Ahmad *et al. Hum. Cent. Comput. Inf. Sci.* (2019) 9:24
https://doi.org/10.1186/s13673-019-0185-6

Human-centric Computing and Information Sciences

RESEARCH Open Access

Detection and classification of social media-based extremist affiliations using sentiment analysis techniques

Shakeel Ahmad[1*], Muhammad Zubair Asghar[2], Fahad M. Alotaibi[3] and Irfanullah Awan[4]

*Correspondence:
sarahmad@kau.edu.sa
[1] Faculty of Computing and Information Technology at Rabigh (FCITR), King Abdulaziz University, Jeddah, Saudi Arabia
Full list of author information is available at the end of the article

Abstract

Identification and classification of extremist-related tweets is a hot issue. Extremist gangs have been involved in using social media sites like Facebook and Twitter for propagating their ideology and recruitment of individuals. This work aims at proposing a terrorism-related content analysis framework with the focus on classifying tweets into extremist and non-extremist classes. Based on user-generated social media posts on Twitter, we develop a tweet classification system using deep learning-based sentiment analysis techniques to classify the tweets as extremist or non-extremist. The experimental results are encouraging and provide a gateway for future researchers.

Keywords: Social media, Sentiment classification, Emotions, Extremist sentiments, Terrorism, Extremist affiliations, Deep learning

Introduction

With the tremendous increase in the use of social network sites like Twitter and Facebook, online community is exchanging information in the form of opinions, sentiments, emotions, and intentions, which reflect their affiliations and aptitude towards an entity, event and policy [1–3]. The propagation of extremist content has also been increasing and being considered as a serious issue in the recent era due to the rise of militant groups such as Irish Republican Army, Revolutionary Armed Forces of Colombia (FARC), Al Quaeda, ISIS (Daesh), Al Shabaab, Taliban, Hezbollah and others [4]. These groups have spread their roots not only at the community levels but also their networks are gaining control of social networking sites [5]. These networking sites are vulnerable and approachable platforms for the group strengthening, propaganda, brainwashing, and fundraising due to its massive impact on public sentiments and opinions.

Opinions expressed on such sites give an important clue about the activities and behavior of online users. Detection of such extremist content is important to analyze user sentiment towards some extremist group and to discourage such associated unlawful acts. It is also beneficial in terms of classifying user's extremist affiliation by filtering tweets prior to their onward transmission, recommendation or training AI Chatbot from tweets [6].

The traditional techniques of filtering extremist tweets arenot scalable, inspiring researchers to develop automated techniques. In this study, we focus on the problem of classifying a tweet as extremist or non-extremist. The task faces different challenges, such as different kinds of extremism, various targets and multiple ways of representing the same semantics. The existing studies of extremism informatics are based on classical machine learning techniques [7, 8] or use classical feature representation schemes followed by a classifier.

In their work, Wei et al. [8] proposed a machine learning based classification system for identifying extremist-related conversations on Twitter. Different features are investigated for identifying extremist behavior o Twitter based on public tweets by applying KNN classifier. Based on the social media communication, Azizan and Aziz [7] conducted a study for the detection of extremist affiliations using machine learning technique, namely Naïve Bayes algorithm. It has shown best results over other ML classifiers. However, in the state of the art work [7] performed on extremist affiliation detection, authors have applied machine learning classifier with classical features. Furthermore, they have classified user reviews into positive and negative sentiments reflecting affiliations with extremist groups. However, classification of tweets into positive and negative classes does not provide an efficient way of distinguishing between extremist and non-extremist tweets. Another, major limitation of their approach is that it lacks the ability to take the overall dependencies related to a sentences in a document. Therefore, the machine learning model does not provide an efficient way for classifying text into extremist and non-extremist.

To overcome the aforementioned limitations of state of the art study [7], we investigate deep learning-based sentiment analysis techniques, which have already shown promising performance across a large number of complicated problems in different domains like vision, speech and text analytics [9, 10]. We propose to apply LSTM-CNN model, which works as follows: (i) CNN model is applied for feature extraction, and (ii) LSTM model receives input from the output of the CNN model and retains the sequential correlation by taking into account the previous data for capturing the global dependencies of a sentence in the document with respect to tweet classification into extremist and non- extremist.

We take the task of extremist affiliation detection as a binary classification task. We take the training set $Tr = \{t1, t2, t3, \ldots..tn\}$ and class tags (labels) $has\ Extrimist\_affliation = \{yes, no\}$. Each tweet is assigned a tag. The aim is to design a model which can learn from the training data set and can classify a new tweet as either extremist or non-extremist. The Twitter-based messaging is a major element of communication among individuals and groups, including extremists and extremist's groups. Using this sort of communication, future terrorist activities can potentially be traced. We propose a technique to identify tweets containing such content. Additionally, we classify sentiments of users in terms of emotional affiliations expressed towards individuals and groups having extremist thoughts. For this purpose, we apply IBM Watson API for tone analysis [11].

In this work, we experiment with multiple Machine Learning (ML) classifiers such as Random Forest, Support Vector Machine, KN-Neighbors, Naïve Bayes Classifiers, and deep learning (DL) classifiers. The feature set for such classifiers is encoded by task-driven embedding trained over different classifiers: CNN, LSTM, and CNN + LSTM. As

baselines, we compare with feature set which consists of n-grams [12], TF–DF, and bag of words (BoW) [13].

The proposed system aims at applying deep learning-based sentiment analysis technique to answer following research questions:

RQ#1: How to recognize and classify tweets as extremist vs non-extremist, by applying deep learning-based sentiment analysis techniques?
RQ#2: What is the performance of classical feature sets liken-grams, bag-of-words, TF-IDF, bag-of-words (BoW) over word embedding learned using CNN, LSTM, FastText, and GRU?
RQ#3: What is the performance of proposed technique for extremist affiliation classification with respect to the state-of-the-art methods?
RQ#4: How to perform the sentiment classification of user reviews w.r.t emotional affiliations of Extremists on Twitter and Deep Web?

Following contributions are made in this study:

i. Classifying user reviews (Tweets) as extremist or non-extremist affiliations deep learning-based sentiment analysis techniques.
ii. To investigate the classical feature sets like n-grams, bag-of-words, TF-IDF, bag-of-words (BoW) over word embedding learned using CNN, LSTM, FastText, and GRU, for tweet classification as extremist and non-extremist.
iii. Sentiment classification of user reviews w.r.t emotional affiliations of Extremists on Twitter and Deep Web?
iv. Comparing the efficiency of the proposed model with other baseline methods.
v. Our method outperforms baseline methods by a significant margin in terms improved precision, recall, f-measure an accuracy.

The rest of the article is organized as follows: related work is presented in "Proposed approach" section; "Experimental setup" section presents proposed methodology; in "Conclusions and future work" section, we present experimental setup; in "Experimental setup" section, we analyze the results obtained from experiments and final section concludes the study and gives a recommendation for future work.

Related work

In this section, we present a review of relevant studies conducted on the classification of social media-based extremist affiliations.

With the development of machine learning, it has gradually been applied to the analysis of extremist content and sentiments. Ferrara et al. [5] applied machine learning techniques on social media text to detect the interaction of extremist users. The proposed system has experimented on a set of more than 20,000 tweets generated from extremist accounts, which were later suspended by Twitter. The main emphasis was on three tasks, namely: (i) detection of extremist users, (ii) identifying users having with extremist content, and (iii) predicting users' response to extremists' postings. The experiments are conducted in two dimensions, i.e. time-independent

and real-time prediction tasks. An accuracy of about 93% is achieved with respect to extremist detection. With the same purpose, a machine learning-based technique is proposed by [7] for classifying of extremist affiliations. The Naïve Bayes algorithm is applied with the classical feature set. The system is based on the classification of user reviews into positive and negative classes with less focus on identifying, which sentiment class (positive or negative) is associated with extremist communication. In contrast to Ferrara et al. [5] work, which mainly emphasizes on the classification extremist's affiliations on skewed data; their method applies NB algorithm on balanced data giving more robust results. However, the overall dependencies in the sentence are not considered. This issue can be handled by applying deep learning models based on word embedding features. Researchers have also begun to investigate various ways of automatically analyzing extremist affiliations in languages other than English. In this connection, Hartung et al. [13] proposed a machine learning technique for detecting extremist posts in German Twitter accounts. Different features are experimented, such as emotions, linguistic patterns, and textual clues. The system yielded improved results over the state-of-the-art works. Studies on classifying extremist affiliations in the context of social media content are also noticeable in illegal drug usage. For example, in their work on marijuana-related microblogs, Nguyen et al. [14] collected more than thirty thousand tweets pertaining to marijuana during 2016. The text mining technique provides some useful insights to the acquired data such as (i) user attitude can be categorized as positive or negative, (ii) more than 65% tweets are originated from mobile phones, and (iii) frequency of tweets on weekend is higher than other days.

Lexicon-based unsupervised techniques for sentiment classification mainly rely on some sentiment lexicon and sentiment scoring modules [15]. Like other areas of sentiment analysis, extremist affiliation has been investigated by Ryan et al. [16], by proposing a novel technique based on part-of-speech tagging and sentiment-driven detection of extremist writers from web forums. The study was based on about 1 million posts from more than 25,000 distinct users surfing on four extremist forums. The proposed method was based on the user's sentiment score, computed by aggregating the score of no. of negative posts, duration of negative posts and severity of negative posts. The system is flexible to detect online suspicious activities on extremist users. In 2012, Chalothorn and Ellman [17] proposed a sentiment analysis model to analyze online radical posts using different lexical resources such as SentiWordNet, WordNet and NLTK toolkit. The sentiment class and intensity of the text is computed. Initially, textual data were acquired from different web forums such as Montada and Qawem, and after performing necessary pre-processing tasks, different feature-driven measures were applied to detect and manipulate religious and extremists content. Experimental results show that Montada forum has more positive posting than the Qawem forum. It was concluded that Qawem forum is suffered from more radical postings. Another worth noting work is carried out by [18] by collecting a huge dataset from YouTube group with extremist ideology. Different sentiment analysis techniques were applied to examine the topics under discussion and classified into positive (radical) and negative (non-radical) classes. Furthermore, gender-wise sentiments were also highlighted to observe the opinions expressed by male and female users.

Unsupervised techniques like clustering, have successfully been applied in different domains like aspect-based sentiment analysis [19], stock prediction [20] and sentiment classification. Skillicorn [21], in his work on crime investigations, proposed a framework for the adversarial analysis of data. The framework is comprised of three major segments including data collection, detection of suspects, and finding of suspicious individuals using network-driven association methods. Another method is based on the data clustering and visual analysis techniques for the investigation and implementation of terrorism informatics [22]. For this purpose, authors have used Twitter to detect and classify terrorist events by utilizing civilian sentiments.

Hybrid approach for developing sentiment-based applications have received considerable attention of researchers in different domains, such as business, health-care and politics [23]. In such approaches, different features of supervised, unsupervised and semi supervised techniques are adopted [19]. In the context of extremist affiliation classification, Zeng et al. [24] worked on the Chinese text segmentation issue in terrorism domain using a suffix tree and mutual information. The core module uses mutual information and the suffix tree for manipulating data in terrorism domain. The technique has applicability for processing huge amount of Chinese textual data. Analyzing militant conversation from online conversation, Prentice et al. [25] investigated the intents and content generated during the militant's conversation on social media with respect to Gaza violence in 2008/2009. Over 50 online text conversations were analyzed by applying both qualitative and quantitative techniques. Their proposed system includes a manual coding approach to detect the presence of a persuasive metaphor and semantics of the underlying text.

The aforementioned studies on detection and classification of social media-based extremist affiliations have used different approaches, such as supervised machine learning, an unsupervised technique like lexicon-based and clustering-based, and hybrid models. However, there is a need to investigate the applicability of state of the art sentiment-based deep learning models for classifying extremist affiliations using social media content.

Proposed approach

Data collection

We used Twitter streaming API [26] to scrap tweets containing one or more extremism-related keywords (ISIS, bomb, suicide etc.). Furthermore, we also investigated different Dark Web forums, such as Al-Firdaws, Montada, alokab, and Islamic Network [27]. The first three are Arabic forums and third one is in English. We collected over 25,000 postings by translating the non-English postings to English using Python-based Google Translate API (https://pypi.org/project/googletrans/). Each review is matched with the seed words present in the manually built extremist's vocabulary lexicon acquired from BiSAL [28], a bilingual sentiment lexicon for analyzing dark web forums. In this way, all postings containing one or more keywords from the manually acquired lexicon, are collected. For this purpose, we used a python-based beautiful-soupscript (https://codeburst.io/web-scraping-101-with-python-beautiful-soup-bb617be1f486).

The acquired data is stored is in a machine readable ".CSV" file. In this way, we acquired manually tagged training datasets for conducting experiments. The

Table 1 Dataset statistics

Class label	# of Tweets	Avg. length of a Tweet	# of tokens with stop words	# of tokens without stop words
Extremist	12,754	10.47	86,971	74,354
Non-extremist	8432	8.92	64,183	52,485

Table 2 Top 25 frequently occurring terms

S. no.	Key-word/phrase	Term frequency (TF)	Document frequency (DF)	User frequency (UF)
3	ISIS	21,445	12,144	2113
4	Islamic State	24,791	15,342	3002
5	Daesh	20,812	11,762	2711
6	Jihadi	29,313	16,433	4244
7	Khalifa	15,787	9378	2001
9	Amaarat	22,524	13,487	3537
10	Khillafat	30,646	18,553	4423
11	Albaghdadi	26,822	16,543	3875
12	Khawarij	27,653	18,326	4131
13	Al-Qaeda	35,224	20,019	4836
14	Khud-kush/suicide	26,835	15,014	1374
15	Narcotics	20,431	11,387	2538
16	Kafir/kufar	39,766	21,172	5672
17	Poison	8256	5009	1265
18	Bomb	11,267	6871	2112
19	Drugs	25,726	14,873	4227
20	Smuggling	2007	1092	538
21	Slaughter	9221	6591	1337
22	Kill	24,946	13,764	3635
23	Attack	27,863	15,846	3753
24	Custody	11,911	7248	1622
25	Charity	2563	1373	951

training dataset is comprised of 12,754 tweets labeled as "*extremist*" and 8432 as "*non-extremist*" [29]. Table 1 shows the detail of the used dataset. Table 2 shows a sample list of frequently occurring terms in the dataset, showing term frequency (tf), document frequency (df) and user frequency (uf).

Preprocessing

We applied different preprocessing techniques, such as tokenization, stop word removal, case conversion, and special symbol removal [30]. The tokenization yields a set of unique tokens (356,242), which assist in building a vocabulary from the training set, used for encoding the text.

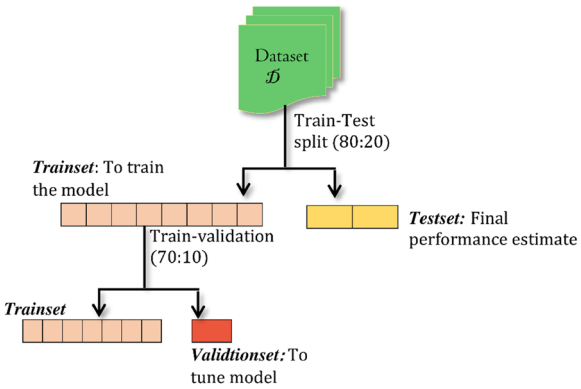

Fig. 1 Train, validation and test split

Table 3 A partial listing of training data

Tweet id	Tweet	Tweet label
1	Oh Allah, we are helpless	Non-extremist
2	Great news, ISIS fight Afghan forces to capture Helmand..	Extremist
3	Love you Baghdadi, Who is interested to know the truth about ISIS Yes.. visit our page to listen # Montada	Extremist
4	A very painful fight……operation zarbe-azab gave us a big blow #TTP	Extremist
5	What a great news, a suicide bomber destroyed the police station and killed 20 people	Extremist

Training, validation and testing

We divided the dataset into three parts: train, validate, and test. The DL model is trained with the Keras library [31] based on TensorFlow. The hardware requirements include 4 Titan X GPUson, a 128 GB memory with Intel Core i7 node. Figure 1 shows a diagramatic representation of train, validation and test split.

Training data Training data is used to train the model. In this work, 80% of the data is used training and it may vary as per requirements of the experiment. The training data includes both the input and the expected output. It includes both the input and the corresponding expected output [32].

Table 3 shows a sample list of review sentences in training data.

Data validation Data validation is used to minimize the overfitting and under fitting [33], which usually happens because the accuracy of the training phase is often high and performance gets degraded against test data. Therefore, 10% validation set is used to avoid performance error by applying parameter tuning. For this purpose, we applied automatic verification of dataset [34], which provides an unbiased evaluation of the model and minimize the overfitting [35].

Testing data The test data (20%) is used to check whether the trained model performs well on the unseen data. It is used for the final evaluation of the model when it is trained completely. A list of sample entries in the test data is presented in Table 4.

Table 4 A partial listing of test data

Tweet id	Tweet	Label
7	Baghdadi… our last hope, I simply love you #ISIS	Extremist
8	We condemn a suicide attack in Peshawar today	Non-extremist

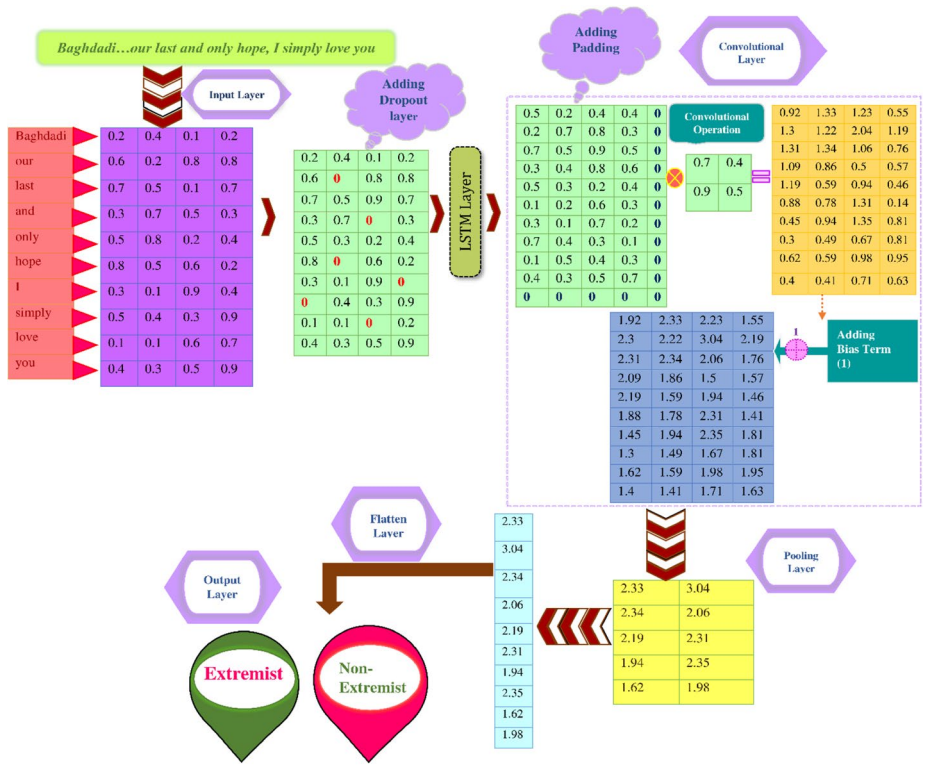

Fig. 2 LSTM + CNN architecture for extremist affiliation classification

Proposed network model

The proposed method implements and evaluates the performance of long short term memory with Convolutional Neural Network (CNN) model to identify tweets/reviews containing content with extremist clues. We train the neural classifier, for the classification of extremist affiliation content. The working flow of the network is comprised of following steps: (i) word embedding, in which each word in a sentence is assigned a unique index to form a fixed-length vector, (ii) dropout layer is used to avoid overfitting, (iii)LSTM layer is incorporated to capture long-distance dependency across tweets/reviews (iii) feature extraction is performed using a convolution operation, (iv) pooling layer aims at minimizing the dimension of feature map by aggregating the information, (v) Flatten layer converts the pooled feature map into a column vector, and (vi) at the output layer, softmax function is used for the classification. Figure 2 presents a network diagram for classifying the sentence as *"extremist" or "non-extremist content".* In the rest of this paper, we give detailed working of these layers.

Word embedding (input) layer

The embedding or input layer is the first layer of LSTM + CNN model, which transforms the words into real-valued vector representation, i.e. a vocabulary of the words is created, which is then converted into a numeric form, known as word embedding. The word embedding is given as input (sentence matrix) to the next layer. As shown in the pseudocode, there are different parameters, namely (i) max features, (ii) embed dim, and (iii) input length. The "*max_features*" holds the top words and presents the size of vocabulary; "*embed_dim*" shows the dimension of the real-valued vector, and the "*input_length*" describes the length of each of the input sequence.

The sentence consists of a sequence of words: x_1, $x_2 \ldots x_n$, and each word are assigned an exclusive index number. The embedding layer transforms such indices into D dimensional word vector. For this purpose, an embedding matrix of size [*vocabulary size* × *embedding size*]) is learned over a 10×4-dimensional matrix i.e. ($[V \times D] = [10 \times 4]$). As in this case, the vocabulary size is 10, while embedding size is 4, therefore, the individual word "*Baghdadi*" is represented as 1 by 4-dimensional vector i.e. $(1 \times D = [1 \times 4]$. For example, the word "*Baghdadi*" with index "1" contains an embedding vector [0.2, 0.4, 0.1, 0.2], represented by the first row shown in Fig. 3. Similarly, the second row is [0.6, 0.2, 0.8, 0.8] and same is the case for others. Thus, we can clearly see that each word has an embedding of size "$1 \times D$", as depicted in Fig. 3. The embedding matrix is denoted as $E \epsilon R^{V \times D}$. The word embedding process is illustrated as follows:

Dropout layer

The function of the dropout layer is to avoid overfitting. The value 0.5 represents the "*rate*" parameter of the dropout layer and the value of this parameter falls between 0 and 1 [36]. The dropout layer randomly deletes or turnoff the activation of neurons in

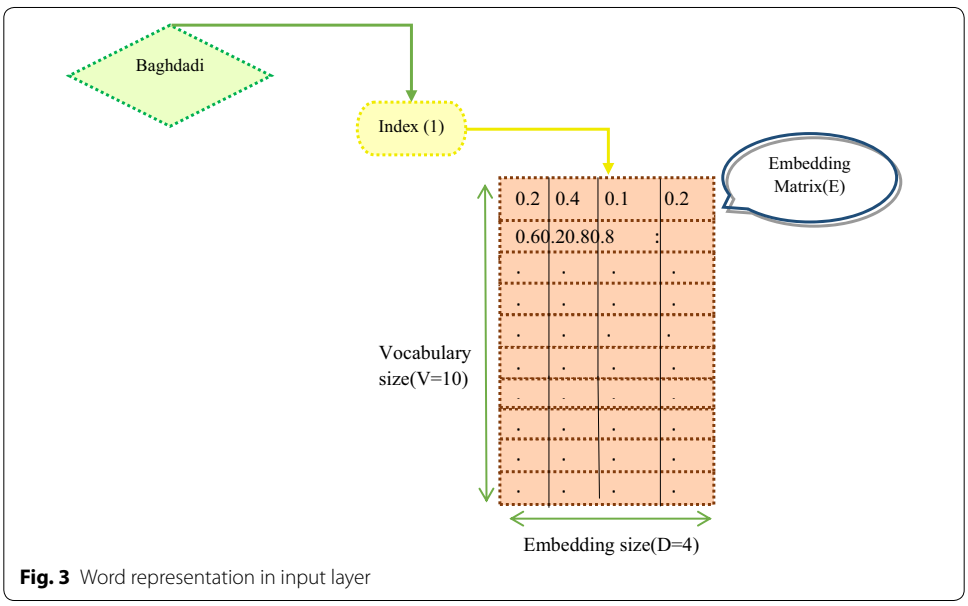

Fig. 3 Word representation in input layer

the embedding layer as the dropout is applied on embedding layer, whereas each neuron in the embedding layer depicts the dense representation of a word in a sentence. The modeling of dropout on a single neuron is presented in Eq. (1):

$$f(k,p) = \begin{cases} p & if\ k = 1 \\ 1-p & if\ k = 0 \end{cases} \tag{1}$$

k depicts the desirable results and p is the probability related to real-valued word representation. So, when p value is 1 the neuron holding a real value will be deleted and is activated otherwise. Figure 4 shows the working of dropout layer.

Figure 4 illustrates the embedding layer which holds the real-valued representation of a given sentence: *Baghdadi… our last and only hope, I simply love you.* so, after adding a dropout layer some of the values in the embedding layer are deactivated randomly (Fig. 4).

Long short term memory

We used single LSTM layer, which consists of a 100 lstm cells/units. LSTM performs some pre-calculations before it produces an output. In each cell, four independent calculations are performed using four gates: forget (ft), input (it), candidate (c ~ t) and output (ot). The equations for these gates are given below [37]:

$$ft = \sigma\left(Wfxt + Ufht - 1 + bf\right) \tag{2}$$

$$it = \sigma(Wixt + Uiht - 1 + bi) \tag{3}$$

$$Ot = \sigma(Woxt + Uoht - 1 + bo) \tag{4}$$

$$C \sim t = \tau(Wcxt + Ucht - 1 + bc) \tag{5}$$

$$Ct = ftoCt - 1 + itoC \sim t \tag{6}$$

$$ht = Oto\tau(Ct) \tag{7}$$

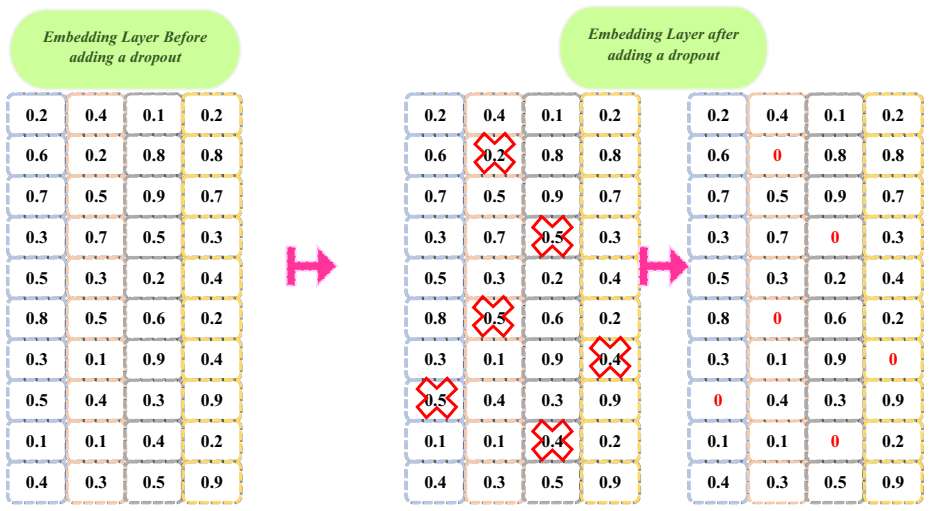

Fig. 4 Operation of dropout layer

The graphical illustration of entire LSTM cell in green block is shown in Fig. 5.

Now, the example sentence: "*Baghdadi... our last and only hope, I simply love you*", is passed through the LSTM cell. The execution first starts with the forget gate using Eq. (2) in which input x_t and the previous output h_t is multiplied with their respective weights *Wf, Uf*. Next bias *bf* is added which outputs (4×1) vector. Then sigmoid activation function is applied to transform the values between 0 and 1 and the values greater than 0.5 is assumed as 1 for three gates ft, it, Ot as shown in below computation [38]. Next, for input and output gate, the same procedure is repeated and for candidate gate, instead of the sigmoid, tangent function is used. Finally, the output h_t and the next cell state c_t vectors, are calculated using Eqs. (6) and (7).

Putting values in Eqs. (2), (3), (4), (5), (6), (7)

$$f_t = \sigma\left(\begin{bmatrix}0.1 & 0.3 & 0.2 & 0.4\\ 0.2 & 0.1 & 0.5 & 0.3\\ 0.3 & 0.6 & 0.3 & 0.2\\ 0.1 & 0.2 & 0.5 & 0.3\end{bmatrix} \times \begin{bmatrix}0.2\\0.4\\0.1\\0.2\end{bmatrix} + \begin{bmatrix}0.1 & 0.2 & 0.5 & 0.6\\ 0.3 & 0.4 & 0.1 & 0.7\\ 0.2 & 0.3 & 0.5 & 0.4\\ 0.4 & 0.1 & 0.3 & 0.2\end{bmatrix} \times \begin{bmatrix}0\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0.1\\0.2\\0.3\\0.4\end{bmatrix}\right) = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$$

$$i_t = \sigma\left(\begin{bmatrix}0.4 & 0.3 & 0.1 & 0.5\\ 0.2 & 0.6 & 0.4 & 0.1\\ 0.3 & 0.5 & 0.6 & 0.7\\ 0.2 & 0.1 & 0.4 & 0.3\end{bmatrix} \times \begin{bmatrix}0.2\\0.4\\0.1\\0.2\end{bmatrix} + \begin{bmatrix}0.5 & 0.1 & 0.4 & 0.6\\ 0.4 & 0.3 & 0.2 & 0.7\\ 0.6 & 0.2 & 0.4 & 0.8\\ 0.2 & 0.7 & 0.8 & 0.1\end{bmatrix} \times \begin{bmatrix}0\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0.1\\0.3\\0.4\\0.2\end{bmatrix}\right) = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$$

$$c \sim t = \tau\left(\begin{bmatrix}0.3 & 0.2 & 0.5 & 0.6\\ 0.1 & 0.3 & 0.4 & 0.2\\ 0.5 & 0.6 & 0.7 & 0.1\\ 0.4 & 0.3 & 0.2 & 0.7\end{bmatrix} \times \begin{bmatrix}0.2\\0.4\\0.1\\0.2\end{bmatrix} + \begin{bmatrix}0.5 & 0.1 & 0.4 & 0.6\\ 0.4 & 0.3 & 0.2 & 0.7\\ 0.6 & 0.2 & 0.4 & 0.8\\ 0.2 & 0.7 & 0.8 & 0.1\end{bmatrix} \times \begin{bmatrix}0\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0.4\\0.1\\0.2\\0.3\end{bmatrix}\right)$$
$$= \begin{bmatrix}0.6 & 0.3 & 0.5 & 0.5\end{bmatrix}$$

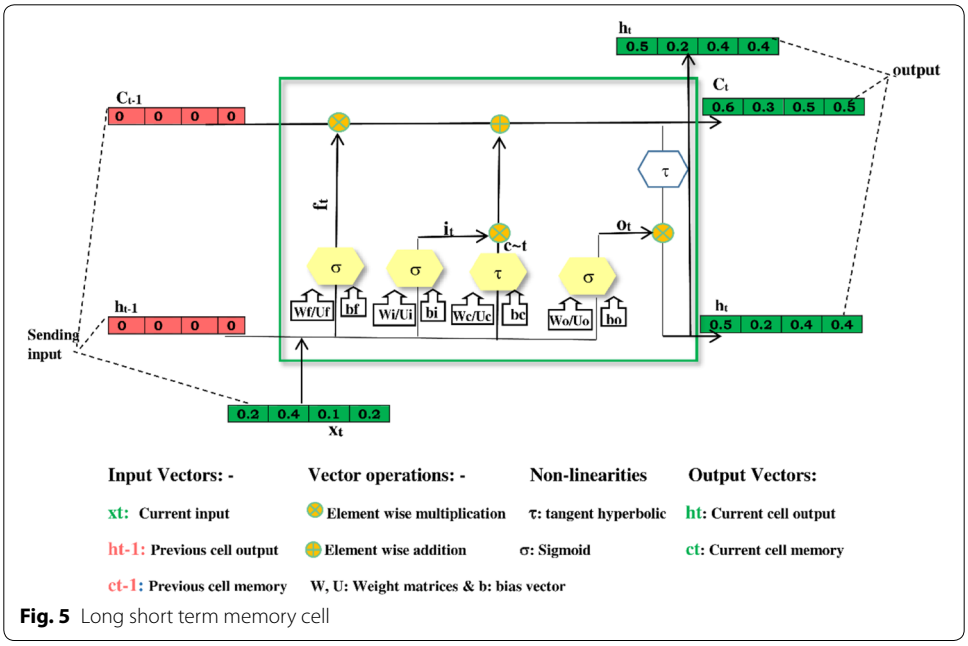

Fig. 5 Long short term memory cell

$$o_t = \sigma\left(\begin{bmatrix}0.2 & 0.1 & 0.5 & 0.6\\ 0.3 & 0.4 & 0.2 & 0.1\\ 0.1 & 0.5 & 0.6 & 0.7\\ 0.6 & 0.4 & 0.3 & 0.2\end{bmatrix} \times \begin{bmatrix}0.2\\0.4\\0.1\\0.2\end{bmatrix} + \begin{bmatrix}0.1 & 0.2 & 0.5 & 0.3\\ 0.6 & 0.3 & 0.2 & 0.4\\ 0.7 & 0.4 & 0.1 & 0.8\\ 0.8 & 0.5 & 0.2 & 0.3\end{bmatrix} \times \begin{bmatrix}0\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0.7\\0.6\\0.5\\0.4\end{bmatrix}\right) = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$$

$$c_t = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix} \cdot \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix} + \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix} \cdot \begin{bmatrix}0.6 & 0.3 & 0.5 & 0.5\end{bmatrix}$$

$$h_t = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix} \cdot \tau\begin{bmatrix}0.6 & 0.3 & 0.5 & 0.5\end{bmatrix} = \begin{bmatrix}0.5 & 0.2 & 0.4 & 0.4\end{bmatrix}$$

From here, we pass our h_t and c_t to begin the next lstm cell calculations. As a result, LSTM produces an output sequence $P = [p_0, p_1, p_2, \ldots\ldots, p_l]$ of a matrix $P \in R^{lxw}$. Finally, this representation is fed to the CNN layer.

Convolutional layer

In this layer, a convolutional operation is performed which is a mathematical operation implemented on two functions, yielding a third function. To perform the convolutional operation, the dimensions of an input matrix (P), filter matrix (F) and output matrix (T), are represented as follows:

$$P = R^{lxw} \tag{8}$$

In Eq. (8), P represents input matrix, produced by the LSTM layer, R denotes all real numbers, l is the length and w is the width of input matrix which is shown as $R^{10\times4}$.

$$F = R^{nxm} \tag{9}$$

In Eq. (9), F represents filter matrix, R denotes all real numbers, n is the length and m is the width of the filter matrix, which is shown as $R^{2\times2}$, and

$$T = R^{sxd} \tag{10}$$

In Eq. (10), T represents output matrix, R denotes all real numbers, s is the length and d is the width of output matrix, which is shown as $R^{10\times4}$.

The convolutional operation is formulated as shown in Eq. (11):

$$t_{i,j} = \sum_{l=1}^{n}\sum_{w=1}^{m} f_{l,w} \otimes p_{i+l-1,j+w-1} \tag{11}$$

where, $t_{i,j}\ \epsilon\ R^{s\times d}$ is the tth element of output matrix, $f_{l,w}\ \epsilon\ R^{n\times n}$ represents fth element of weight matrix, $\otimes$ denotes element-wise cross multiplication, and $p_{i+l-1,j+w-1}\ \epsilon\ R^{l\times w}$ represents the pth elements of the input matrix.

For the example sentence: "*Baghdadi… our last and only hope, I simply love you*", the convolutional operation is executed as follows: (i) *Elements of input matrix* 0.5, 0.2, 0.2, 0.7, (ii) *Elements of filter matrix* 0.7, 0.4, 0.9, 0.5, and (iii) *Convolutional operation* $0.5\times0.7+0.2\times0.4+0.2\times0.7+0.7\times0.5=0.92$, where 0.92 is the first element of the output matrix. Similarly, the process of element-wise cross multiplication and addition continues until all the values of input matrix are covered and this will be done through the sliding of the filter over the input matrix.

Feature map After adding bias and an activation function to the output matrix, a feature map (A) for a given sentence is computed as follows [see Eq. (12)]:

$$A = a_{i,j} = f\left(t_{i,j} + b\right) \tag{12}$$

where dimension of feature map (A) for a given sentence $= R^{q\times r} = R^{10\times 4}$, where, $a_{i,j} \in R^{q\times r}$, is an ath element of a feature map, b is a bias term, and f is an activation function.

In Fig. 2, each element of the output matrix is then added to the bias term after the convolutional operation. For example, adding a bias value to the first element of the output matrix that is $0.92 + 1 = 1.92$ which represents the first element of the feature map for a given sentence (Fig. 2).

As, shown in Algorithm no. 1, the parameters used in this layer are: (i) *"filters"*, it describes the filter number within the convolutional layer; (ii) *"kernel_size"*, shows the dimensionality of convolutional window; (iii) *"padding"* holds single value among the three values: *"valid"*, *"same"*, and *"casual"*. If padding contains the value of *"valid"*, then it shows no padding, and padding with the value *"same"*, depicts that original input length equals to output length. Furthermore, when padding contains the value *"casual"*, then it produces dilated convolution; and (iv) the parameter "*activation = relu*" means activation is exploited to reveal nonlinearity.

Finally, a feature map is generated on which relu activation function is applied to remove non-linearity and its mathematical expression is: *Output = max (Zero, Input)*, where Input means an element of the feature map.

For example, in the input sentence, the first element of the feature map is 1. 92. If we apply relu activation function on it, then *Output = max (0, 1.92)*, will be *Output* = 1.92, since $1.92 > 0$. So, in this way, other elements of rectified feature map for a given sentence is calculated.

Pooling layer

The pooling layer is used to minimize the dimension of the feature map by aggregating the information. So, the max pooling is applied to every sentence in a dataset. We used max pooling to get the required feature of a sentence by selecting the maximum value. Equation (13) presents the formula for pooling layer.

$$g_{p,h} = max_{i,j\in Z_{p,h}} a_{i,j} \tag{13}$$

Suppose, $Z_{p,h}$ is a small matrix with size 2×2, also $g_{p,h} \in R^{5\times 2}$ is an element of matrix G, and $a_{i,j}$ is an element of matrix A. The elements $g_{p,h}$ of matrix G is obtained after picking the maximum element from the matrix A, which is a rectified feature map for a given sentence, within the given window matrix that is $Z_{p,h}$. Hence, the matrix G depicts the pooled feature map of the given sentence, which is illustrated in Fig. 6.

A pooled feature map is created by setting a window size (2×2), placing it on the feature map, and finally extracting the maximal element inside the window. Since in our case, among the selected window size, that is max (1.92, 2.3, 2.33, 2.22), the largest element is 2.33, which shows the first element of the pooled feature map related to the given sentence. Hence, the same procedure will be performed for the other values of the pooled feature map.

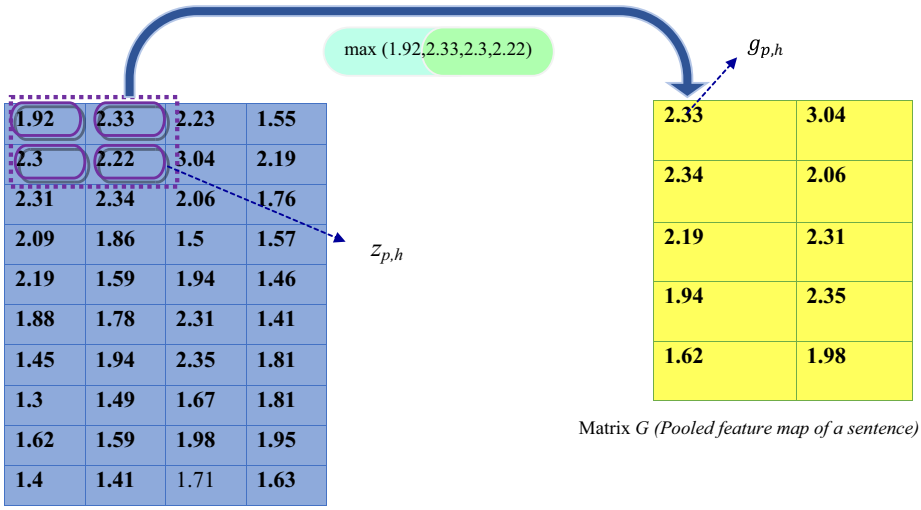

Fig. 6 Pooling layer

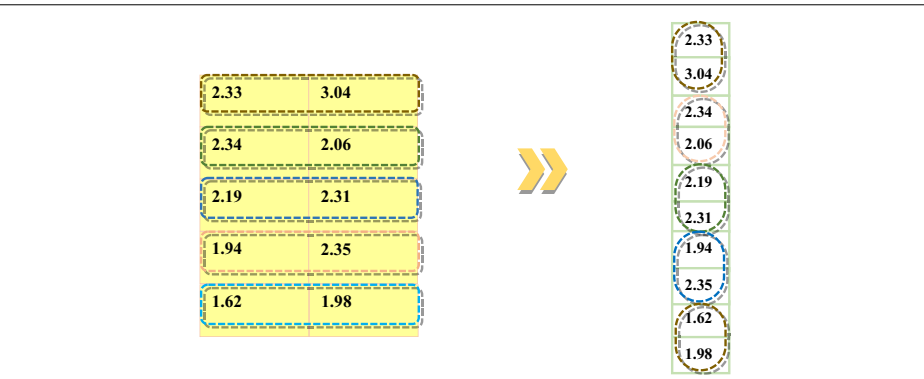

Fig. 7 Flatten layer

Flatten layer

The flatten layer of Convolutional Neural Network transforms the pooled feature map into a column vector, which is made input to the neural network for the classification task [39]. The column vector represents the feature map for the desired sentence. To make the feature vector rows concatenated, the pooled feature map is flattened through reshape function of numpy as shown in an Eq. (14):

$$Flattening = pooled.reshape\,(4 * 2,\ 1) \tag{14}$$

This equation takes row 1, row 2, row 3 and so on, then append them all to make a single column vector. Figure 7 describes the function of flatten layer in which the matrix shows the pooled feature map for a sentence and the single column vector depicts that flattening operation is performed on a sentence pooled feature map.

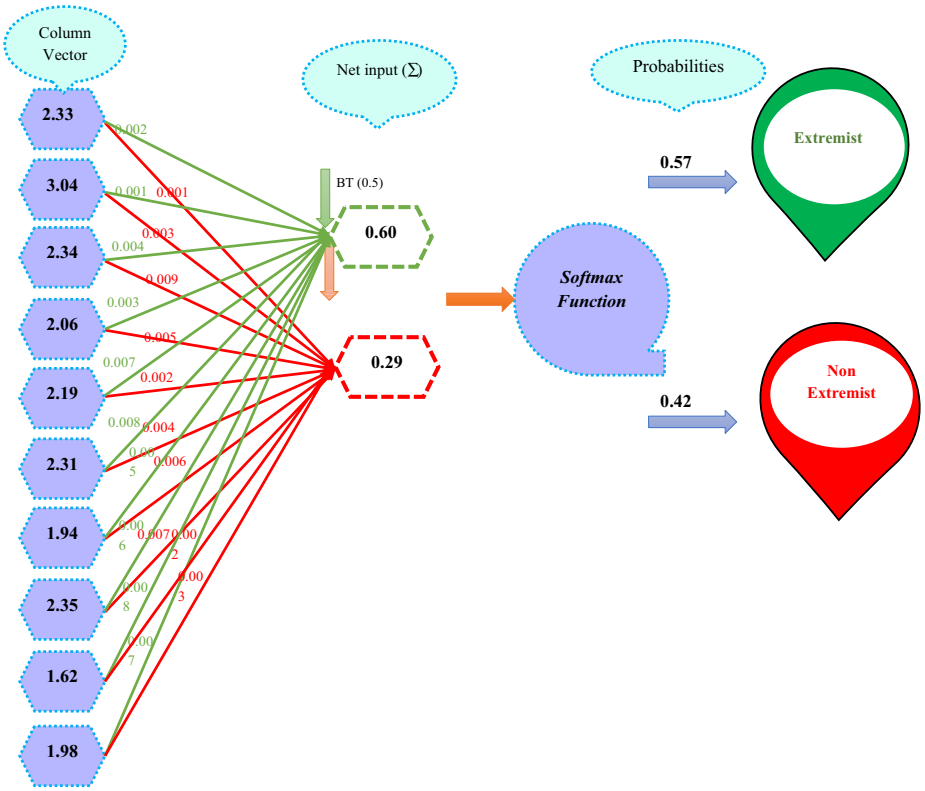

Fig. 8 Applying softmax function for classification

Output layer

At the output layer, an activation function like sigmoid, tanh, or softmax, is applied to compute the probability for the two classes, i.e. extremist and non-extremist. For example, the desired input text "*Baghdadi… our last and only hope, I simply love you*", is tagged as "*extremist*", when passed through the proposed CNN model.

In Fig. 8 the classification of the input vector is performed using the softmax function. The net input is obtained by applying Eq. (15).

$$u_j = \sum_{i}^{l} w_i x_i + b \tag{15}$$

where "*w*" represents weight vector, "*x*" represents an input vector and "*b*" is a bias term.

Softmax layer

Following are additional functions used in LSTM + CNN model, as shown in Algorithm 1.

Compile function The compile method is used for model configuration. It covers different parameters: (i) *Loss*, it is an objective function, (ii) *Optimizer,* an instance/name of an optimizer which is used for the model compilation, and (iii) *Metrics,* it holds the evaluation metrics.

Summary of the model The summary of the model will be shown using *summary* function after model creation.

Fitting a model This section of the pseudo-code involves training of the required dataset and after that, evaluation of the model is performed on the test dataset. Finally, accuracy will be the output of the model.

Algorithm 1. Pseudocode for Extremist Tweet Classification Using LSTM+CNN Model

Input: Training Intent Dataset trR*X*, Training Intention Values trI*Y,* Testing Intent Dataset teR*X*, Testing Intention Values teI*Y*

Output: Accuracy(totalAcc)

1. **ProcedureCNN MODEL (trRX, trIY)**

2. batchSize = 16; epoch = 1; filters = 4; pool_size = 2, verbose = 2, NEpoch=7
3. max_features = 2000, embed_dim = 128, classes=2, input_length=3382
4. model = Sequential()

 # Embedding Layer
5. model.add (Embedding (max_features, embed_dim, input_length))

 # Dropout Layer
6. model.add(Dropout (0.5))

 # LSTM Layer
7. model.add(LSTM (100))

 # Convolutional Layer
8. model.add (Conv1D (filters, kernel_size=8, padding='same', activation='relu')

 # Maxpooling Layer
9. model.add(MaxPooling1D(pool size))

 # Flatten Layer
10. model.add(Flatten())

 # Softmax Layer
11. model.add(Dense (classes, activation='softmax'))

Compile Function
12. model.compile (loss = binary_crossentropy, optimizer = adamax, metrics= [accuracy])

 # Summary of the model
13. print (model.summary ())
14. for all epochs in (1: NEpoch) do
 # Fitting a Model
15. model.fit (X_train, Y_train, epoch, validation_data(X_test,Y_test,batch_size=batchsize))
#Model Evaluation
16. model.evaluate (X_test, Y_test, verbose, batch_size = batchsize)
17. totalAcc.append(validationAcc)
18. End for
19. return totalAcc
20. End Procedure

Table 5 Extremist-related emotion classification

Text id	Input text	Emotion class
2	Oh Allah, destroy US and Israel	Anger
3	Great news, ISIS fight Afghan forces to capture Helmand.	Joy
3	Love you Baghdadi, Who is interested to know the truth about ISIS Yes.. visit our page to listen # Montada	Joy
4	A very painful fight……clean up operation gave us a big blow #ISIS	Fear
5	A suicide bomber destroyed the police station and killed 20 people	Sadness
6	OMG…Hashish trade has provided us strong footing in achieving the 2014 fund raising target for the recurring expenditure of Khilafat #ISIS	Joy
7	Kafir governments do not allow us to transfer funds to mujahid accounts… instead sell enormously amount of heroin and opium to needy persons, earn in bitcoins and help us by donations…(Al-Firdaws)	Analytical
8	Baghdadi… our last hope, I simply love you #ISIS	Joy
9	#ISIS. today heard a sad news…our commander in Iraq has been shot dead in counter-attack…	Sadness

A sample implementation code is given in Additional file 1: Appendix A.

Analyzing sentiments of users w.r.t emotional affiliation with extremists

This module deals with emotion classification of the user's sentiments showing affiliation with the extremists' postings. Each of the input text is tagged with an emotion category using the Python-based tone analyzer API [11]. It returns an emotion set: *{anger, sadness, fear, joy, confident, analytical and tentative}*. In this module, an input text is analyzed and the corresponding emotion class is identified. For example, when the input text: "*Great news, ISIS fight Afghan forces to capture Helmand..*" is passed through the emotion analyzer, it returns an emotion class "*joy*". A sample set of user reviews and the detected emotion class is shown in Table 5.

Experimental setup

To conduct the experiments, we used the Python and Anaconda framework (https://anaconda.org/anaconda/python).

Results and discussion

In this section, we discuss results obtained by conducting different experiments to answer the posed research questions.

Answer to RQ#1: How to recognize and classify tweets as extremist vs non-extremist, by applying deep learning-based sentiment analysis techniques?

To answer this research question, we applied deep-learning-based sentiment analysis technique, namely LSTM + CNN (discussed in detail in the proposed methods section). Additionally, we conducted experiments on 1-Layer CNN, 1-Layer LSTM. Table 6 shows results obtained on account of applying different DL models and it is obvious that the proposed LSTM + CNN model achieves best results.

The LSTM + CNN model attained best results, as the LSTM layer generates a new representation of the input tweet received from the embedding layer by capturing information

Table 6 Experimental results of different DL-based SA models

Models	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)
1-layer LSTM	85.07	87	85	85
1-layer CNN	83.09	84.4	82	82
LSTM + CNN	92.66	88.32	89.47	90.71

Table 7 Parameter setting regarding proposed LSTM + CNN model

LSTM + CNN model layers	Parameters with values
Convolutional layer	Number of filters = 2, 4, 6, 8, 10, 12, 14, 16
	Kernel size = 2 × 2
	Padding = same
Pooling layer	Pooling size = 2 × 2
LSTM layer	Units = 100
Additional	Vocabulary size = 2000 Input vector size = 50 Embedding dimension = 128 Batch size = 8 Number of epochs = 7

Table 8 All variants of LSTM + CNN with parameter setting

Model	LSTM units	Pooling size	Kernel size	Number of filters
LSTM + CNN1	100	2 × 2	2 × 2	2
LSTM + CNN2	100	2 × 2	2 × 2	4
LSTM + CNN3	100	2 × 2	2 × 2	6
LSTM + CNN4	100	2 × 2	2 × 2	8
LSTM + CNN5	100	2 × 2	2 × 2	10
LSTM + CNN6	100	2 × 2	2 × 2	12
LSTM + CNN7	100	2 × 2	2 × 2	14
LSTM + CNN8	100	2 × 2	2 × 2	16

from both the current and previous inputs, reducing the information loss. The LSTM model retains the context information using current and previous states for sufficient duration to make predictions. At the next level, the CNN layer captures additional features (n-gram) from the richer collection, yielding better and improved performance results.

Parameter settings

Table 7 presents the parameter setting for the proposed model (LSTM + CNN). The experiment is conducted using different parameters, listed as follows: the parameter, namely '*number of filter*', receives a value varying from 2 to 16, while the values of other parameters, such as kernel size, padding, pooling size, optimizer, batch size, epochs, and units, are fixed.

The configuration setting regarding 8 LSTM + CNN models with selected parameters (number of filters, kernel size, pooling size, and LSTM units), is shown in Table 8.

We have listed the accuracy, loss score, and training time in Table 9. After conducting experiments with varying parameter setting of LSTM + CNN models, it is noted that the performance of LSTM + CNN8 model is better with lstm units = 100 (cells), pooling

Table 9 LSTM + CNN models training time, loss score and accuracy

Model	Training time (s)	Loss score	Accuracy (%)
LSTM + CNN1	23	1.25	62
LSTM + CNN2	28	1.17	74
LSTM + CNN3	44	1.11	80
LSTM + CNN4	24	0.98	88
LSTM + CNN5	19	0.98	88.12
LSTM + CNN6	48	0.97	90.01
LSTM + CNN7	44	0.99	90.4
LSTM + CNN8	29	0.96	92.66

size = 2×2, and number of filters = 16 and it's achieved accuracy is 92.66%. It is noted that the accuracy of the model increases by increasing the number of filters.

Answer to RQ2: What is the performance of classical feature sets liken-grams, bag-of-words, TF-IDF, bag of-words (BoW) over word embedding learned using CNN, LSTM, FastText, and GRU?

Firstly, we discuss a few baseline/state-of-the-art methods, in all such techniques, a feature vector is created for a given tweet, which is applied as its feature set with the classifier.

Baseline methods

To conduct experiments, we used different classifiers for the following three feature representation techniques in the baselines [7, 8, 12]: (i) *n-gram* It is the state-of-the-art technique [12]) (ii) *bag-of-words* The bag-of-word, also called Count vectorizer technique, makes use of word frequency [7, 13], and (iii) *TF-IDF* A feature vector is created for text classification [8].

Deep learning methods with variants

Different variants of DL techniques (CNN + Random Embedding, LSTM + Random Embedding, FastText + Random Embedding, and GRU + Random Embedding), are used for tweet classification as extremist and non-extremist. The results are reported in Table 10. The proposed LSTM + CNN performs better than the other DL methods.

Proposed method

For the extremist classification task, we experimented different DL-based SA models, namely CNN, LSTM, FastText and GRU, initialized with random word embeddings. The proposed LSTM + CNN model for extremist classification outperforms baseline methods (Part A of Table 10) and it also performs better than the other DL models (Part B of Table 10).

Table 10 Comparison of proposed work with baseline methods

Study	Technique	Results		
		P	R	F
Part-A Baselines	N-gram + SVM [12]	73	78	79
	Bag of words +SVM [13]	73	74	73
	TF-IDF +RF	78	84	90
	BOW + KNN [8]	76	75	76
Part-B Deep learning methods with variants	CNN + word embedding	84	84	84
	LSTM + word embedding	86	86	86
	FastText + word embedding	87	87	87
	GRU + word embedding	85	85	85
Part-C Proposed	LSTM + CNN	0.90	0.88	0.88

Parameter setting for LSTM + CNN

Following parameters are used for the LSTM + CNN model namely: (i) Max features (10 000), (ii) embedding dimension (128), (iii) LSTM unit size (200), convolutional filter size (200), and (iv) a batch size 32 with 3 epochs which yielded best performance results as shown in (Part C of Table 10).

Answer to RQ3#: What is the performance of proposed model for extremist affiliation classification with respect to state-of-the-art methods?

To find an answer for RQ3, we conducted experiments using supervised, lexicon-based, benchmark proposed technique for extremist Classification on Twitter.

Supervised techniques

In Table 11, the results of the proposed method (*LSTM + CNN*), are compared with different state-of-the-art techniques [7, 8] based on machine classifiers, namely, KN-Neighbors, Naïve Bayes Classifier. Furthermore, we also applied other machine and deep learning classifiers, namely Random Forest, Support Vector Machine, LSTM, and CNN. The objective of the experimentation is to perform extremist classification of Tweets using different ML and DL classifiers. The performance evaluation results of various machine learning classifiers are presented in terms of accuracy, precision, recall, and f-measure. The KNN exhibited the lowest performance result (72% accuracy).

Lexicon-based technique

The SentiWordNet is used [17] to classify the reviews as extremist or nob-extremist using sentiment analysis. The reviews tagged as positive are treated as having affiliations with extremist, whereas negative reviews are treated as non-extremist reviews. The experimental results are presented in Table 11.

Proposed

The proposed technique (LSTM + CNN) produced the best results (Table 11), when compared with the other comparing methods.

Table 11 Comparison with state-of-art techniques

Study	Techniques	Features	P (%)	R (%)	F (%)	A (%)
Wei et al. [8]	Machine learning (KNN)	Classical features (tf, idf, tfxidf) BOW	73	71	71	74
Azizan and Aziz [7]	Machine learning (NB)	Classical features (tf, idf, tfxidf)	70	68	69	72
Proposed	Deep learning (LSTM + CNN)	Word embedding	90	86	84	92
Chalothorn and Ellman [17]	Lexicon-based (Senti WordNet)	Sentiment scores and polarity classes	69	68	69	73
Other Models	Deep learning (LSTM)	Word embedding	85	84	83	85
	Deep learning (CNN)	Word embedding	88	84	83	88
	Machine learning (SVM)	Tf xidf	79	78	79	79
	Machine learning (RF)	Tf xidf	83	81	82	84

Table 12 Comparative results of sentiments of users w.r.t emotional affiliation with extremists

Technique/classifier	Accuracy (%)	Precision (%)	Recall (%)	F1 measure (%)
Random forest (RF)	0.82	0.84	0.83	0.83
Support vector machine (SVM)	0.79	0.79	0.78	0.79
K-nearest neighbour (KNN)	0.72	0.72	0.72	0.71
Naïve Bayesian (NB)	0.71	0.70	0.70	0.69
Classification of users sentiments w.r.t emotional affiliation with extremists (proposed)	0.90	0.86	0.84	0.90

Answer to RQ4: How to perform the sentiment classification of user reviews w.r.t emotional affiliations of Extremists on Twitter and Deep Web?

To find an answer for RQ2, we conducted experiments using tone analyzer API [11] for Emotion Classification of User reviews w.r.t Extremist's Affiliation on Dark Web. Results reported in Table 12 show that the proposed module for emotion classification outperforms the comparing supervised machine learning classifiers in terms of improved accuracy, precision, recall, and F-measure.

Conclusions and future work

This study presents a sentiment-based extremist classification system based on users' postings made on Twitter. The proposed work operates in three modules: (i) users' tweet collection, (ii) preprocessing, and (iii) classification with respect to extremist and non-extremist classes using LSTM + CNN model and other ML and DL classifiers.

The experimental results show that the proposed system outperformed the comparing methods in terms of better precision, recall, f-measure, and accuracy. However, the system has certain limitations, such as (i) lack of an automated method for crawling, cleaning and storing Twitter content, (ii) lack of considering visual and social context features for obtaining more robust results, and (iii) investigating other types of extremism by apply the DL methods for multi-class label classification. Our future work aims at

applying more advanced techniques, such as attention-based mechanism for extremist affiliation detection with multi-class labels. Furthermore, the inclusion of context-aware features can also improve the performance of the system.

Additional file

Additional file 1: Appendix A. A sample implementation of LSTM+CNN for extremist classification.

Abbreviations

AI: artificial intelligence; ML: machine learning; DL: deep learning; CNN: Convolutional Neural Network; LSTM: long short-term memory; TF–DF: term frequency and inverse document frequency; BoW: bag of words; KNN: K-nearest neighbors; SVM: Support Vector Machine; NB: Naïve Bayesian; NLTK: Natural Language Toolkit.

Acknowledgements

This article was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah. The authors, therefore, acknowledge with thanks DSR for technical and financial support.

Authors' contributions

Conceptualization: SA, MZA. Data curation: FMA, IA. Formal analysis: SA, FMA. Investigation: MZA, FMA. Methodology: SA, MZA. Project administration: SA. Resources: SA, IA, FMA. Software: MZA, SA. Supervision: SA. Validation: MZA, IA. Visualization: SA, IA, Writing—original draft: MZA. Writing—review and editing: FMA, IA. All authors read and approved the final manuscript.

Funding

The Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, funded this article. The authors, therefore, acknowledge with thanks DSR for technical and financial support.

Availability of data and materials

The research data used to support the findings of this study are available from the corresponding author upon request.

Competing interests

The authors declare that they have no competing interests.

Author details

[1] Faculty of Computing and Information Technology at Rabigh (FCITR), King Abdulaziz University, Jeddah, Saudi Arabia. [2] Institute of Computing and Information Technology, Gomal University, Dera Ismail Khan (KP), Pakistan. [3] Department of Information Systems, Faculty of Computing and Information Technology (FCIT), King Abdulaziz University, Jeddah, Saudi Arabia. [4] Department of Computing, University of Bradford, Bradford, UK.

Received: 10 January 2019 Accepted: 8 June 2019
Published online: 01 July 2019

References

1. Hao F, Park DS, Pei Z (2018) When social computing meets soft opportunities and insights. Human-centric Comput Inform Sci 8(8):1–18
2. Hao F, Li S, Min G, Kim HC, Yau SS, Yang LT (2015) An efficient approach to generating location-sensitive recommendations in Ad hoc social network environments. IEEE Trans Serv Comput. 8(3):520–533
3. Hao F, Min G, Pei Z, Park DS, Yang LT (2017) k-clique communities detection in social networks based on formal concept analysis. IEEE Syst J. 11(1):250–259
4. Iskandar B (2017) Terrorism detection based on sentiment analysis using machine learning. J Eng Appl Sci 12–3:691–698
5. Ferrara E, Wang WQ, Varol O, Flammini A, Galstyan A (2016) Predicting online extremism, content adopters, and interaction reciprocity. International conference on social informatics. Springer, New York, pp 22–39
6. Badjatiya P, Gupta S, Gupta M, Varma V (2017) Deep learning for hate speech detection in tweets. In: Proceedings of the 26th international conference on world wide web companion. International World Wide Web conferences steering committee, pp 759–760
7. Azizan SA, Aziz IA (2017) Terrorism detection based on sentiment analysis using machine learning. J Eng Appl Sci 12(3):691–698
8. Wei Y, Singh L, Marti S (2016) Identification of extremism on Twitter. Proceedings of the IEEE/ACM international conference on advances in social networks analysis and mining. IEEE, New Jersey, pp 1251–1255
9. Zhang H, Wang J, Zhang J, Zhang X (2017) Ynu-hpcc at semeval 2017 task 4: using a multi-channel cnn-lstm model for sentiment classification. In: Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017) pp 796–801

10. Yenter A, Verma A (2017) Deep CNN-LSTM with combined kernels from multiple branches for IMDB review sentiment analysis. In: 2017 IEEE 8th annual ubiquitous computing, electronics and mobile communication conference (UEMCON). IEEE. pp 540–546
11. IBM Watson tone analyzer. https://www.ibm.com/watson/developercloud/tone-analyzer/api/v3/. Accessed 19 July 2018
12. Sureka A, Agarwal S, Schmidtke F (2014) Learning to classify hate and extremism promoting tweets. 2014 IEEE Joint intelligence and security informatics conference (JISIC). IEEE, New Jersey, p 320
13. Hartung M, Klinger R, Schmidtke F, Vogel L (2017) Identifying right-wing extremism in german Twitter profiles: a classification approach. International conference on applications of natural language to information systems. Springer, Cham, pp 320–325
14. Nguyen A, Hoang Q, Nguyen H, Nguyen D, Tran T (2017) Evaluating marijuana-related tweets on Twitter. IEEE 7th annual computing and communication workshop and conference (CCWC). IEEE, New Jersey, pp 1–7
15. Asghar MZ, Khan A, Ahmad S, Qasim M, Khan IA (2017) Lexicon-enhanced sentiment analysis framework using rule-based classification scheme. PLoS ONE 12(2):e0171649
16. Ryan S, Garth D, Richard F (2018) Searching for signs of extremism on the web: an introduction to sentiment-based identification of radical authors. Behav Sci Terror Pol Aggres 10:39–59. https://doi.org/10.1080/19434472.2016.1276612
17. Chalothorn T, Ellman J (2012) Using SentiWordNet and sentiment analysis for detecting radical content on web forums
18. Bermingham A, Conway M, McInerney L, O'Hare N, Smeaton AF (2009) Combining social network analysis and sentiment analysis to explore the potential for online radicalisation. In: IEEE international conference on advances in social network analysis and mining, ASONAM'09. pp 231–236
19. Asghar MZ, Khan A, Zahra SR, Ahmad S, Kundi FM (2017) Aspect-based opinion mining framework using heuristic patterns. Cluster Comput pp 1–19
20. Asghar MZ, Rahman F, Kundi FM, Ahmad S (2019) Development of stock market trend prediction system using multiple regression. In: Computational and mathematical organization theory. pp 1–31
21. Skillicorn D (2011) Computational approaches to suspicion in adversarial settings. Inform Syst Front. https://doi.org/10.1007/s10796-010-9279-4
22. Cheong M, Lee VC (2011) A microblogging-based approach to terrorism informatics: exploration and chronicling civilian sentiment and response to terrorism events via Twitter. Inform Syst Front 13–1:45–59
23. Asghar MZ, Kundi FM, Ahmad S, Khan A, Khan F (2018) T-SAF: Twitter sentiment analysis framework using a hybrid classification scheme. Expert Syst 35(1):e12233
24. Zeng D, Wei D, Chau M, Wang F (2011) Domain-specific Chinese word segmentation using suffix tree and mutual information. Inform Syst Front. https://doi.org/10.1007/s10796-010-9278-5
25. Prentice S, Taylor P, Rayson P, Hoskins A, O'Loughlin B (2011) Analyzing the semantic content and persuasive composition of extremist media: a case study of texts produced during the Gaza conflict. Inform Syst Front. https://doi.org/10.1007/s10796-010-9272-y
26. Consuming streaming data. https://developer.twiiter.com/en/docs/tutoroals/consuming-streaming-data.html. Accessed 19 July 2018
27. Zhang Y, Zeng S, Fan L, Dang Y, Larson CA, Chen H (2009) Dark web forums portal: searching and analyzing jihadist forums. In: IEEE international conference on intelligence and security informatics, ISI'09. pp. 71–76
28. BiSAL-A Bilingual sentiment analysis lexicon to analyze dark web forums for cyber security
29. Omer E (2015) Using machine learning to identify jihadist messages on Twitter
30. Asghar MZ, Khan A, Khan F, Kundi FM (2018) RIFT: a rule induction framework for Twitter sentiment analysis. Arabian J Sci Eng 43–2:857–877
31. Erickson BJ, Korfiatis P, Akkus Z, Kline T, Philbrick K (2017) Toolkits and libraries for deep learning. J Digit Imaging 30–4:400–405
32. Chomba B (2018) What is the difference between a training set and a test set? https://www.quora.com/What-is-the-difference-between-a-training-set-and-a-test-set. Accessed 10 Dec 2018
33. Acharya A (2017) Comparative study of machine learning algorithms for heart disease prediction
34. Jason B (2018) Evaluate the performance of deep learning models in Keras. https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/#comment-460892. Accessed 14 Oct 2018
35. What's is the difference between train, validation and test set, in neural networks? https://stackoverflow.com/questions/2976452/whats-is-the-difference-between-train-validation-and-test-set-in-neural-netwo. Accessed 2 Dec 2018
36. A Gentle introduction to dropout for regularizing deep neural networks. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. Accessed 4 Nov 2018
37. Understanding LSTM cells using C#. https://msdn.microsoft.com/en-us/magazine/mt846470.aspx. Accessed 16 Oct 2018
38. A numerical example of LSTMs. https://statisticalinterference.wordpress.com/2017/06/01/lstms-in-even-more-excruciating-detail/. Accessed 05 Oct 2018
39. Convolutional Neural Networks (CNN): Step 3—flattening. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening. Accessed 10 Dec 2018

Publisher's Note