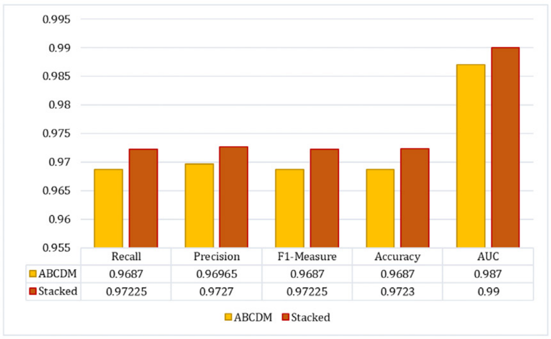
	Recall	Precision	F1-Measure	Accuracy	AUC
🟨 ABCDM	0.8843	0.89445	0.8877	0.9275	0.965
🟥 Stacked	0.88325	0.9096	0.89545	0.934	0.9681

(f)



	Recall	Precision	F1-Measure	Accuracy	AUC
🟨 ABCDM	0.81815	0.8277	0.81685	0.8182	0.9104
🟥 Stacked	0.83435	0.8354	0.8342	0.8343	0.9153

(g)



	Recall	Precision	F1-Measure	Accuracy	AUC
🟨 ABCDM	0.9687	0.96965	0.9687	0.9687	0.987
🟥 Stacked	0.97225	0.9727	0.97225	0.9723	0.99

(h)

Fig. 10. Comparison of the results obtained using our proposed model and the stacked version of the proposed model on: (a) Kindle, (b) App, (c) Electronics, (d) Movies, (e) CDs, (f) Airline Twitter, (g) Sentiment140, and (h) T4SA datasets.

These semantic representations are passed to a convolutional layer consisting of different kernel sizes to generate various feature maps and reduce the dimensionality of the feature space. Another motivation for employing CNN in ABCDM is to enable the model to extract local features in addition to those long dependencies extracted by LSTM and GRU layers. To make the resulting feature maps more robust to features' positional changes, maximum and average pooling layers are stacked independently at the outputs of CNNs. Finally, a dense fully connected layer with a sigmoid function is used to transform the vector into a high-level sentiment representation and perform the binary sentiment polarity classification of comments. Many experiments were conducted on five review and three Twitter datasets to evaluate the performance of developed model. Six recently published deep neural models for sentiment analysis are used for comparisons. Experimental results on these datasets indicate that ABCDM achieves state-of-the-art results on both long review and short tweet classification. Nonetheless, the comparison of the results obtained for review and tweet datasets shows that the amount of improvements on short tweet datasets is less than the similar case for the long review datasets. The key reason may be that the first feature extraction layer in ABCDM is the RNN-based network which is designed to capture long dependencies. To further improve the performance of ABCDM, a stack generalization algorithm is used in which ABCDM and six baseline algorithms are used as level-0 base learners, and logistic regression is used as level-1 meta learner. This stacked model outperforms all level-0 models, showing their diversity and different power of sentiment polarity classification.

This paper focused on polarity detection in document-level sentiment analysis. In future, we propose to investigate the effectiveness of our proposed ABCDM for other sentiment analysis tasks such as rating prediction and helpfulness prediction, as well as other levels including sentence- and aspect-level sentiment analysis. Finally, ABCDM has been developed for the English language but it could be easily extended to other languages.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A18A2b0046).

References

[1] A. Hussain, E. Cambria, Semi-supervised learning for big social data analysis, Neurocomputing 275 (2018) 1662–1673.
[2] Y. Xia, E. Cambria, A. Hussain, H. Zhao, Word polarity disambiguation using Bayesian model and opinion-level features, Cogn. Comput. 7 (3) (2015) 369–380.
[3] I. Chaturvedi, E. Ragusa, P. Gastaldo, R. Zunino, E. Cambria, Bayesian Network based extreme learning machine for subjectivity detection, J. Franklin Inst. 355 (4) (2018) 1780–1797.
[4] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: Advances in Neural Information Processing Systems, 2014, pp. 2177–2185.
[5] S. Lai, K. Liu, S. He, J. Zhao, How to generate a good word embedding, IEEE Intell. Syst. 31 (6) (2016) 5–14.
[6] M. Li, Q. Lu, Y. Long, L. Gui, Inferring affective meanings of words from word embedding, IEEE Trans. Affect. Comput. 8 (4) (2017) 443–456.
[7] M. Song, H. Park, K.-s. Shin, Attention-based long short-term memory network using sentiment lexicon embedding for aspect-level sentiment analysis in Korean, Inf. Process. Manage. 56 (3) (2019) 637–653.
[8] J. Pennington, R. Socher, C.D. Manning, Glove: Global vectors for word representation. 2014, 2018, URL: https://nlp.stanford.edu/projects/glove/ [accessed 2018-01-11][WebCite Cache ID 6wOYSJxnU].
[9] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, Deep learning for computer vision: A brief review, Comput. Intell. Neurosci. 1 (2018).
[10] I. Chaturvedi, R. Satapathy, S. Cavallari, E. Cambria, Fuzzy commonsense reasoning for multimodal sentiment analysis, Pattern Recognit. Lett. 125 (264–270) (2019).
[11] A. Kathua, A. Khatua, E. Cambria, A tale of two epidemics: Contextual Word2Vec for classifying twitter streams during outbreaks, Inf. Process. Manage. 56 (1) (2019) 247–257.
[12] F. Xing, E. Cambria, R. Welsch, Intelligent asset allocation via market sentiment views, IEEE Comput. Intell. Mag. 13 (4) (2018) 25–34.
[13] L. Zhang, S. Wang, B. Liu, Deep learning for sentiment analysis: A survey, Wiley Interdiscipl. Rev.: Data Min. Knowl. Disco. 8 (4) (2018) e1253.
[14] E. Cambria, H. Wang, B. White, Guest editorial: Big social data analysis, Knowl.-Based Syst. 69 (2014) 1–2.
[15] Y. Mehta, N. Majumder, A. Gelbukh, E. Cambria, Recent trends in deep learning based personality detection, Artif. Intell. Rev. 53 (2020) 2313–2339.
[16] A. Chatterjee, U. Gupta, M.K. Chinnakotla, R. Srikanth, M. Galley, P. Agrawal, Understanding emotions in text using deep learning and big data, Comput. Hum. Behav. 93 (2019) 309–317.
[17] G. Liu, J. Guo, Bidirectional LSTM with attention mechanism and convolutional layer for text classification, Neurocomputing 337 (2019) 325–338.
[18] S.M. Rezaeinia, R. Rahmani, A. Ghodsi, H. Veisi, Sentiment analysis based on improved pre-trained word embeddings, Expert Syst. Appl. 117 (2019) 139–147.
[19] U.A. Chauhan, M.T. Afzal, A. Shahid, M. Abdar, M.E. Basiri, X. Zhou, A comprehensive analysis of adverb types for mining user sentiments on amazon product reviews, World Wide Web (2020) 1–19.
[20] B. Liu, Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, Cambridge University Press, 2015.
[21] W. Zhao, H. Peng, S. Eger, E. Cambria, M. Yang, Towards scalable and reliable capsule networks for challenging NLP applications, in: ACL, 2019, pp. 1549–1559.
[22] A. Duric, F. Song, Feature selection for sentiment analysis based on content and syntax models, Decis. Support Syst. 53 (4) (2012) 704–711.
[23] A. Abbasi, S. France, Z. Zhang, H. Chen, Selecting attributes for sentiment classification using feature relation networks, IEEE Trans. Knowl. Data Eng. 23 (3) (2010) 447–462.
[24] S. Poria, I. Chaturvedi, E. Cambria, F. Bisio, Sentic LDA: Improving on LDA with semantic similarity for aspect-based sentiment analysis, in: IJCNN, 2016, pp. 4465–4473.
[25] I. Chaturvedi, Y.-S. Ong, I. Tsang, R. Welsch, E. Cambria, Learning word dependencies in text by means of a deep recurrent belief network, Knowl.-Based Syst. 108 (2016) 144–154.
[26] M.E. Basiri, A. Kabiri, Words are important: improving sentiment analysis in the persian language by lexicon refining, ACM Trans. Asian Low-Resour. Lang. Inf. Process. (TALLIP) 17 (4) (2018) 26.
[27] M.E. Basiri, N. Ghasem-Aghaee, A. Reza, Lexicon-based sentiment analysis in Persian, Curr. Future Dev. Artif. Intell. 1 (2017) 154.
[28] M.E. Basiri, A. Kabiri, HOMPer: A new hybrid system for opinion mining in the Persian language, J. Inf. Sci. (2019) 0165551519827886.
[29] M. Abdar, M.E. Basiri, J. Yin, M. Habibnezhad, G. Chi, S. Nemati, S. Asadi, Energy choices in Alaska: Mining people's perception and attitudes from geotagged tweets, Renew. Sustain. Energy Rev. 124 (2020) 109781.
[30] E. Cambria, Y. Li, F. Xing, S. Poria, K. Kwok, SenticNet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis, in: CIKM, 2020.
[31] L. Zhang, R. Ghosh, M. Dekhil, M. Hsu, B. Liu, Combining Lexicon-Based and Learning-Based Methods for Twitter Sentiment Analysis, HP Laboratories, Technical Report HPL-2011, 89, 2011.
[32] A. Mudinas, D. Zhang, M. Levene, Combining lexicon and learning based approaches for concept-level sentiment analysis, in: Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining, ACM, 2012, p. 5.
[33] M. Ghiassi, S. Lee, A domain transferable lexicon set for twitter sentiment analysis using a supervised machine learning approach, Expert Syst. Appl. 106 (2018) 197–216.
[34] P. Chikersal, S. Poria, E. Cambria, A. Gelbukh, C.E. Siong, Modelling public sentiment in twitter: Using linguistic patterns to enhance supervised learning, in: Computational Linguistics and Intelligent Text Processing, Springer, 2015, pp. 49–65.
[35] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1480–1489.
[36] S. Wen, J. Li, Recurrent convolutional neural network with attention for twitter and yelp sentiment classification: ARC model for sentiment classification, in: Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence, ACM, 2018, p. 49.
[37] J. Wang, L.-C. Yu, K.R. Lai, X. Zhang, Dimensional sentiment analysis using a regional CNN-LSTM model, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2016, pp. 225–230.
[38] S. Jameel, Z. Bouraoui, S. Schockaert, Unsupervised learning of distributional relation vectors, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, 2018, pp. 23–33.

[39] R. Sharma, A. Somani, L. Kumar, P. Bhattacharyya, Sentiment intensity ranking among adjectives using sentiment bearing word embeddings, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 547–552.

[40] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, M. Zhou, Sentiment embeddings with applications to sentiment analysis, IEEE Trans. Knowl. Data Eng. 28 (2) (2015) 496–509.

[41] S. Xiong, H. Lv, W. Zhao, D. Ji, Towards twitter sentiment classification by multi-level sentiment-enriched word embeddings, Neurocomputing 275 (2018) 2459–2466.

[42] M. Dragoni, G. Petrucci, A neural word embeddings approach for multi-domain sentiment analysis, IEEE Trans. Affect. Comput. 8 (4) (2017) 457–470.

[43] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent trends in deep learning based natural language processing, IEEE Comput. Intell. Mag. 13 (3) (2018) 55–75.

[44] J. Xu, D. Chen, X. Qiu, X. Huang, Cached long short-term memory neural networks for document-level sentiment classification, 2016, arXiv preprint arXiv:1610.04989.

[45] R. Moraes, J.F. Valiati, W.P.G. Neto, Document-level sentiment classification: An empirical comparison between SVM and ANN, Expert Syst. Appl. 40 (2) (2013) 621–633.

[46] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, B. Qin, Learning sentiment-specific word embedding for twitter sentiment classification, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014, pp. 1555–1565.

[47] D. Tang, B. Qin, X. Feng, T. Liu, Effective LSTMs for target-dependent sentiment classification, 2015, arXiv preprint arXiv:1512.01100.

[48] X. Zhu, P. Sobihani, H. Guo, Long short-term memory over recursive structures, in: International Conference on Machine Learning, 2015, pp. 1604–1612.

[49] A. Mousa, B. Schuller, 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: Volume 1, Long Papers, 2017, pp. 1023–1032.

[50] K.S. Tai, R. Socher, C.D. Manning, Improved semantic representations from tree-structured long short-term memory networks, 2015, arXiv preprint arXiv:1503.00075.

[51] Y. Ma, H. Peng, T. Khan, E. Cambria, A. Hussain, Sentic LSTM: a hybrid network for targeted aspect-based sentiment analysis, Cogn. Comput. 10 (4) (2018) 639–650.

[52] X. Zhou, X. Wan, J. Xiao, Attention-based LSTM network for cross-lingual sentiment classification, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 247–256.

[53] R. He, W.S. Lee, H.T. Ng, D. Dahlmeier, Exploiting document knowledge for aspect-level sentiment classification, 2018, arXiv preprint arXiv:1806.04346.

[54] R. He, W.S. Lee, H.T. Ng, D. Dahlmeier, Effective attention modeling for aspect-level sentiment classification, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 1121–1131.

[55] R. Johnson, T. Zhang, Effective use of word order for text categorization with convolutional neural networks, 2014, arXiv preprint arXiv:1412.1058.

[56] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, 2014, arXiv preprint arXiv:1404.2188.

[57] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.

[58] C. Dos Santos, M. Gatti, Deep convolutional neural networks for sentiment analysis of short texts in:, Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 2014, pp. 69–78.

[59] Y. Kim, Convolutional neural networks for sentence classification, 2014, arXiv preprint arXiv:1408.5882.

[60] H. Zhao, Z. Lu, P. Poupart, Self-adaptive hierarchical sentence model, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[61] M.E. Basiri, M. Abdar, M.A. Cifci, S. Nemati, U.R. Acharya, A novel method for sentiment classification of drug reviews using fusion of deep and machine learning techniques, Knowl.-Based Syst. (2020) 105949.

[62] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, J. Gao, Deep learning based text classification: a comprehensive review, 2020, arXiv preprint arXiv:2004.03705.

[63] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019, arXiv preprint arXiv:1907.11692.

[64] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, 2019, arXiv preprint arXiv:1909.11942.

[65] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, C. Zhang, Disan: Directional self-attention network for rnn/cnn-free language understanding, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[66] M. Sardelich, S. Manandhar, Multimodal deep learning for short-term stock volatility prediction, 2018, arXiv preprint arXiv:1812.10479.

[67] T. Chen, R. Xu, Y. He, Y. Xia, X. Wang, Learning user and product distributed representations using a sequence model for sentiment analysis, IEEE Comput. Intell. Mag. 11 (3) (2016) 34–44.

[68] H.T. Nguyen, M. Le Nguyen, An ensemble method with sentiment features and clustering support, Neurocomputing (2019).

[69] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, arXiv preprint arXiv:1502.03167.

[70] G. Chen, D. Ye, Z. Xing, J. Chen, E. Cambria, Ensemble application of convolutional and recurrent neural networks for multi-label text categorization, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 2377–2383.

[71] S. Minaee, E. Azimi, A. Abdolrashidi, Deep-sentiment: Sentiment analysis using ensemble of CNN and bi-lstm models, 2019, arXiv preprint arXiv:1904.04206.

[72] R. He, J. McAuley, Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering, in: Proceedings of the 25th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2016, pp. 507–517.

[73] J. McAuley, C. Targett, Q. Shi, A. Van Den Hengel, Image-based recommendations on styles and substitutes, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2015, pp. 43–52.

[74] Airline twitter sentiment dataset, 2019, https://data.world/crowdflower/airline-twitter-sentiment. (Accessed 30 April 2019).

[75] A. Go, R. Bhayani, L. Huang, Twitter Sentiment Classification Using Distant Supervision, CS224N Project Report, Stanford, 1, 2009, p. 2009.

[76] L. Vadicamo, F. Carrara, A. Cimino, S. Cresci, F. Dell'Orletta, F. Falchi, M. Tesconi, Cross-media learning for image sentiment analysis in the wild, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 308–317.

[77] P. Jamadi Khiabani, M.E. Basiri, H. Rastegari, An improved evidence-based aggregation method for sentiment analysis, J. Inf. Sci. (2019) 0165551519837187.

[78] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (Jan) (2006) 1–30.

[79] M.P. Sesmero, A.I. Ledezma, A. Sanchis, Generating ensembles of heterogeneous classifiers using stacked generalization, Wiley Interdiscipl. Rev.: Data Min. Knowl. Discov. 5 (1) (2015) 21–34.

[80] Z. Ma, P. Wang, Z. Gao, R. Wang, K. Khalighi, Ensemble of machine learning algorithms using the stacked generalization approach to estimate the warfarin dose, PLoS One 13 (10) (2018) e0205872.

Mohammad Ehsan Basiri received his B.S. degree in software engineering from Shiraz University, Shiraz, Iran, in 2006, and his M.S. and Ph.D. degrees in artificial intelligence from the University of Isfahan, Isfahan, Iran, in 2009 and 2014, respectively. Since 2014, he has been an Assistant Professor with the Computer Engineering Department, Shahrekord University, Shahrekord, Iran. He is the author of three books and more than 35 articles. His current research interests include sentiment analysis, natural language processing, machine learning, and data mining.

Shahla Nemati was born in Shiraz, Iran, in 1982. She received her B.S. degree in hardware engineering from Shiraz University, Shiraz, in 2005, her M.S. degree from the Isfahan University of Technology, Isfahan, Iran, in 2008, and her Ph.D. degree in computer engineering from Isfahan University, Isfahan, in 2016. Since 2017, she has been an Assistant Professor with the Computer Engineering Department, Shahrekord University, Shahrekord, Iran. She has written several articles in the fields of data fusion, emotion recognition, affective computing, and audio processing. Her current research interests include data fusion, affective computing, and data mining.

Moloud Abdar received his master's degree in computer science and engineering from the University of Aizu, Aizu, Japan, in 2018. He is currently pursuing a Ph.D. degree with the Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Geelong, VIC, Australia. He has written several articles in the fields of data mining, machine learning and user modelling in some refereed international journals and conferences. He is also very active in several international conferences, including the TPC in the ECML PKDD 2020, IEEE AINA 2018-2021, IEEE NCA 2020 and several referred international journals as a reviewer. His research interests include data mining, machine learning, sentiment analysis, deep learning and medical image analysis.

Erik Cambria is the Founder of SenticNet, a Singapore-based company offering B2B sentiment analysis services, and an Associate Professor at Nanyang Technological University, where he also holds the appointment of Provost Chair in Computer Science and Engineering. He earned his Ph.D. through a joint programme between the University of Stirling and MIT Media Lab. Erik is recipient of many awards, e.g., the 2018 AI's 10 to Watch and the 2019 IEEE Outstanding Early Career award. He is Associate Editor of several journals, e.g., INFFUS, IEEE CIM, and KBS, Special Content Editor of FGCS, Department Editor of IEEE Intelligent Systems, Editorial Board Chair of Cognitive Computation, and is involved in many international conferences as area or program chair and invited speaker.

U. Rajendra Acharya is a senior faculty member at Ngee Ann Polytechnic, Singapore. He is also (i) Adjunct Professor at University of Malaya, Malaysia, (ii) Adjunct Professor at Asia University, Taiwan, (iii) Associate faculty at Singapore University of Social Sciences, Singapore and (iv) Adjunct Professor at University of Southern Queensland, Australia. He has published more than 500 papers with more than 33,000 citations in Google Scholar (h-index = 96). He has worked on various funded projects, with grants worth more than 5 million SGD. He is ranked in the top 1% of the Highly Cited Researchers for the last five consecutive years (2016 to 2020) in Computer Science according to the Essential Science Indicators of Thomson. He is on the editorial board of many journals.