Neural Processing Letters (2024) 56:42
https://doi.org/10.1007/s11063-024-11500-8

A Hybrid Model Based on Convolutional Neural Network and Long Short-Term Memory for Multi-label Text Classification

Hamed Khataei Maragheh[1] · Farhad Soleimanian Gharehchopogh[1] · Kambiz Majidzadeh[1] · Amin Babazadeh Sangar[1]

Accepted: 30 October 2023 / Published online: 16 February 2024

Abstract
Multi-label text classification (MLTC) is a popular method for organizing electronic documents, which is crucial for accessing and processing data. As the number of classes increases, learning multi-label data will be challenging. The number of possible states for various labels increases exponentially, and learning algorithms in single-label data cannot be used to solve these problems. In the meantime, using single-label data algorithms could be very time-consuming. In MLTC, complexity costs should be reduced. Deep-learning neural networks that can learn intricate patterns are used in many real-world problems because of their high power and accuracy. This paper proposed a hybridization of the long short-term memory (LSTM) neural network and the convolutional neural network (CNN) method for MLTC. The proposed model uses LSTM to enhance CNN to improve the proposed model's accuracy. Also, the competitive search algorithm (CSA) is used to improve the LSTM hyperparameters. The LSTM hyperparameters play an important role in increasing the detection accuracy. The CSA algorithm finds the best values for the hyperparameters by searching the problem space. It was tested on four different datasets of multi-label texts: Reuters-21578, RCV1-v2, EUR-Lex, and Bookmarks. The result showed that the proposed model performed better than CNN and LSTM-CSA in terms of accuracy percentage and that it has improved by an average of more than 10%. Also, the results show that the LSTM-CSA model has higher detection accuracy compared to LSTM—Gradient-based optimizer (GBO) and LSTM—whale optimization algorithm (WOA).

Keywords Multi-label text classification · Long short-term memory · Convolutional neural network · Competitive search algorithm

✉ Farhad Soleimanian Gharehchopogh
bonab.farhad@gmail.com

[1] Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran

1 Introduction

A considerable rise in electronic text documents has resulted from the rapid expansion of web pages, social networks, and online storage spaces [1]. Organizing text documents for users to access the desired content quickly is a necessity. Due to the rapid expansion of the web, creating an automatic method to classify texts will increase classification accuracy and efficiency [2]. Text classification means the automatic assignment of predetermined categories to natural language texts based on their content [3], assuming that the set $D = \{d_1, d_2, \ldots, d_m\}$ has a training sample and sets of classes is defined as $C = \{c_1, c_2, \ldots, c_k\}$ with the training sample involving a class label [4]. After then, the training data is utilized to build a classification model. A test sample label with an unspecified class is predicted [5].

Depending on how many labels are applied to each sample, there are two classification issues: Single-Label Classification (SLC) problems and Multi-Label Classification (MLC) problems. In SLC problems, each data has a unique label. Most studies on machine learning are related to Single-Label Text Classification (SLTC) [6, 7]. However, many texts require Multi-Label Classification (MLC) to be solved. In SLC, each sample is related to a class, which specifies the sample's characteristics [8]. In MLC, each sample may pertain to several classes, and all of these classes determine the features of the samples [9]. In other words, in MLC, each instance is specified by sampling the classes. In many real-world applications, data consist of multiple classes.

MLTC is a more general state of single-class text classifications, with each instance belonging to a set of classes. As for the MLTC problem and SLTCs, there is a set of training data and labels, with each sample represented by a vector of the features [10, 11]. A classifier is trained on training data to predict experimental data labels [12, 13]. In the MLTC, it is assumed that the sample sets belong to at least one class. Each sample can involve any number of labels defined from the label set [14].

If X is a sample of the primary dataset and the sample includes features d and Y a set of possible labels of size q, this set of labels will be represented as $\{Y_1, Y_2, \ldots, Y_q\}$. The S dataset is then a multi-label dataset defined as $\{(x_i \cdot Y_i)|1 \leq i \leq l\}$, where x_i is a training sample and Y_i is a subset of possible labels [15]. Finally, a multi-label classifier for each experimental sample predicts one of the 2^q states of the possible label subset. For instance, it is assumed that a training set has five labels $M = \{Y_1, Y_2, Y_3, Y_4, Y_5\}$ and a training sample such as x is considered; If sample x in the dataset has labels $\{Y_1, Y_4\}$ then the label set for this sample is divided into two sets; a set of related labels that are the same as $\{Y_1, Y_4\}$ and a set of unrelated labels that include the remaining labels, i.e., $\{Y_2, Y_3, Y_5\}$ [16].

A subset of machine learning algorithms is known as deep learning to discover and extract functional patterns from primary datasets by multiple layers [17]. A deep graph is used to model this process, with both linear and non-linear transformation layers spread throughout multiple processing layers. Deep learning uses artificial neural network (ANN) architecture in that deep learning models are often recognized as deep ANNs. Deep learning is the same learning by ANNs with many latent layers. In deep learning, each latent layer is responsible for training a unique feature set based on the previous layer's output. As the number of latent layers is added, the temporal complexity increases. This type of hierarchical learning hybrids low-level features with high-level features, making it easier to identify essential features.

This paper uses CNN [18] and LSTM neural network [19] for MLTC. Hybridization of the two models aims to increase the accuracy percentage and reduce the prediction error. In this paper, the competitive search algorithm (CSA) [20] is used to optimize LSTM hyperparameters. The CSA is a crowd-based meta-heuristic algorithm that has been used to solve complex

optimization problems. The CSA is proposed based on some social activities in human life, such as all-around sports competitions and talent shows. The CSA has obvious advantages in search accuracy, convergence speed, and stability. The CSA is used to optimize three important hyperparameters of the LSTM, which are the number of hidden neurons, dropout rate, and learning rate. The following mainly contributed to the paper:

- Increasing the accuracy of MLTC using hybridization of CNN and LSTM
- Comparing the proposed model with other models
- Improving CNN using LSTM
- Optimizing LSTM hyperparameters by CSA. CSA algorithm discovers optimal values for LSTM.
- Evaluating the proposed model on different datasets of multi-label texts

This paper's structure is generally organized: Sect. 2 explains previous studies. Section 3 defines the CSA algorithm. In Sect. 4, the proposed model's steps are based on the hybridization of CNN and LSTM-CSA. The proposed model's performance is assessed in Sect. 5 along with a comparison to other models. Section 6 concludes by detailing the recommendations for future work.

2 Related Works

So far, much progress has been made in SLTC. However, MLTC is one of the subjects focused on attention in recent years. Multi-label data is a more general form of single-class data. In MLTC, the goal is to classify samples with more than one label. The algorithms that are being considered for use in this area need to have the capability of predicting numerous labels for a new sample. The Hierarchical Multi-Label Arabic Text Classification (HMATC) model has been proposed using a hierarchical method to MLTC [21]. In MLTC, multiple labels are assigned to each text document simultaneously. To determine the importance of each word, a technique known as the Term Frequency-Inverse Document Frequency (TF-IDF) method is utilized.

The more a word is repeated in a text (TF) but less in other texts (IDF), the greater the TF-IDF value, and this can be an excellent criterion for recognizing the weight of a word in a sentence. It shows how unique and essential a word can be. In the pre-processing stage, static words are removed, words within the text are fragmented, and etymology is done. Keywords are a set of essential words in a document that describe the content of a document used to extract keywords using the TF-IDF method, while the Chi-square method is also used to select important features. The dataset used includes 26,470 samples labeled documents with 11,000 features. The total number of labels is 578. Naïve Bayes multi-label algorithms, Support Vector Machine (SVM), and J48 decision tree were used.

A Deep Neural Network (DNN) model is introduced for MLTC [22]. The model is made up of three key modules: a text coding embedding module, a deep learning feature extraction module, and a universal classification module. By transforming each word into a numerical vector, the embedded module produces a good illustration of the supplied text. Bag of Words is a model in natural language processing used to develop a numerical vector. The principal idea behind it is to assign each word a unique number, and the feature will be obtained based on the frequency of each word repeating. The concept of Part of Speech (POS) is also used to define a word (grammatical points such as nouns, verbs, and adjectives) and the content area in which the word appears. Equation (1) has been used to train the network to train the ANNs in the deep learning section. In Eq. (1), the parameters y_l and x_l are predictions and

objectives for each label $l \in L$, respectively.

$$loss(x \cdot y) = -\sum_{l \in L}\left[\left(y_l \cdot log\frac{1}{1+\exp(-x_l)}\right) + \left((1-y_l)\cdot log\frac{\exp(-x_l)}{1+\exp(x_l)}\right)\right] \quad (1)$$

Twenty-seven thousand seven hundred fifty-five documents performed evaluation and simulation on the PubMed dataset. The average word count per document is 209. The maximum number of words chosen for the DNN was 400. A CNN-based model was proposed for MLTC [23]. The model consists of two stages of CNN and LSTM-based coding and decoding methods. The n-gram method sequences the texts' words in the CNN-based coding step. N-gram predicts a sequence of words. During the decoding process, a recursive LSTM neural network is utilized to make predictions regarding the labels of the text documents.

Simulation and evaluation were performed on three different datasets, i.e., (RCV1-V2, AAPD, and Ren-CECPS). Hamming Loss and Micro-F1 criteria were used for evaluation. The HL criterion calculates erroneous samples to detect predicted unrelated labels. The Micro-F1 criterion is a weighted average of accuracy and recalls criteria calculated as the total number of false positives, false negatives, and real positives. The CNN performed the best and worst performance in accuracy and called on three datasets.

In [24], a DNN architecture is proposed for MLTC problems based on Feature Selection (FS). Pearson correlation was used for selecting the feature in the pooling layer. The simulations that were run made use of fifteen MLC datasets taken from the RUMDR dataset. In these examples, the range of the number of samples is from 207 to 269,648, the range of the number of features is from 72 to 2150, and the range of the number of labels is from 4 to 400. The results demonstrated that the DNN model had a more significant percentage of accuracy in class recognition.

In [25], an MLTC method based on dynamic semantic representation and DNN has been proposed. The Dynamic Semantic Representation Model and DNN (DSRM-DNN) use the embedded word and clustered algorithms to select semantic words. The chosen words become DSRM-DNN elements with weights. Hybridizing the deep belief network with the post-dissemination neural network creates a text classifier. Low-frequency words and semantic terms are specified during categorization. Word bag extracts characteristics in pre-processing. DSRM-DNN is tested using Reuters-21578, RCV1-V2, EUR-Lex, and Bookmarks. Because DSRM-DNN includes more representative words and the dynamic semantic approach adds less, the suggested technique performs poorly in RCV1-V2. The EUR-Lex technique worked better.

LSTM network model for text recognition and a CNN-based VGG-16 for image recognition was proposed. The LSTM is best characterized by learning long-term dependency that is not possible by a recursive neural network (RNN). To accurately predict the next time step, it is necessary to update the weights in the network. This, in turn, necessitates the storage of the information from the earlier time steps. An RNN can only learn a limited number of short-term dependencies. However, an RNN cannot learn long-term time series such as 1000-time steps, while LSTM can learn these long-term dependencies correctly. The TF-IDF method was used to weigh the words. VGG-16 is a deep CNN architecture that uses 1000 classes to categorize ImageNet datasets. 13 convolutional layers and 3 fully linked layers made up this network. The input photos are $224 \times 224 \times 3$ and the filters are 3×3. The model is tested on seven datasets: Hurricane Maria, Hurricane Harvey, Hurricane Irma, the Iran-Iraq earthquake, the Mexico earthquake, Sri Lanka food, and California wildfires [26].

A deep RNN model was tested on two datasets of IMDB and Hotel Reviews [27]. A specific loop structure with memory units retains input information or latent layer states in

the deep RNN. A deep RNN can train consecutive data because the outputs depend on the previous inputs. The IMDB database comprises 50,000 documents and ten classes, with the Hotel Reviews containing 14,895 documents and five classes. The Down-Sampling operation was used to balance the data in the pooling layer. GRNN outperformed CNN, LSTM, and CNN-LSTM in accuracy.

Two updated CNN and LSTM models were tested and implemented on six different datasets to classify multi-label and single-label texts [28]. Initially, CNN layers were adjusted to select the features and FS. N-gram recognized the features in the convolutional layer at different input positions using different convolutional filters. Classification results suggested that CNN's recognition rate was above 90% as it performed better than LSTM.

A CNN-based model with seven different classification methods was tested on six different datasets [29]. This architecture creates document vectors with different words, and t-filters are then applied to these vectors in a convolutional layer to generate t-feature maps. A fully linked layer with $softmax$ output recognizes labels. CNN's recognition accuracy was found to be above 90%.

A recursive convolution neural network model was proposed to increase recognition accuracy and reduce computations [30]. Weight vectors were produced based on the TF-IDF method with a length of 1000. The number of filters to reduce the size of the data was 128. The Reuters-21578 and RCV1-V2 datasets were evaluated. The evaluation showed that the accuracy of the hybrid model was more significant than other models.

A CNN and TF-IDF-based model were proposed for multi-label and SLTC [31]. The words' weight was first injected into the network in the CNN-based architecture, and sentence vectors were generated. The weighting operation was performed on the vectors, and then the filter operation was performed to select the features. Evaluation of five different datasets revealed that the CNN architecture had better recognition accuracy than other models.

A new approach to online, distinguishing linear and non-linear handwritten words was proposed in Devanagari and Bengali texts based on two extended RNN, LSTM and Bidirectional LSTM models [32]. Most word recognition systems use the word labeling approach for both scripts, while the BLSTM system uses the primary word labeling approach based on word movement. A comprehensive dataset experiment was performed to evaluate the performance of the BLSTM model using RNN and HMM. Experimental results suggested that the RNN-based system's accuracy with HMM in Devanagari and Bengali scripts was 99.50% and 95.24%, respectively, as it performed better than the HMM-based system.

MLTC proposes History-based Label Attention (HLA) and History-based Context Attention (HCA) [33]. HCA analyzes context word weight patterns to forecast labels and avoids labeling traps. HLA weights past labels based on a hidden state and combines them to forecast labels. HLA has two benefits: first, it explores label connections to find new labels to improve memory; second, it mitigates the effect of a wrong label in history by influencing other accurate labels. HCA + HLA outperformed HCA, HLA, and Seq2seq.

Label Embedding and the embedded module produce a good illustration of the supplied text by transforming each word into a numerical vector developed to overcome the problem of MLTC [34]. The LELC model examines label information and correlation using the co-occurring label matrix and label correlation matrix. BI-GRU extracts fundamental features and a multi-layer attention framework selects label-relevant valid features. Second, the label correlation matrix, which is necessary for multi-label learning, is examined during this LSDR procedure. LELC's efficacy was shown by experimental findings on real-world datasets. Table 1 compares the proposed MLTC-based DNN models.

In Table 1, the proposed models were compared based on DNN. RNN and LSTM networks were found to be more widely applicable than DNN. These networks had a more remarkable

Table 1 Comparison of proposed models for MLTC based on DNN

References	Models	Type	Datasets	Metrics	Advantages
[21]	Multi-label Arabic text classification	CNN	Arabic text	*Hamming loss *Accuracy *Recall *Precision *F-measure	*Low computational cost *Fewer training instances *High predictive performance
[22]	DNN for hierarchical extreme MLTC	CNN	PubMed	*Accuracy *Recall *Precision *F-measure	*Semantic indexing *MLTC issue with an ample label space hierarchically
[23]	MLTC via CNN and Initialized Fully Connection	CNN-LSTM	RCV1-v2, AAPD, Ren-CECPS	*Recall *Precision *F-measure	*Increase in detection accuracy
[24]	DNN to extract high-level features and labels in MLTC problems	CNN	15 MLC datasets from the RUMDR repository	*Accuracy *Performance	*Extracting high-level features and labels on datasets
[25]	An MLTC method via dynamic semantic	CNN	RCV1-v2, Reuters-21578, EUR-Lex, and Bookmarks	*F-measure	*Hybridization deep belief network and back-propagation neural network *Hybridization word embedding model and clustering algorithm to select semantic words
[26]	Classification of multi-label images based on deep ANN	CNN-VGG-16 CNN-LSTM	Seven different disaster-related datasets	*Recall *Precision *F-measure	LSTM and CNN are used to extract essential features from text and pictures, respectively

Table 1 (continued)

References	Models	Type	Datasets	Metrics	Advantages
[27]	Multi-label review rating classification using deep RNN	GRU	IMDB and hotel reviews	*Recall *Precision *F-measure	*Down-sampling method aims to solve the class imbalance issue *Domain Specific Word Embeddings with Gated Recurrent Neural Networks (DSWE-GRNN)
[28]	Investigating the transferring capability of capsule networks for text classification	CNN-LSTM	Six different datasets	*Accuracy	*Fast training *Increase in detection accuracy
[29]	Deep learning for extreme MLTC	CNN	Wiki-30 K, Wiki-500 K, Amazon-12 K, RCV1, EUR-Lex, and Amazon-670 K	*Recall *Precision *F-measure	*Increase efficiency* Bow-CNN is relatively fast in training
[30]	Ensemble application of convolutional and RNN	CNN-RNN	Reuters-21578, RCV1-v2	*Recall *Precision *F-measure	*Reduce complexity *Increase classification accuracy *Use of large datasets
[31]	Classify multi-label and single-class texts	CNN	Yelp Review Full dataset, Yelp Review 10,000 dataset, AGNews dataset, and Yahoo! Answers Topic dataset	*Recall *Precision *F-measure	*Good classification performance *Less computational power and memory

Table 1 (continued)

References	Models	Type	Datasets	Metrics	Advantages
[32]	Comparative productiveness of CNN and RNN architectures for radiology text report classification	RNN, LSTM, BLSTM	Stanford dataset (2512 reports), Stanford (1000 reports), Duke (1000 reports), Colorado Children (1000 reports), and University of Pittsburg medical center (858 reports)	*Accuracy *Recall *Precision *F-measure	*High classification performance
[33]	History-based attention in the Seq2Seq method for MLTC	LSTM	Arxiv Academic, RCV1-V2	*Recall *Precision *Micro-F1*Hamming *Loss	*Performance increase *Weight updating
[34]	MLTC via joint learning from label embedding and label correlation	Bi-GRU	Movielens, IMDB, RCV1-V2-6 K, TJ AAPD, Slashdot, RCV1-V2, Enron, Medical, Ohsumed, and AAPD-3 K	*Hamming loss *Micro-F1	*Better performance *Label co-occurrence matrix

ability to recognize and be accurate. Based on studies on DNN, it is concluded that there are still shortcomings, such as feature extraction, pooling number, and recognition accuracy in the DNN structures. If the structure of a DNN is well designed and the number of layers and training functions are well injected into the network, more accuracy can be obtained [35]. So, our goal in this paper is to improve the structure of the DNN and reduce the shortcomings of CNN.

3 Competitive Search Algorithm

The CSA [20] is a new intelligent optimization algorithm with a simple structure, better optimization, and stronger robustness, which was invented based on social activities in human life. The initial population consists of n factors that are produced according to Eq. (2).

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{bmatrix} \tag{2}$$

where d represents the dimensions (number of variables) of the optimization problem. The fitness value of the factors is calculated according to Eq. (3).

$$F(x) = \begin{bmatrix} f\left(\left[x_{1,1}, x_{1,2}, \cdots x_{1,d}\right]\right) \\ f\left(\left[x_{2,1}, x_{2,2}, \cdots x_{2,d}\right]\right) \\ \vdots \\ \vdots \\ f\left(\left[x_{n,1}, x_{n,2}, \cdots x_{n,d}\right]\right) \end{bmatrix} \tag{3}$$

where n represents the number of agents. The value of each line represents the fit obtained by each factor.

In the CSA, each factor is evaluated and their fitness value is ranked after each round of competition. According to the ranking of the fit value, all factors are divided into two groups: excellent and general. Factors are grouped by fit. It is assumed that 60% of the factors are in the excellent group and the rest are general. The agents of the excellent team with stronger learning abilities and superior ratings are updated according to Eq. (4) ($A(i) > L_1$). Also, the agents of the excellent team with weaker learning abilities and higher ranks are updated according to Eq. (4) ($A(i) \leq L_1$).

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t} + A(i) \times S_1 \times p \times \left(u_b^j - l_b^j\right); S_1 = (U_B \times \text{rand}(1) + L_B) & A(i) > L_1 \\ X_{i,j}^{t} + A(i) \times S_2 \times p \times \left(u_b^j - l_b^j\right); S_2 = (L_B \times \text{rand}(1)) & A(i) \leq L_1 \end{cases} \tag{4}$$

where S_1 and S_2 are the search range functions of agents with strong learning ability (exploration) and normal learning ability. t current iteration; j is several dimensions. X_{ij} is the jth value of the evaluation index of the ith factor, which is the information position in the jth dimension. u_b^j and l_b^j represent the upper and lower limits of the problem space. ρ is a value for the learning direction of agents randomly set in the range [− 1,0,1]; $A(i)$ is the

learning ability of the current agent; L_1 is the threshold value that determines the strength of the learning ability in the superior group, and L_1 belongs to the (0,1) matrix. $A(i)$ is in [1, n].

The performance of Eq. (4) updating the position of agents is reflected in two groups S_1 and S_2. Agents with normal learning ability (S_2) mainly explore the range (LB-0). Agents with strong learning ability (S_1) mainly search the range (LB-UB). S_1 performs the search range more comprehensively. When $p = -1$ means that the agents learn in the opposite direction, when $p = 1$ means that the agents learn in the positive direction, and when $p = 0$ means that the agents do not learn anything in this round. The updating of factors in the normal group is evaluated according to Eq. (5).

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t} + \alpha \times Q \times D & A(i) > L_1 \\ X_{i,j}^{t} \times L_2 \times F \times A(i); F = P \cdot o & A(i) \leq L_1 \end{cases} \tag{5}$$

F is a negative factor; α is a random number in [1]; Q is a random number in [0,2]; D and L_2 are $d \times 1$ matrices, however, all elements in D are equal to 1, elements in L_2 are randomly assigned to -1 and 1; P is a standard normal distribution with mean 0 and variance 1. O is a positive random factor smaller than 0.5. In addition to their learning ability, the agents learn from the best agent to approach the optimal points. The learning of one of the best agents is defined according to Eq. (6)

$$X_{i,j}^{t+1} = X_{i,j}^{t+1} + \left(G_{best}(X_j^t) - X_{i,j}^{t+1}\right) \times A(i) \quad \text{if } A(i) > L_3 \tag{6}$$

where $G_{best}(X_j^t)$ index value in dimension j is the best factor in iteration period t; L3 is the threshold value in the interval (0,1). Which $G_{best}(X_j^t) - X_{i,j}^{t+1}$ shows the distance between the current and optimal agent. By multiplying $G_{best}(X_j^t) - X_{i,j}^{t+1}$ by the learning ability of A(i), the current agent can be closer to the best agent. According to Eq. (5), some agents cannot enter the next competition for various reasons after each round of the competition and are eliminated. Therefore, the corresponding number of factors is randomly added to keep the number of factors constant, and all evaluation indices and learning abilities are randomly generated. In the iteration process, new inputs are generated in the random selection mechanism and the search is performed around the new inputs until the algorithm leaves the local optimal solution. According to the settings of the competitive search algorithm, the best values for $L_1 = 0.8$ and $L_3 = 0.3$. The flowchart of CSA is shown in Fig. 1.

4 Proposed Model

The CNN-LSTM model generalizes RNN models. The LSTM gateways allow it to decide whether to keep the current memory, unlike a traditional RNN, which refreshes content at each time step. LSTM is an RNN architecture designed to store and retrieve information more efficiently than ordinary RNNs. CNN is a type of neural network known as feed-forward that is an effective way to extract features automatically. CNN's essential advantages are: a) Extraction and identification features are in the body of CNN, allowing CNN to learn the process of optimizing features via raw data and b) Because CNN neurons have poor relations with previous layers, they can be helpful for large datasets. Figure 2 shows the proposed model in its entirety. The hybrid CNN-LSTM model is proposed for feature detection and spatial generalization of CNN and increase efficiency. The stability of the hybrid model is relatively high and useful data are not removed.

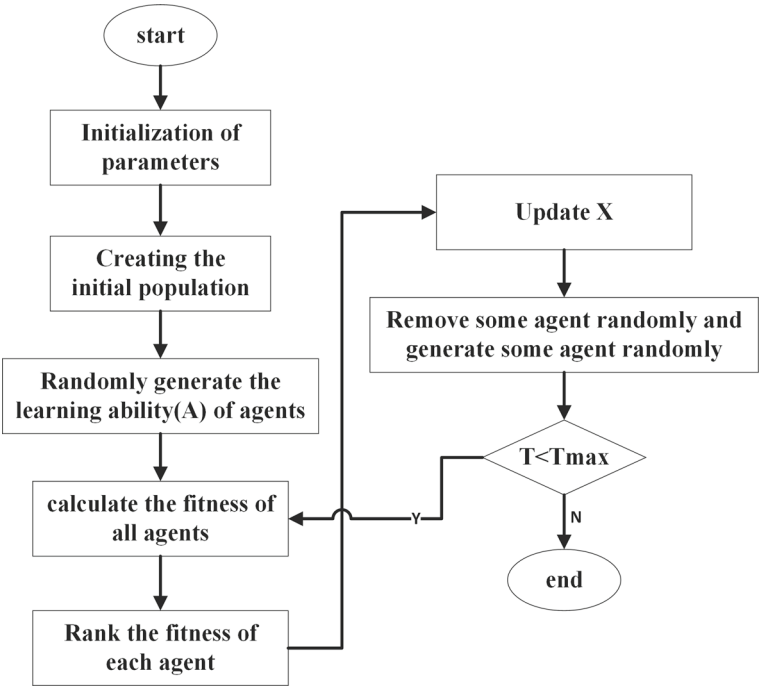

Fig. 1 Flowchart of CSA

The proposed model architecture is depicted in Fig. 3. In the proposed architecture, the goal is to reduce classification error. Therefore, using LSTM, processing operations on data are carried out to increase the accuracy with CNN error generalized. CNN is made up of layers that are convolutional, pooling, and completely linked. In CNN, text documents are first received as inputs, and text documents then enter a complex network of several convolutional and non-linear layers. In each of these layers, operations such as numeric conversion and FS are done. The operation performed by CNN is given to LSTM to select better vectors and increase the accuracy percentage. LSTM hyperparameters are improved by the CSA algorithm. The exact value of the hyperparameters leads to an increase in accuracy. In the LSTM-CSA model, the CSA algorithm automatically determines the type and value of LSTM hyperparameters. As a result, the most optimal model can be obtained within a short period.

The pseudo-code of the proposed model is depicted in Fig. 4.

Pre-processing, feature selection, CNN, and LSTM are the four primary steps of MLTC operations.

4.1 Pre-processing

One of the most important processes in categorizing multi-label texts is pre-processing. The following are the most important procedures to do at the pre-processing text stage:

- Removing numbers from textual data or converting numbers to words
- Removing punctuation, accents, and diagnostic marks
- Removing blanks from textual data

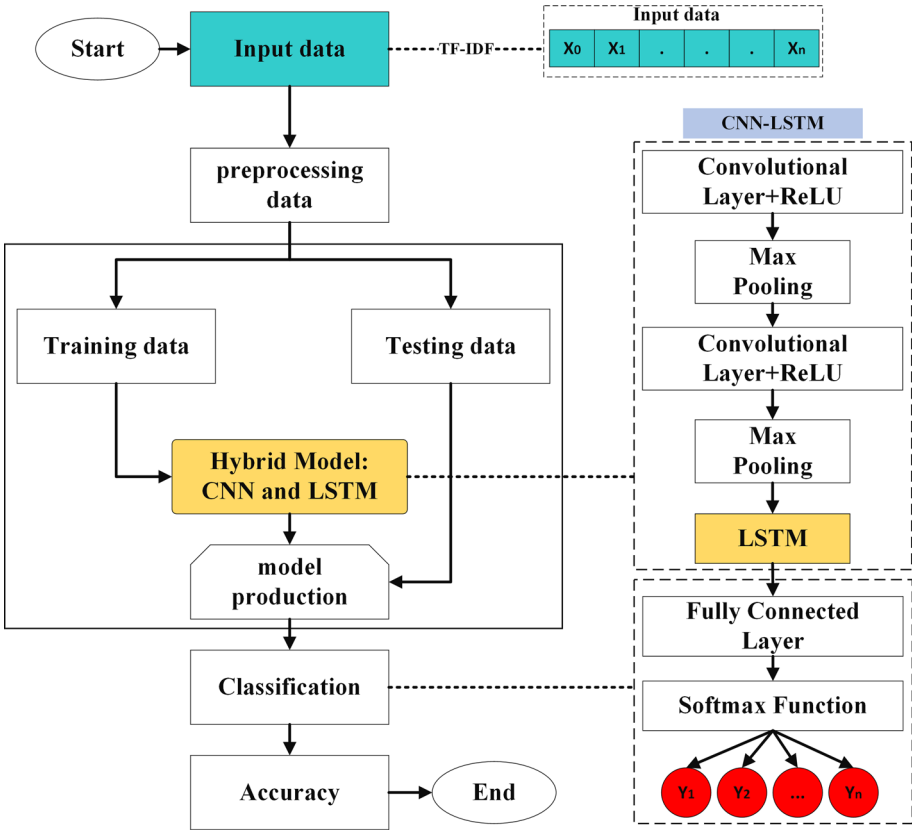

Fig. 2 The overall proposed model

- The roots of the words are discovered at the etymological stage, which is characterized as a basis.
- Creating and expanding abbreviations
- Eliminating neutral and unique terms

4.1.1 Text Data Standardization

Synonymous terms are eliminated from lexical databases and substituted with their more generic meaning.

4.2 Feature Representation

The appropriate structure for representing the texts must first be considered to apply MLTC algorithms. This paper uses the TF-IDF weighting method.

In Eq. (2), parameter (t_i, d_k) represents the number of words t_i in the text document d_k. Parameter D represents the total text documents in the entire dataset and parameter $d(t_i)$ represents the number of text documents in which t_i occurs. Assuming that $x \in R^{L \times V}$, x

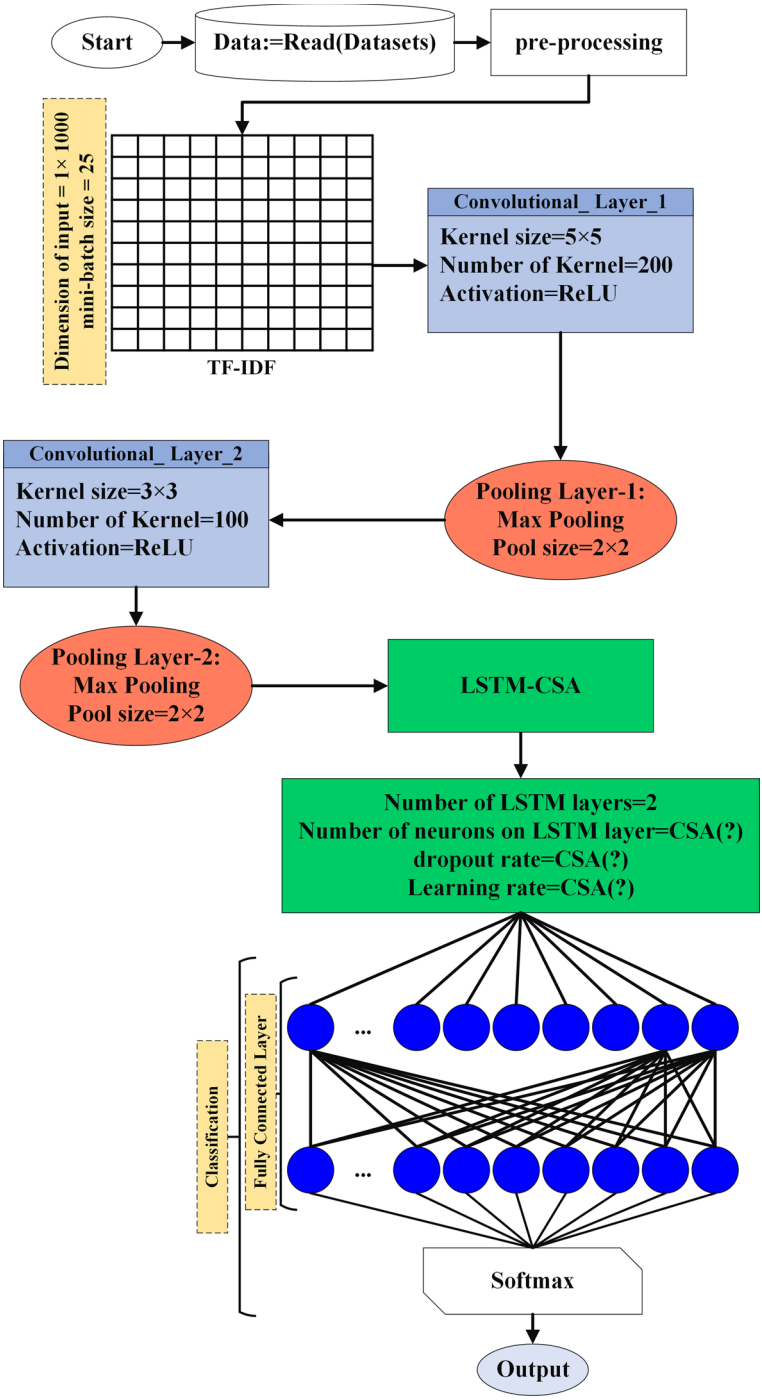

Fig. 3 Depicts the proposed model's architecture

Input: Multi-label text dataset
Output: Classification percentage
1) Read a set of documents
2) Pre-processing operations (Stage 1)
- Remove numeric and empty texts
- Remove punctuation from texts
- Convert words to lowercase
- Remove stop words
- Stemming

3) Display features in weighted space: TF-IDF (Stage 2)
- Term frequency of unprocessed sample text
- Assigning weight to words

4) set of documents of training: building a model for learning
5) set of documents of test: Model testing for performance accuracy
6) CNN (Stage 3)
6-1) batch size×sequence length
6-2) Creating vectors based on word weight: A vector defines the sentence x with n words.
6-3) Creation of convolution layers
- the output of the first convolutional layer⇒ $y_{ij}^{1} = \sigma\left(b_j^1 + \sum_{m=1}^{M} w_{m.j}^{1}\, x_{i+m-1.j}^{0}\right)$
- the output of the l convolutional layer⇒ $y_{ij}^{l} = \sigma\left(b_j^l + \sum_{m=1}^{M} w_{m.j}^{l}\, x_{i+m-1.j}^{0}\right)$

6-3) Creating Pooling layers
7) LSTM-CSA neural network (Stage 4): LSTM-CSA (Optimize the value of hyperparameters for LSTM)
7-1) Improving word vectors
7-2) Increase efficiency and useful features selection
7-3) Layer Activation
8) Fully Connected Layer⇒ $D_i^k = \sum_j w_{ji}^{k-1}\left(\sigma\left(h_i^{k-1}\right) + b_i^{k-1}\right)$
9) softmax function: class detection
10) Percentage of diagnosis
11) end

Fig. 4 Pseudo-code of the proposed model

represents the text's input sentence, L represents the sentence's length, and V the word's size. This $x_i \in R^V$ is a vector of the following V words matching with the input ith word. $W^a \in R^{K_1 \times V}$ represents the filter for a convolutional action so that K_1 is the window size on an input sequence to recognize the features.

High-level features are extracted effectively by the CNN model's convolutional and max-pooling layers. LSTM models can establish stable associations between groups of words. Due to their superior performance, the LSTM model and CNN were used for MLTC. Both LSTM and CNN depend critically on the availability and accuracy of labeled data to achieve their full potential. The more complex the neural network, the more information it needs to train. Furthermore, as word embedding quality evaluates the connection among word vectors in vector space, it substantially impacts the classification outcomes. Consequently, TF-IDF uses word vectors as input data.

4.3 Convolutional Neural Network (CNN)

The CNN layer is made up of three layers: an input layer that receives variables as input, an enhancement layer that extracts features using LSTM, and a final layer that classifies texts. A convolutional layer, an activation function, and a pooling layer are the traditional components of the hidden layer. Local characteristics extracted from high-layer inputs may be sent down to lower layers for more complex features via the CNN layer. In the proposed model, the inputs are given to two convolutional layers. For a hypothetical text such as x, represented as $x = (x_1, x_2, \ldots, x_n)$, $x_i \in R^d$, d represents the size of the word and n the number of words in the text. The dimensions of the primary vectors are 1000, so the mini-batch size for words is

25. Features are extracted from the given inputs, and a ReLU activation function-based non-linear function is added to the convolutional network. The ReLU activation function operates like the sigmoid and tangent functions. The sigmoid and tangent functions are saturated in substantial and minor values, causing these functions' gradients to be zero. In the ReLU activation function, the neurons' weights are updated according to the network structure, with the neurons' upgrading not leading to an increase in error and distance from the optimal value. In other words, the output will be zero if the value of the input variable is less than zero, and it will be x if the value of the input variable is more than zero. Activation makes the network synchronize faster than others. The map of the features created is then transferred to the pooling layer that sampling to decide the active properties using Max Pooling.

Important components of the computation process in CNN are bias value and convolution kernel. The weight value inside the convolution kernel can remain unchanged when the convolution kernel is moved on the feature map by the weight-sharing mechanism of the convolution layer. Equation (7) describes the mathematical expression of the convolution operation process.

$$X_j^N = R\left(\sum_{i \in M_j} X_i^{N-1} \cdot w_{ij}^N + b_{\mathrm{i},\,j}\right) \tag{7}$$

where $R(\cdot)$ is the activation function; M_j is the input feature set; N is the current number of layers; X_i^{N-1} is the input feature map of the N − 1 layer; X_i^N is the output of the N layer; $b_{\mathrm{i},\,j}$ is the bias value; w_{ij}^N is the weight matrix of the N layer.

Equation (8) is the result of the first convolutional layer's vector y output, where y is derived from the previous layer's output vector x, b is the bias for the j feature map, w is the kernel weight, m is the filter index value, and sigma is the activation function similar to ReLU. The result of Eq. (9) is the vector y output of the l convolutional layer.

$$y_{ij}^1 = \sigma\left(b_j^1 + \sum_{m=1}^{M} w_{m\cdot j}^1 x_{i+m-1\cdot j}^0\right) \tag{8}$$

$$y_{ij}^l = \sigma\left(b_j^l + \sum_{m=1}^{M} w_{m\cdot j}^l x_{i+m-1\cdot j}^0\right) \tag{9}$$

Equation (10) shows the max-pooling layer process. This phase determines how much the input data area will be moved, and R is the pooling size smaller than the input y.

$$p_{ij}^1 = \max_{r \in R} y_{i\times T+r.j}^{l-1} \tag{10}$$

The Max Pooling layer is in charge of reducing or selecting features. It does this by decreasing the size of the data, which in turn requires fewer computational resources to process. In Max Pooling, only a some-dimensional feature map from the pooling stage is converted into a one-dimensional feature vector via a fully linked layer. In the suggested architecture, there are two completely linked layers, each with 1024 nodes. A completely linked layer functions similarly to the ANNs layers. The fully connected layer makes the network results represented in a vector of a specified size. This vector can be used for classification.

Sliding filters extract useful information from the convolutional layer. ReLU accelerates convergence and improves model durability in the convolution layer. Max-pooling layers follow convolution layers. To simplify data, the max-pooling layer halves the data amount.

The dropout layer follows the pooling layer to prevent overfitting. During each training period, a random fraction of dropout layer neurons are excluded from weight optimization.

4.4 Long Short-Term Memory (LSTM)

Each LSTM block gets help from a C_t memory at time t. Memory blocks replace latent layer neurons. The output of h_t or LSTM network block activation is $h_t = \Gamma_o \cdot \tanh(C_t)$ where Γ_o is the output gate that controls the rate of data provided through memory. The equation calculates the output gate $\Gamma_o = \sigma\left(W_o \cdot \left[h_{t-1} \cdot X_t\right] + b_o\right)$ where sigma is the sigmoid activation function. W_o is also a weight matrix.

The memory cell C_t is also updated by forgetting the present memory and inserting new memory content in the form of $\overset{'}{C_t}$ based on $C_t = \Gamma_f \cdot C_{t-1} + \Gamma_u \cdot \overset{'}{C_t}$ where the new memory content is calculated through the expression $\overset{'}{C}_t = tanh\left(W_C \cdot \left[h_{t-1} \cdot X_t\right] + b_c\right)$. The forgetting gate controls the extent of current memory to be forgotten Γ_f, and that of new memory content to be added to the memory cell is controlled by the upgrading gate. Figure 5 illustrates the standard LSTM cell diagram.

LTSM intra-network data flow is controlled via three gates. These three gates are:

4.4.1 Update Gate

The update gate, represented by Γ_u, is responsible for controlling the flow of new information. This gate specifies the rate of new information in the current time step.

4.4.2 Forget Gate

The information flow from the prior time step is under the responsibility of forget gate (shown as Γ_f) who is responsible for its management. This gateway demonstrates whether or not the memory information from the preceding step is used, hence setting the entry's data rate based on the information from the preceding step.

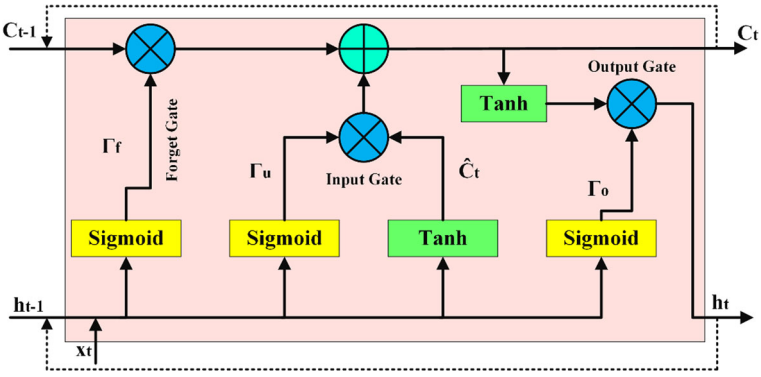

Fig. 5 Standard LSTM cell diagram

4.4.3 Output Gate

The output gate is represented as Γ_o. Taking into account the existing information, this gate decides the degree of information flow from the previous phase to the next.

Variable t is used to represent a cell. Therefore, the cells adjacent to t are positioned in the same layer of $t-1$ and $t+1$. Each cell of the forgetting gate has input and output gates. Update, forget, and output gates are defined based on Eqs. (11) to (16). These gates open and close based on weight matrices $\left(W_u,\ W_f,\ W_o\right)$ and operate based on the sigmoid activation function. Parameters b_c, b_u, b_f and b_o are bias vectors.

Equations	Activation function	Goal	#
$\overset{'}{C}_t = tanh\left(W_C \cdot \left[h_{t-1} \cdot X_t\right] + b_c\right)$	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	Candidate cell state	(11)
$\Gamma_u = \sigma\left(W_u \cdot \left[h_{t-1} \cdot X_t\right] + b_u\right)$	$\sigma(x) = \frac{1}{1+e^{-x}}$	Input gate or update gate	(12)
$\Gamma_f = \sigma\left(W_f \cdot \left[h_{t-1} \cdot X_t\right] + b_f\right)$	$\sigma(x) = \frac{1}{1+e^{-x}}$	Forget gate	(13)
$\Gamma_o = \sigma\left(W_o \cdot \left[h_{t-1} \cdot X_t\right] + b_o\right)$	$\sigma(x) = \frac{1}{1+e^{-x}}$	Output gate	(14)
$C_t = \Gamma_u \times \overset{'}{C}_t + \Gamma_f \times C_{t-1}$	–	Update cell state	(15)
$h_t = \Gamma_o \cdot \tanh(C_t)$	–	Confirm outcome	(16)

In Eq. (11), W_f is a weight matrix that controls the behavior of the forget gate. If the values of the forget gate vector of Γ_f are zero (or tend to zero), it practically means that the content of C_{t-1} is not taken into account. In other words, it means it removes the network of information provided by C_{t-1} and disregards it. Similarly, if the values of the vector Γ_f Tend to one, and the network stores this information. The three gate states that execute the LSTM operation, which regulates word information as a continuous value between 0 and 1, are represented by Eq. (16). Each cell has a forget gate in addition to an input and an output. Equation (17) is shown with each gate output values i, f, and o. In addition, the h hidden state of the LSTM cell is written in each t step to store long-term information. For LSTM, each weight vector cell is stored in W, and the value of vector b is to adjust the bias.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} sigmoid \\ sigmoid \\ sigmoid \\ tanh \end{pmatrix} w^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} + \begin{pmatrix} b_i \\ b_f \\ b_o \\ b_c \end{pmatrix} \tag{17}$$

According to Eq. (18), the last layer of the proposed model is a fully linked layer that uses the feature that was retrieved from the LSTM layer.

$$D_i^k = \sum_j w_{ji}^{k-1} \left(\sigma\left(h_i^{k-1}\right) + b_i^{k-1}\right) \tag{18}$$

In Eq. (18) σ is the non-linear activation function, $h^k = \{h^1, h^2, \ldots, h^k\}$ is the feature vector, k is the number of LSTM units, w_{ji}^{k-1} weight size of the ith unit of the $k-1$ unit, and b_i^{k-1} i is biased. The LSTM network converts the feature maps to the hidden states. This feature map has n hidden states h_n. The feature map is indicated by the letter H (Eq. 19).

$$H = (h_1, h_2, \ldots, h_n) \tag{19}$$

The softmax function helps probability distribution generate values such that the calculated values depend on a particular class. The proposed model defines the softmax function according to Eq. (20). So, x_i is a numerical value entering the prior layer, and M shows the number of classes. The softmax function computes the probabilities of each target class compared to all.

$$softmax\ function = \hat{y}_i = \emptyset(x_i) = \frac{e^{x_i}u}{\sum_{k=1}^{M} e^{x_k}u}; \quad k = 1, 2 \ldots M;\ x_i \in R \tag{20}$$

$$u_t = \tanh(Wh_t + b) \tag{21}$$

In Eq. (21) u_t is a hidden situation of h_t, the hidden state h_t is first fed into a fully connected layer with an activation function of $tanh$. The alignment coefficient of accuracy is calculated by multiplying the transpose of the output u_t by the trainable parameter vector x.

4.5 The CSA Algorithm for Optimizing the Hyperparameters of the LSTM

The CSA algorithm achieves the optimal value in the hyperparameter space with different movements of the agents. The different behaviors of the agents lead to the updating of the hyperparameters, and an agent discovers the best hyperparameter value (the best position). Table 2 shows the LSTM hyperparameter search space by the CSA algorithm.

As part of the process, the input is divided into training and testing, with 80% divided into training and 20% divided into testing. After that, in the training stage, the proposed method generates a set of solutions corresponding to each parameter of the LSTM. In the following step, the fitness function can be applied. In this study, RMSE is used as a fitness function, which can be expressed via Eq. (22).

$$RMSE = \sqrt{\frac{1}{n}\sum(y - \tilde{y})^2} \tag{22}$$

where y is the real value, and $\tilde{y}$ is the predicted value.

Table 2 LSTM hyperparameter search space by CSA algorithm

Hyperparameter	Search range	Hyperparameter characteristics
Number of neurons on the first LSTM layer	[25,150]	Discrete
Number of neurons on the second LSTM layer	[20,80]	Discrete
Dropout rate	[0,0.5]	Continuous
Learning rate	$[10^{-4},10^{-2}]$	Continuous

4.6 Evaluation Criteria

In this section, the most important assessment criteria for MLTC [21] are broken down and discussed. Assume that the total number of samples contained in the dataset under test is m. If i is less than or equal to m, then it denotes a sample from the test dataset. Zi and Yi are shorthand for the labels that were expected and actual, respectively.

$$\text{Accuracy} = \frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} \frac{|\mathrm{Z_i} \cap \mathrm{Y_i}|}{|\mathrm{Z_i} \cup \mathrm{Y_i}|} \tag{23}$$

$$\text{Precision} = \frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} \frac{|\mathrm{Z_i} \cap \mathrm{Y_i}|}{|\mathrm{Z_i}|} \tag{24}$$

$$\text{Recall} = \frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} \frac{|\mathrm{Z_i} \cap \mathrm{Y_i}|}{|\mathrm{Y_i}|} \tag{25}$$

$$F-Measure = \frac{1}{m} \sum_{i=1}^{m} \frac{2 * |Z_i \cap Y_i|}{|Z_i| + |Y_i|} \tag{26}$$

5 Evaluation and Results

This section evaluated the Python 3.8 programming language model based on Anaconda (Free and open-source circulation of Python) on various textual datasets. Python is used $TensorFlow\&Keras$ based on a deep learning library to MLTC. TensorFlow is one of the most popular libraries for developing and training neural networks. In Table 3, the value of the parameters for the implementation of the proposed model is determined. The value of LSTM parameters is determined by CSA. The initial population and the number of iterations in the CSA algorithm are 50 and 200, respectively. Also, to show the efficiency of the CSA algorithm, the Whale Optimization Algorithm (WOA) [36] and Gradient-based Optimizer (GBO) [37] algorithms have been used for comparison. The main goal of WOA and GBO is to find the optimal value for LSTM hyperparameters.

Filters, dropout rate to avoid overfitting, pool size, activation function, kernel size, batch size, learning rate, hidden layer number, and epochs are the essential parameters. The learning and dropout rates were 0.001 and 0.26, respectively. A Core i7 CPU 2.4 GHz, 8 GB RAM, and Windows 10 were used for the evaluation. 80% of the randomly selected texts were utilized for training classification and 20% for testing.

Table 4 shows that four datasets were evaluated. Four multi-label text datasets—RCV1-v2, EUR-Lex, Reuters-21578, and Bookmarks—are chosen. These are text datasets. RCV1-v2, EUR-Lex, Reuters-21578, and Bookmarks feature 47,236, 26,575, 18,637, and 2150.

5.1 Evaluation Based on Training and Testing

Table 5 illustrates the models' results based on 80% training and 20% testing. The proposed model has a more significant percentage of accuracy compared to CNN and LSTM. The proposed model has accuracy percentages of 84.71, 45.73, 63.92, and 41.82 on RCV1-v2, EUR-Lex, Reuters-21578, and Bookmarks datasets. Table 5 shows that the accuracy percentages of CNN and LSTM models on the RCV1-V2 dataset were 79.52 and 81.52, respectively. On the EUR-Lex dataset, the accuracy percentages of CNN and LSTM models

Table 3 The proposed model parameters

Models	Parameters	Values
CNN	Convolutional_layer_1	Filters = 20
		Activation function = ReLU
		Kernel size = 5 × 5
	Max-pooling	Pool-size = 2 × 2
	Convolutional_layer_2	Filters = 20
		Activation function = ReLU
		Kernel size = 3 × 3
	Max-pooling	Pool-size = 2 × 2
LSTM-CSA	Number of layers	2
	batch size	25
	epochs	100
	hidden neurons for the first layer	80
	hidden neurons for the second layer	50
	Dropout rate	0.26
	Learning rate	0.001
CSA [20]	Maximum number of iterations	200
GBO [37]	Population size	50
WOA [36]		

Table 4 Specifications of multi-label text datasets [25]

Datasets	Number of total texts	Number of labels	Number of train texts	The average number of labels per text	Size of total features	Number of test texts
Reuters-21578	10,789	90	8631	1.13	18,637	2158
EUR-Lex	19,348	3993	15,478	5.32	26,575	3870
RCV1-v2	804,414	103	723,531	3.24	47,236	160,883
Bookmarks	87,856	208	70,285	2.03	2150	17,571

were 40.39 and 42.86, respectively. On the Reuters-21578 dataset, the accuracy percentages of the CNN and LSTM models were 61.25 and 62.61, respectively. On the Bookmarks dataset, the accuracy percentages of the CNN and LSTM models were 39.74 and 40.19, respectively. The proposed model on RCV1-v2 had a relative improvement of 6.13% and 3.77% compared to CNN and LSTM. The proposed EUR-Lex model had a relative improvement of 11.68% and 6.28% compared to CNN and LSTM.

Table 6 shows the results of the LSTM-CSA model with LSTM-GBO and LSTM-WOA models. The accuracy percentage of the LSTM-CSA model is higher compared to LSTM-GBO and LSTM-WOA. The accuracy percentage of LSTM-GBO and LSTM-WOA models on RCV1-v2 is 83.17 and 82.34. The accuracy percentage of LSTM-GBO and LSTM-WOA

Table 5 Comparison of models based on 80% training and 20% testing

Datasets	Models	Recall	Precision	Accuracy	F-measure
RCV1-v2	CNN	79.63	79.15	79.52	79.39
	LSTM	81.94	80.27	81.52	81.10
	Proposed model	84.09	83.58	84.71	83.83
EUR-Lex	CNN	40.52	40.11	40.39	40.33
	LSTM	43.68	42.72	42.86	43.19
	Proposed model	46.35	45.85	45.73	46.27
Reuters-21578	CNN	61.41	60.54	61.25	60.97
	LSTM	61.82	61.19	62.61	61.50
	Proposed model	63.79	63.34	63.92	63.56
Bookmarks	CNN	39.16	39.94	39.74	39.55
	LSTM	41.57	40.48	40.19	41.02
	Proposed model	40.32	41.56	41.82	40.93

Table 6 LSTM-CSA model results with LSTM-GBO and LSTM-WOA models

Datasets	Models	Recall	Precision	Accuracy	F-measure
RCV1-v2	Proposed model (LSTM-CSA)	84.09	83.58	84.71	83.83
	Proposed model (LSTM-GBO)	82.92	82.67	83.17	82.79
	Proposed model (LSTM-WOA)	82.14	81.83	82.34	81.98
EUR-Lex	Proposed model (LSTM-CSA)	46.35	45.85	45.73	46.10
	Proposed model (LSTM-GBO)	45.11	44.79	45.23	44.95
	Proposed model (LSTM-WOA)	42.34	41.94	42.61	42.14
Reuters-21578	Proposed model (LSTM-CSA)	63.79	63.34	63.92	63.56
	Proposed model (LSTM-GBO)	62.22	62.12	62.54	62.17
	Proposed model (LSTM-WOA)	60.95	60.84	61.19	60.89
Bookmarks	Proposed model (LSTM-CSA)	40.32	41.56	41.82	40.93
	Proposed model (LSTM-GBO)	39.08	38.61	39.37	38.84
	Proposed model (LSTM-WOA)	37.92	37.55	38.12	37.73

models on bookmarks is 41.82 and 39.37. In this comparison, CSA shows varying exploration behaviors at different stages of optimization, proving that it is a strong exploration operator.

The proposed model confirms achieving the highest performance on datasets of MLTC in comparison with CNN and LSTM models. Figure 6 shows the accuracy according to the learning epoch. After 100 epochs on datasets (Bookmarks, EUR-Lex, Reuters-21578, and RCV1-v2), the experiments show higher accuracy than the basic proposed model.

Table 7 illustrates the model's results based on 70% training and 30% testing of the texts. It demonstrates that the proposed model outperforms CNN and LSTM in terms of accuracy. The suggested model's accuracy percentages on the RCV1-v2, EUR-Lex, Reuters-21578, and Bookmarks datasets were 82.03, 45.28, 62.26, and 39.43, respectively. The accuracy percentages of CNN and LSTM models on the RCV1-V2 dataset were 78.27 and 79.92,

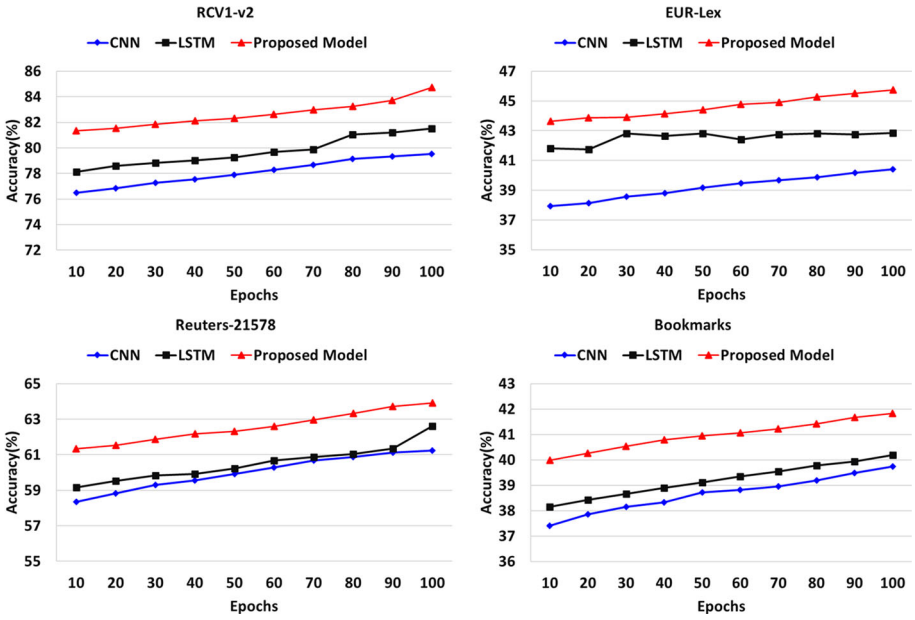

Fig. 6 Performance comparison according to epoch

Table 7 Comparison of models based on 70% training and 30% testing

Datasets	Models	Recall	Precision	Accuracy	F-measure
RCV1-v2	CNN	78.94	78.31	78.27	78.62
	LSTM	80.26	79.35	79.92	79.80
	Proposed model	81.51	81.12	82.03	81.31
EUR-Lex	CNN	40.15	39.96	39.86	40.05
	LSTM	43.32	42.25	42.45	42.78
	Proposed model	46.06	45.21	45.28	45.63
Reuters-21578	CNN	60.14	59.88	60.23	60.01
	LSTM	61.29	60.85	61.91	61.07
	Proposed model	62.31	62.15	62.26	62.23
Bookmarks	CNN	36.82	35.69	35.95	36.25
	LSTM	37.42	36.91	36.94	37.16
	Proposed model	39.65	39.25	39.43	39.45

respectively, as shown in Table 7. On the EUR-Lex dataset, the accuracy percentages of CNN and LSTM models were 39.86 and 42.45, respectively. On the Reuters-21578 dataset, the CNN and LSTM models had accuracy percentages of 60.23 and 61.91, respectively. On the Bookmarks dataset, the accuracy percentages of the CNN and LSTM models were 35.95 and 36.94, respectively.

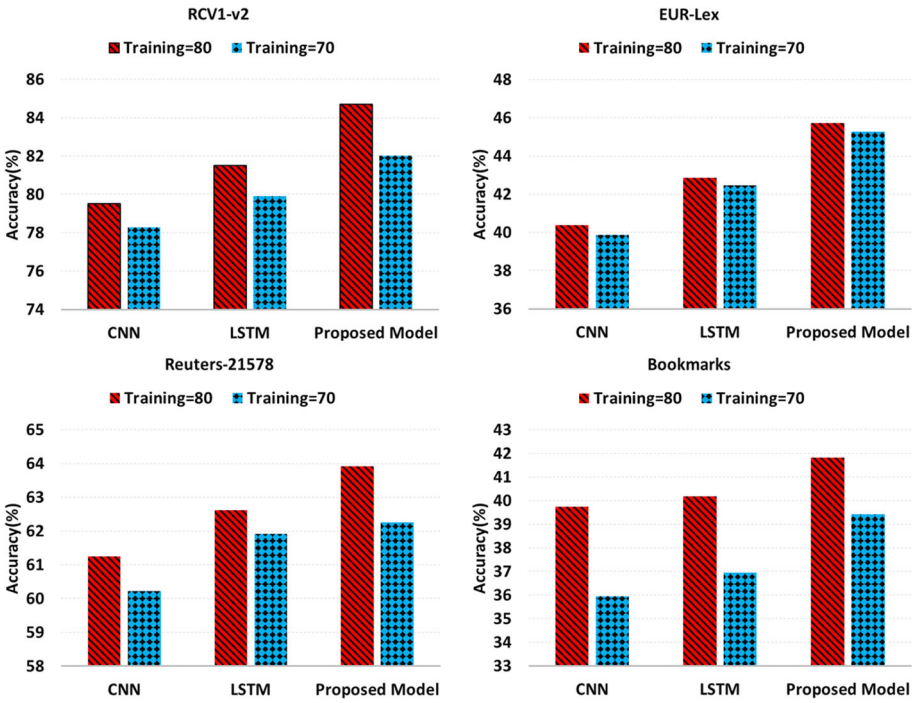

Fig. 7 Comparison chart based on the percentage of training

Figure 7 shows a comparison chart based on the percentage of training for the proposed model. The accuracy rate of the proposed model is significantly greater than that of CNN and LSTM. If Training is equal to 80, then the proposed model has a better percentage of accuracy than it did before.

5.2 Evaluation of Models Based on the Number of Iterations

Figure 8 shows the results of the proposed model based on different iterations on different datasets. The results show that the proposed model has achieved higher accuracy with 200 iterations. The accuracy percentage of the proposed model with 100 and 200 repetitions on RCV1-v2 is 81.25 and 84.25, respectively. The accuracy percentage of the proposed model with 100 and 200 repetitions on EUR-Lex is 43.36 and 46.72, respectively. The number of repetitions has a direct effect on increasing accuracy. The number of iterations produces optimal solutions and creates a complete search in the problem space. The proposed model was tested with more than 200 repetitions, but there was no significant change in the accuracy of the results.

5.3 Evaluation Based on Error Indicators

In this section, four indicators of mean absolute error (MAE), mean absolute percentage error (MAPE), root-mean-square error (RMSE), and R-square are used. The best value for

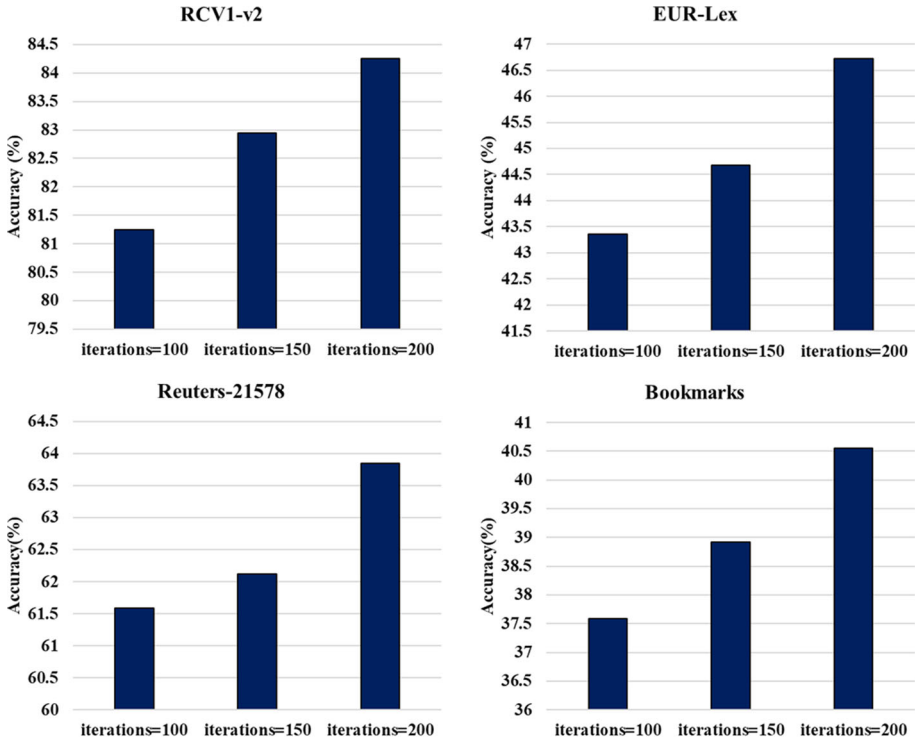

Fig. 8 The results of the proposed model based on different iterations of different datasets

RMSE, MAE, and MSE indicators is close to 0. The best R-square index value is close to 1, which indicates that the predicted value is closer to the actual value and the prediction result is better.

$$\mathrm{MAE} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \left| \hat{z}_i - \mathrm{z_i} \right| \tag{27}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \left(\hat{\mathrm{z}}_\mathrm{i} - \mathrm{z_i} \right)^2} \tag{28}$$

$$R^2 = 1 - \frac{\sum_i \left(\hat{z}_i - z_i \right)^2}{\sum_i \left(\overline{z}_i - z_i \right)^2} \tag{29}$$

where $\hat{z}_i$ is the predicted value, $\mathrm{z_i}$ is the true value, $\overline{z}_i$ is the mean of the true value, and n is the number of training or test sets. Table 8 shows the MAE, RMSE, and R-square results. The proposed model has the lowest MAE. The value of the R-square in the proposed model is closer to 1. The value of MAE on RCV1-v2 by the proposed model is 8.624. The value of MAE on EUR-Lex by the proposed model is 12.924. The RMSE value on Reuters-21578 by the proposed model is 7.032. The value of RMSE on Bookmarks by the proposed model is 4.621.

Table 8 Results of the MAE, RMSE, and R-square

Dataset	Evaluation indicators	MAE	RMSE	R-square
RCV1-v2	CNN	14.621	7.912	0.823
	LSTM	11.658	5.062	0.861
	Proposed model	8.624	3.321	0.941
EUR-Lex	CNN	15.617	11.003	0.826
	LSTM	13.834	9.607	0.849
	Proposed model	12.924	8.351	0.886
Reuters-21578	CNN	15.942	10.030	0.830
	LSTM	12.535	8.621	0.861
	Proposed model	10.384	7.032	0.895
Bookmarks	CNN	18.065	8.321	0.837
	LSTM	13.282	6.621	0.860
	Proposed model	9.684	4.621	0.916

5.4 Evaluation Based on Word Frequency Methods

The findings of the proposed model based on various word frequencies are illustrated in Table 9. Some of the frequency models have effects on the accuracy percentage. For example, the TF-IDF method had a particular advantage over many other methods. In the TF-IDF method, words are first collected individually to create a feature set for each text. Second, the TF-IDF score is calculated for each word in each document. Third, all different words in a document are arranged according to their scores calculated by the TF-IDF. Various percentages of discrete words with greater TF-IDF scores) are stored to develop the feature set (vocabulary). A set of features is generated for the entire set of texts by correlating each text's words. If frequency methods correctly extract each text's essential features at the beginning of a classification operation, then the proposed model's class recognition accuracy will increase.

5.5 Evaluation Based on the Number of Convolutional Layers

Table 10 displays the proposed model's results, which are broken down by the number of convolutional layers. It is determined that the network layers have a significant influence on the proposed model's performance. The accuracy % increases as the number of layers are increased to two. The accuracy percentage on the RCV1-v2 dataset is 84.71 when the number of layers is two, and 82.29 when the number of layers is four. On the EUR-Lex dataset, the accuracy % is 45.73 when the number of layers is equal to 2, and 45.62 when the number of layers is equal to 4. On the Reuters-21578 dataset, the accuracy percentage is 63.92 when there are two layers, and it is 63.34 when there are four levels. On the Bookmarks dataset, the accuracy % is 41.82 when the number of layers is 2, and 39.46 when the number of layers is 4.

Based on the results from four different datasets, it was determined that the proposed model had a more significant accuracy percentage. The proposed model had different functions based

Table 9 The results of the proposed model based on various word frequency methods

Datasets	Frequency models	Recall	Precision	Accuracy	F-measure
RCV1-v2	TF-IDF	84.09	83.58	84.71	83.83
	TF-RF	82.54	82.17	83.36	82.35
	TF-IDF-ICF	83.63	82.43	83.56	83.03
	TF-IDF-ICSDF	84.81	83.25	84.79	84.02
	TF-IGM	83.85	82.46	83.61	83.15
EUR-Lex	TF-IDF	46.35	45.85	45.73	46.27
	TF-RF	46.14	45.31	44.16	45.72
	TF-IDF-ICF	45.95	45.76	45.51	45.85
	TF-IDF-ICSDF	46.22	45.65	45.82	45.93
	TF-IGM	46.41	45.81	45.89	46.11
Reuters-21578	TF-IDF	63.79	63.34	63.92	63.48
	TF-RF	63.26	63.17	63.23	63.21
	TF-IDF-ICF	63.24	63.26	63.37	63.25
	TF-IDF-ICSDF	63.56	63.38	63.69	63.47
	TF-IGM	63.67	63.27	63.84	63.47
Bookmarks	TF-IDF	39.97	39.56	39.74	39.72
	TF-RF	39.23	39.18	39.19	39.20
	TF-IDF-ICF	39.46	39.41	39.26	39.43
	TF-IDF-ICSDF	39.42	39.34	39.48	39.38
	TF-IGM	39.81	39.52	39.82	39.66

Table 10 Results of the proposed model based on the number of convolutional layers

Datasets	Layers	Recall	Precision	Accuracy	F-measure
RCV1-v2	2	84.09	83.58	84.71	83.83
	3	82.45	82.23	82.38	82.34
	4	82.32	82.11	82.29	82.21
EUR-Lex	2	46.35	45.85	45.73	46.27
	3	45.59	45.62	45.55	45.60
	4	45.36	45.28	45.62	45.32
Reuters-21578	2	63.79	63.34	63.92	63.56
	3	63.62	63.16	63.71	62.39
	4	63.44	63.38	63.34	63.41
Bookmarks	2	40.32	41.56	41.82	40.93
	3	40.57	40.35	40.22	40.46
	4	39.26	39.17	39.46	39.21

on word frequency and FS. Therefore, selecting features and forming the initial word weights matrix is critical for network training.

5.6 Comparison and Evaluation

This section compares the proposed model to other models [25]. The comparison results indicate that the efficiency percentage of the proposed model was greater than that of other models. On the RCV1-v2 dataset, DSRM-DNN-1 and DSRM-DNN-2 had respective accuracy rates of 0.8164 and 0.8326. Table 11 demonstrates that the suggested model for RCV1-v2 yields the best results, with Precision and Recall values of 83.58% and 84.09%, respectively. On the EUR-Lex dataset, DSRM-DNN-1 and DSRM-DNN-2 had precision percentages of 0.4298 and 0.4315, respectively, while the suggested model had a precision percentage of 0.4585. On the Reuters-21578 dataset, DSRM-DNN-1 and DSRM-DNN-2 had precision percentages of 0.4913 and 0.6147, respectively, while the suggested model had a precision percentage of 0.6334. On the Bookmarks dataset, DSRM-DNN-1 and DSRM-DNN-2 had respective accuracy rates of 0.3841 and 0.4018. With a Precision of 41.56%, a Recall of 40.32%, and a Precision of 40.93%, the model suggested for Bookmarks yields the best results.

6 Conclusions and Future Works

MLTC is a significant and difficult subset of natural language processing. It aims to classify texts based on different classes to provide maximum accuracy. Because of this field's specific complexities, many MLTC methods have faced limitations in accuracy, algorithm complexity, and the inability to predict. Therefore, this paper introduced a new model based on a hybridization of CNN and LSTM-CSA for MLTC. The main goal of the proposed model was to improve CNN using LSTM. The CNN architecture includes a cascading model of pooling and convolutional layers. CNN can manage large amounts of data and has better-classified texts using the ReLU activator function. However, CNN's main problem is that its training can sometimes be very time-consuming. Therefore, using LSTM can solve CNN problems in the face of various data. Significant hyperparameters for LSTM were found by CSA. Experiment findings on four distinct datasets revealed that the proposed model outperformed CNN and LSTM in terms of accuracy. There are several shortcomings and limitations in this article that can be discussed. (1) The hybrid model has not been tested on different data sets with different texts. (2) The CNN network performs poorly in the communication between texts and cannot record similar data. Therefore, the obtained features cannot perform the classification of texts correctly. The number of kernels and the size of the kernels must be determined precisely. In the future, a more robust approach is proposed to overcome the mentioned limitations. In the future, we will investigate CNN models with other optimization, and try to act on this model with more complicated characteristics to improve MLTC outcomes.

Table 11 Comparison of the proposed model with other models

Datasets	Models	Recall	Precision	F-measure
RCV1-v2	Multi-label K-nearest neighbor (ML-KNN)	0.5676	0.5741	0.5708
	Multi-label decision tree (ML-DT)	0.5626	0.3961	0.4649
	Binary relevance (BR)	0.6237	0.7946	0.6989
	Classifier chains (CC)	0.6449	0.7682	0.7012
	Hierarchical ARAM neural network (HARAM)	0.5693	0.7756	0.6566
	Back-propagation (BP)	0.5803	0.4385	0.4995
	Convolutional and recurrent neural networks (CNN-RNN)	0.6465	0.8034	0.7165
	Hierarchical label set expansion (HLSE)	0.5876	0.7825	0.6712
	Supervised representation learning (SERL)	0.6215	0.8015	0.7001
	DSRM-DNN-1	0.6413	0.8164	0.7183
	DSRM-DNN-2	0.6395	0.8326	0.7234
	Proposed model	0.8409	0.8358	0.8383
EUR-Lex	ML-DT	0.2254	0.1567	0.1849
	ML-KNN	0.2318	0.5141	0.3195
	BR	0.3643	0.4260	0.3927
	CC	0.3012	0.2618	0.2801
	HARAM	0.3271	0.4158	0.3662
	BP	0.3063	0.2331	0.2647
	CNN-RNN	0.3103	0.3727	0.3387
	HLSE	0.3942	0.3854	0.3898
	SERL	0.3853	0.4085	0.3966
	DSRM-DNN-1	0.3932	0.4298	0.4109
	DSRM-DNN-2	0.4107	0.4315	0.4208
	Proposed model	0.4635	0.4585	0.4627
Reuters-21578	ML-DT	0.2806	0.3510	0.3119
	ML-KNN	0.2056	0.3422	0.2569
	BR	0.3507	0.4645	0.3997
	CC	0.3613	0.4706	0.4088
	HARAM	0.2424	0.2981	0.2674
	BP	0.4761	0.4809	0.4785
	CNN-RNN	0.2875	0.3697	0.3235
	HLSE	0.3974	0.4582	0.4256
	SERL	0.4520	0.4796	0.4654
	DSRM-DNN-1	0.4831	0.4913	0.4872
	DSRM-DNN-2	0.4785	0.6147	0.5381
	Proposed model	0.6379	0.6334	0.6356

Table 11 (continued)

Datasets	Models	Recall	Precision	F-measure
Bookmarks	ML-DT	0.1458	0.1324	0.1388
	ML-KNN	0.2780	0.2514	0.2640
	BR	0.1880	0.1950	0.1914
	CC	0.3104	0.1642	0.2148
	HARAM	0.3409	0.3614	0.3509
	BP	0.2743	0.1115	0.1586
	CNN-RNN	0.3321	0.2257	0.2688
	HLSE	0.3274	0.3472	0.3370
	SERL	0.3603	0.3715	0.3658
	DSRM-DNN-1	0.3596	0.3841	0.3714
	DSRM-DNN-2	0.3702	0.4018	0.3854
	Proposed model	0.4032	0.4156	0.4093

Author contributions Author Contributions: "Conceptualization, H.K.M, and F.S.G; methodology, H.K.M; software, H.K.M, and F.S.G; validation, F.S.G, K.M, and A.B.S; formal analysis, H.K.M; investigation, F.S.G; resources, H.K.M; data curation, H.K.M; writing—original draft preparation, H.K.M; writing—review and editing, F.S.G; visualization, K.M; supervision, F.S.G, K.M.; project administration, F.S.G; All authors have read and agreed to the published version of the manuscript."

Data Availability The datasets downloaded and analyzed during this paper are available in the UCI repository.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Mulahuwaish A et al (2020) Efficient classification model of web news documents using machine learning algorithms for accurate information. Comput Secur 98(1):102006
2. Liu C, Wang X (2020) Quality-related English text classification based on recurrent neural network. J Vis Commun Image Represent 71(1):102724
3. Mahmoudi M, Soleimanian Gharehchopogh F (2018) An improvement of shuffled frog leaping algorithm with a decision tree for feature selection in text document classification. CSI J Comput Sci Eng 16(1):60–72
4. Allahverdipour A, Soleimanian Gharehchopogh F (2018) An improved K-nearest neighbor with crow search algorithm for feature selection in text documents classification. J Adv Comput Res 9(2):37–48
5. Majidpour H, Soleimanian Gharehchopogh F (2018) An improved flower pollination algorithm with Adaboost algorithm for feature selection in text documents classification. J Adv Comput Res 9(1):29–40

6. Stein RA, Jaques PA, Valiati JF (2019) An analysis of hierarchical text classification using word embeddings. Inf Sci 471(1):216–232
7. Kim H-J et al (2018) Towards perfect text classification with wikipedia-based semantic Naïve Bayes learning. Neurocomputing 315(1):128–134
8. Bharath Bhushan SN, Danti A (2017) Classification of text documents based on score level fusion approach. Pattern Recognit Lett 94(1):118–126
9. Udandarao V et al (2021) InPHYNet: Leveraging attention-based multitask recurrent networks for multi-label physics text classification. Knowl-Based Syst 211(1):106487
10. Liu Y, Bi J-W, Fan Z-P (2017) Multi-class sentiment classification: the experimental comparisons of feature selection and machine learning algorithms. Expert Syst Appl 80(1):323–339
11. Salles T et al (2018) Improving random forests by neighborhood projection for effective text classification. Inf Syst 77(1):1–21
12. Gharehchopogh FS, Ibrikci T (2023) An improved African vultures optimization algorithm using different fitness functions for multi-level thresholding image segmentation. Multimed Tools Appl 83:1–47
13. Gharehchopogh FS, Shayanfar H, Gholizadeh H (2020) A comprehensive survey on symbiotic organisms search algorithms. Artif Intell Rev 53:2265–2312
14. SaiSindhuTheja R, Shyam GK (2021) An efficient metaheuristic algorithm based feature selection and recurrent neural network for DoS attack detection in cloud computing environment. Appl Soft Comput 100(1):106997
15. Gong J et al (2020) Hierarchical graph transformer-based deep learning model for large-scale multi-label text classification. IEEE Access 8(1):30885–30896
16. Wang R et al (2021) A novel reasoning mechanism for multi-label text classification. Inf Process Manag 58(2):102441
17. Li B et al (2021) DNC: a deep neural network-based clustering-oriented network embedding algorithm. J Netw Comput Appl 173(1):102854
18. Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
19. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
20. Xu Y et al (2022) Competitive search algorithm: a new method for stochastic optimization. Appl Intell 52(11):12131–12154
21. Aljedani N, Alotaibi R, Taileb M (2020) HMATC: hierarchical multi-label Arabic text classification model using machine learning. Egypt Inform J 22:225–237
22. Gargiulo F et al (2019) Deep neural network for hierarchical extreme multi-label text classification. Appl Soft Comput 79(1):125–138
23. Liao W et al (2020) Improved sequence generation model for multi-label classification via CNN and initialized fully connection. Neurocomputing 382(1):188–195
24. Bello M et al (2020) Deep neural network to extract high-level features and labels in multi-label classification problems. Neurocomputing 413(1):259–270
25. Wang T et al (2020) A multi-label text classification method via dynamic semantic representation model and deep neural network. Appl Intell 50(8):2339–2351
26. Alam F, Ofli F, Muhammad I (2018) CrisisMMD: multimodal twitter datasets from natural disasters. Soc Inf Netw 1(1):1–9
27. Hassan J, Shoaib U (2020) Multi-class review rating classification using deep recurrent neural network. Neural Process Lett 51(1):1031–1048
28. Yang M et al (2019) Investigating the transferring capability of capsule networks for text classification. Neural Netw 118(1):247–261
29. Liu J, et al (2017) Deep learning for extreme multi-label text classification: In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval. Association for Computing Machinery, Shinjuku, pp 115–124
30. Li Q et al (2020) Improving convolutional neural network for text classification by recursive data pruning. Neurocomputing 414(1):143–152
31. Guo B et al (2019) Improving text classification with weighted word embeddings via a multi-channel TextCNN model. Neurocomputing 363(1):366–374
32. Banerjee I et al (2019) Comparative effectiveness of convolutional neural network (CNN) and recurrent neural network (RNN) architectures for radiology text report classification. Artif Intell Med 97:79–88
33. Xiao Y et al (2021) History-based attention in Seq2Seq model for multi-label text classification. Knowl-Based Syst 224(1):107094
34. Liu, H., et al., *Multi-label text classification via joint learning from label embedding and label correlation.* Neurocomputing, 2021.

35. Peng D, Yang J, Lu J (2020) Similar case matching with explicit knowledge-enhanced text representation. Appl Soft Comput 95(1):106514
36. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
37. Ahmadianfar I, Bozorg-Haddad O, Chu X (2020) Gradient-based optimizer: a new metaheuristic optimization algorithm. Inf Sci 540:131–159