Computing and Informatics, Vol. 42, 2023, 651–666, doi: 10.31577/cai_2023_3_651

CLASSIFICATION OF SENTIMENT USING OPTIMIZED HYBRID DEEP LEARNING MODEL

Chaima Ahle Touate, Rachid El Ayachi, Mohamed Biniz

Faculty of Science and Technology
Sultan Moulay Slimane University
Mghila P3225
230 00 Beni Mellal, Morocco
e-mail: ahle.touate.chaima@gmail.com, rachid.elayachi@usms.ma, mohamedbiniz@gmail.com

Abstract. Sentiment classification plays a pivotal role in natural language processing (NLP), and prior research has established the efficacy of utilizing convolutional neural networks (CNNs) and long short-term memory (LSTM) in this task. However, these approaches suffer from individual performance limitations: CNNs are limited to extracting local information and fail to express context information adequately, while LSTM networks excel at extracting context dependencies but exhibit long training times. To address this issue, we propose a novel text classification algorithm based on a hybrid CNN-LSTM model that leverages the strengths of both approaches and overcomes their limitations by combining them. Our approach is evaluated on the IMDB dataset, and we present a hyperparameter optimization framework utilizing Random Search to increase the likelihood of producing an optimally performing model.

Keywords: Document classification, CNN, LSTM, hybrid models, hyperparameter tuning, random search

1 INTRODUCTION

Text classification has garnered significant attention in recent years and is considered one of the fundamental tasks in natural language processing (NLP) with various applications, such as sentiment analysis and topic labeling.

Traditional text classification methods rely on statistics and feature selection [1], which include commonly used algorithms like Naive Bayes [2], Support Vector Machine (SVM) [3], Decision Trees [4], and others. These classic machine-learning techniques have achieved remarkable results in text classification tasks. However, the introduction of text convolutional neural networks by Yoon Kim has led to the emergence of a variety of deep learning text classification methods [5]. This demonstrates the feasibility of applying artificial neural networks, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to the field of text classification.

Although CNNs have the potential to extract local information, they may fail to capture long-distance dependencies. On the other hand, LSTMs can address this limitation by modeling texts sequentially across sentences. Despite the development of NN-based and word-embedding techniques, sentiment analysis remains challenging [5].

In this study, we propose a hybrid CNN-LSTM model consisting of two parts, CNN and LSTM, to predict the sentiment expressed in texts. We add Hybrid Attention to fully exploit the respective advantages of CNNs and LSTMs and fill their gaps by selectively learning long sequences and making deep neural networks in each training batch. Our proposed model can learn distinct feature forms, improve model learning and expression skills, and prevent overfitting.

The remainder of this paper is structured as follows: Section 2 reviews related works, Section 3 details the architecture of the developed classification system, Section 4 outlines the Hyper-parameters tuning, Section 5 presents the experimental results, Section 6 exhibits the evaluation methods followed by Section 7, which discuss the results, and finally, a conclusion in Section 8.

2 RELATED WORKS

The field of text classification has seen a surge in interest due to the advent of deep learning techniques that require less feature engineering and have the potential to achieve high accuracy. Yoon Kim introduced the convolutional neural network (CNN) to text classification and showed its ability to capture local correlations in the sentence through multiple kernels of varying sizes [6]. Since then, many researchers have proposed CNN-based models, but it was found that CNN lacked context relations. Therefore, to improve classification accuracy, some studies utilized a combination of CNN and Long Short-Term Memory (LSTM) [7, 8, 9].

Zhou et al. proposed a CNN-LSTM model [10], which leverages CNN to extract higher-level phrase representations and feeds them into an LSTM to obtain the sentence representation.

LSTM, CNN, and their hybrid counterparts have been successfully applied in a variety of natural language processing tasks, including sentiment analysis. Rehman et al. proposed an overly deep CNN-LSTM hybrid model [11], which includes dropout techniques, normalization techniques, and rectified linear units to

enhance the prediction accuracy. Although other studies have used similar hybrid approaches [12], the lack of an attention mechanism resulted in less improved results.

However, selecting the hyperparameters to train CNN models can become computationally expensive but is crucial for achieving optimal performance. Several works have addressed this issue and used optimization techniques such as random search [13]. Compared to grid search, random search provides several advantages, such as being able to add new trials to the experiment on the go, allowing changes in resolution, and stopping the experiment at any time [14, 15].

In this paper, we propose an optimized hybrid sentiment classification model that overcomes the limitations of the previous models. The experimental results show that our proposed model can effectively improve text classification accuracy.

3 PROPOSED ARCHITECTURE OF THE DOCUMENT CLASSIFICATION SYSTEM

3.1 Hybrid CNN and LSTM Model

The proposed architecture in this study is based on a hybrid model that combines Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks for text classification, where CNN is applied to extract the complicated features from the text and LSTM is exploited as a classifier. Figure 1 illustrates the structure of this model.

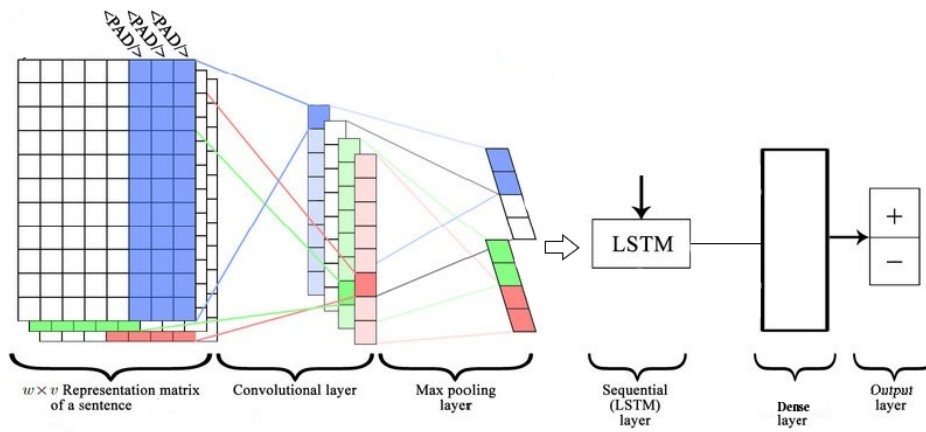

Figure 1. Hybrid model proposed architecture

The Input Layer is the first layer of the model, which consists of a fixed-dimension matrix of distinct vector embeddings representing each review as a row of vectors. Since each sentiment has different tokens, the tokens are based on the words in each review and are embedded with a nonidentical token.

The matrix size of this layer is $w \times v$, where v represents the length of the vector and w represents the number of tokens in the reviews. The maximum length of a review is defined as w, and any review with a lower number of tokens is padded to achieve the same length as the maximum.

The Convolutional Layer is applied to extract complicated features from the text by sliding the filter over the matrix to generate a new feature map. Various filters with different sizes are used to detect different features in the matrix. The filter strides or resorts only one column and one row over the matrix to detect multiple features in a review. An activation function is applied to define these features in the feature map.

The Max-Pooling Layer is utilized to down-sample the features in the feature map and compute the max value as a corresponding feature to a precise filter. The output vectors of this layer are then input to the LSTM networks to measure the long-term dependencies of feature sequences. The top value is selected in this step to attain the most significant feature and reduce the computation in the following layers.

The LSTM Layer is responsible for counting the anterior data and attaining sequential data. The output vectors of the previous layer are taken as inputs to this layer, which consists of a set number of units or cells. The closing vectors output of this layer are interconnected in one matrix in the range between 0 and 1 in the dense layer, and an activation function is used to classify the final output as either positive or negative.

4 TUNING HYPER PARAMETERS

Although this architecture combines CNN and LSTM networks to enhance text classification accuracy, it is computationally expensive to define the hyper-parameters for learning a CNN architecture and testing all the possible sets of hyper-parameters.

To optimize the hyper-parameters, the random search method has been widely used and has more benefits than grid search, as it allows practitioners to change the "resolution" on the go, add new trials to the set, or even ignore the failure test. Figure 2 illustrates a resumed idea about the Tuning method.

Hyper-parameters can be classified into two types [16]:

1. Network structure hyper-parameters, which include:
    - Training optimization algorithm – the method used to train the neural network by minimizing the cost function.
    - Network weight initialization – the process of setting the weights of a neural network to small random values that serve as the starting point for the optimization (learning or training) of the neural network model.
    - Hidden layers – the layers between the input and output layers.
    - Activation functions – mathematical functions that enable the model to learn nonlinear prediction boundaries.

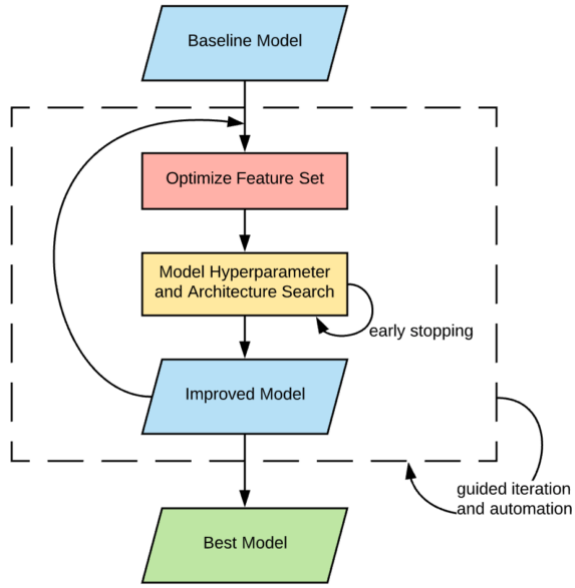

Figure 2. Tuning a model performance process

- Dropout regularization technique – a method for reducing over-fitting in artificial neural networks by preventing complex co-adaptations on training data.

2. Network training hyper-parameters, which include:

- Learning rate – the rate at which the weights are updated at the end of each batch.
- Momentum – a value that controls how much the previous update affects the current weight update.
- Number of epochs – the number of iterations of the entire training dataset to the network during training.
- Batch size – the number of patterns shown to the network before the weights are updated.

As the number of hyper-parameters can exceed 10, identifying the optimal combination can be seen as a search problem. To address this issue, an automatic optimizer, such as Random Search, can be utilized to achieve better results. The figure provided below depicts the process of hyperparameter tuning, which can be broken down into several steps.

The hyper-parameters of the model were trained based on the configurations outlined in Table 1, with values randomly assigned within their specified ranges. The

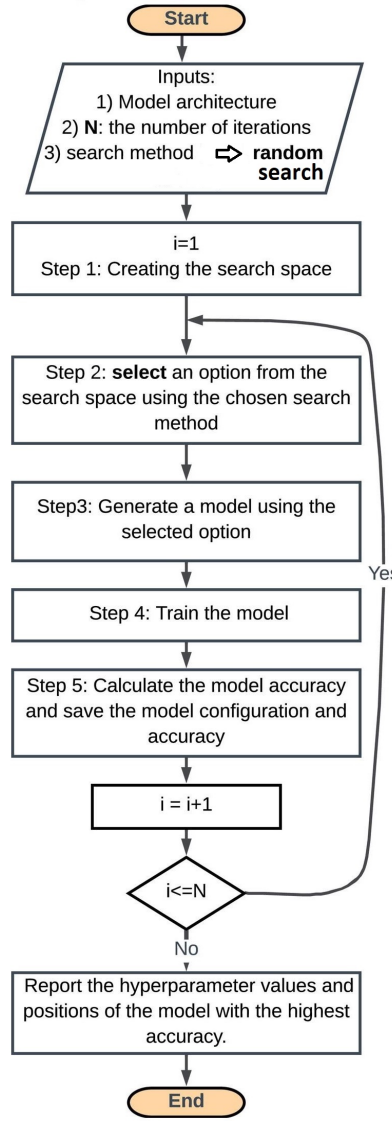

Figure 3. The hyper-parameters tuning process

evaluation of results was performed through multiple experiments. Table 1 presents the hyper-parameters that were considered for this study, with their corresponding ranges indicated within the square brackets.

Hyper-parameter Name	Hyper-parameter Value
Learning rate	[0.001, 0.01, 0.1]
Batch Size	[10, 20, 40, 60, 80]
Epochs	[3, 10, 50]
Momentum	[0.0, 0.2, 0.4, 0.6]
Optimizer	[SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam]
Init mode	[uniform, lecununiform, normal, zero, glorotnormal, glorotuniform, henormal]
Activation Function	[softmax, softplus, softsign, relu, tanh, sigmoid, hardsigmoid, linear]

Table 1. Hyper-parameters and their corresponding values

5 EXPERIMENTAL RESULTS

5.1 Dataset Description

To develop a precise classifier, obtaining an appropriate training dataset is critical, as it should encompass examples that accurately depict the outcomes targeted for prediction. In order to validate the reliability of our model, we conducted experiments utilizing the IMDB dataset for benchmark testing, which is described below.

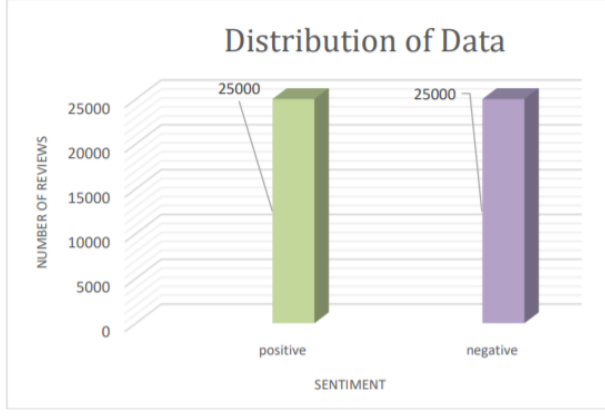

Figure 4. Dataset distribution

The IMDB dataset is a widely used resource in natural language processing and text analytics research. It consists of a large collection of movie reviews, totaling 50 000 in number. The primary objective of this dataset is to facilitate sentiment analysis tasks, which involve determining whether a given review expresses a positive or negative sentiment toward the movie being reviewed.

One of the standout features of the IMDB dataset is its size. Prior to the release of this dataset, benchmark datasets for sentiment analysis typically consisted of only a few thousand reviews. The IMDB dataset, on the other hand, contains a staggering 50 000 reviews, making it one of the largest publicly available datasets for this task.

The dataset is split into two equally sized sets of 25 000 reviews each, one for training and one for testing. This ensures that models developed using the dataset are evaluated on data that is independent of the data used for training, and helps to guard against overfitting. The reviews themselves are highly polar, meaning that they tend to express strong positive or negative sentiments toward the movies being reviewed. This makes the data-set well-suited for tasks such as binary sentiment classification, where the goal is to classify each review as either positive or negative.

In addition to its size and polarity, the IMDB dataset is also noteworthy for its diversity. The reviews cover a wide range of movies, spanning multiple genres, release years, and cultural contexts. This diversity helps to ensure that models developed using the dataset are able to generalize to a wide range of real-world scenarios, rather than being limited to a narrow subset of cases.

6 EVALUATION METHODS

The effectiveness of classifiers in discerning correct outcomes is typically assessed using well-established performance metrics, such as accuracy and loss rate. These measures are defined based on specific characteristics of the classification outcomes, which include:

True Positives (TP) – These refer to instances where the positive outcome is correctly predicted. For instance, when the actual class is positive, and the predicted class is also positive.

True Negatives (TN) – These instances occur when the negative outcome is correctly predicted. For example, when the actual class is negative, and the predicted class is also negative.

False Positives (FP) – These refer to instances where the negative outcome is wrongly predicted as positive. For instance, when the actual class is negative, but the predicted class is positive.

False Negatives (FN) – These refer to instances where the positive outcome is wrongly predicted as negative. For example, when the actual class is positive, but the predicted class is negative.

Accuracy – This is the most intuitive performance measure and is a ratio of the correctly predicted observations to the total observations. Accuracy is a valuable

measure because it provides an estimate of the extent to which the classifier is accurately predicting the outcome, hence its predictive power.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{1}$$

Loss – This is defined as the difference between the predicted and actual values. The most commonly used loss function in deep neural networks is cross-entropy, which is defined as the negative sum of the true value multiplied by the logarithm of the predicted probability. In mathematical terms:

$$CrossEntropy = -\sum_{i}\sum_{j} \log(p_{i,j}) y_{i,j}, \tag{2}$$

where $y_{i,j}$ represents the true value of sample i belonging to class j and $p_{i,j}$ is the predicted probability by the model that sample i belongs to class j.

Overall, the aforementioned evaluation methods enable the researcher to assess the performance of the classifier and determine whether it is effective in predicting the outcomes of interest.

7 RESULTS AND DISCUSSION

7.1 Construction of Environment and Parameter Setting

The use of a custom environment and parameter setting is essential in ensuring the reliability and reproducibility of our results. We recognize the importance of constructing a custom environment and parameter setting for our experiments.

To this end, we chose to use Python as our programming language and TensorFlow as our deep learning framework. Our lab environment, detailed in Table 2, showcases the hardware and software configuration we used in our study.

Despite the fact that our experiment was conducted on a hardware and software configuration with modest specifications, we were able to achieve significant and relevant results. This speaks to the efficiency and effectiveness of the custom environment and parameter settings we constructed, which were tailored to the specific research question we were addressing. Our findings demonstrate that it is possible to achieve meaningful results even with limited hardware resources, as long as the experimental setup is optimized appropriately.

To achieve our aim of sentiment analysis, we implemented a convolutional neural network (CNN) architecture that incorporated word embeddings of length 32 and permitted a maximum review length of 500. We trained the model using a batch size of 60 and applied a 32-filter kernel with a convolution layer kernel size of 3, utilizing ReLU as the activation function and sigmoid in the dense layer. Furthermore, we utilized an early stopping iteration function that minimized the loss value to attain

Software and Hardware	Configuration
CPU	Intel Core(TM) i7-7500UCPU, 2.70 GHz
RAM	8.00 GB
GPU	Intel HD Graphics 620
Operating System	Windows 10 Pro
Development Environment	Python 3.7, Jupyter Notebook

Table 2. Lab environment

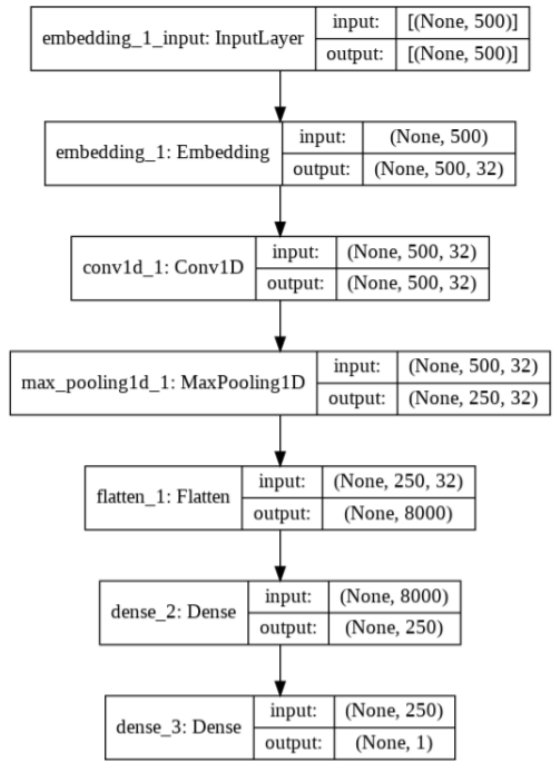

Figure 5. CNN model Layer's visualization

our final classification model. Our findings, illustrated in Figure 5, attest to the efficacy of this approach.

Regarding the CNN-LSTM model employed in our experiment, we utilized the following architecture: word embeddings vector length of 32 and a maximum review length of 500, a batch size of 60, and a convolution layer that applied a filter of 64 with a kernel size of 3 and utilized ReLU as an activation function. Additionally, the max pooling layer was set to a pool size of 2, and the dense layer used sigmoid activation. Finally, the early stopping iteration function was used to obtain the final

classification model with the loss value set to a minimum. This is demonstrated in Figure 6.

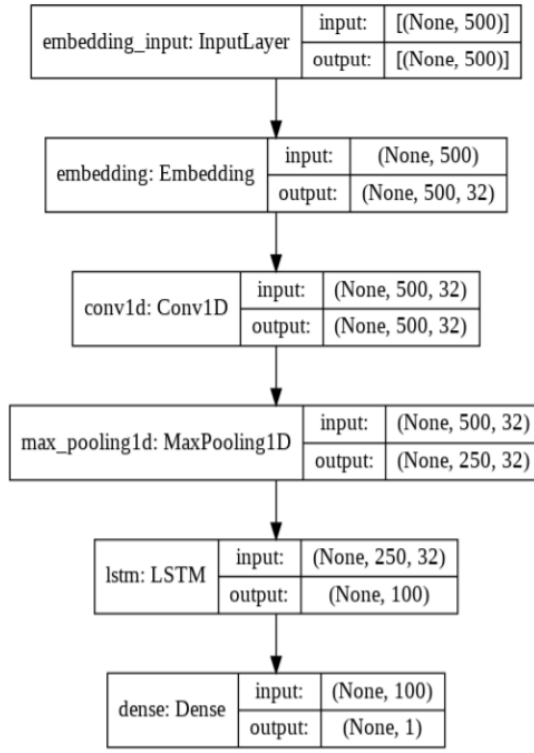

Figure 6. LSTM-CNN model Layer's visualization

The performance of the LSTM-CNN hybrid model was evaluated on both the validation and test sets, with classification accuracy, loss, and total execution time being used as metrics. The classification accuracy measures the percentage of correctly classified instances, while the loss function computes the difference between the predicted and actual values.

After conducting multiple executions of the CNN-LSTM model, accuracy was obtained, which is presented in Figure 7. However, to ensure that this accuracy was the optimal one, a random search tuning process was implemented. This involved varying the hyperparameters while keeping the architecture fixed, in order to explore new combinations. The hyper-parameters that were varied included learning rate, batch size, and the number of filters.

Figure 7 illustrates the best results achieved after numerous iterations and tuning adjustments.

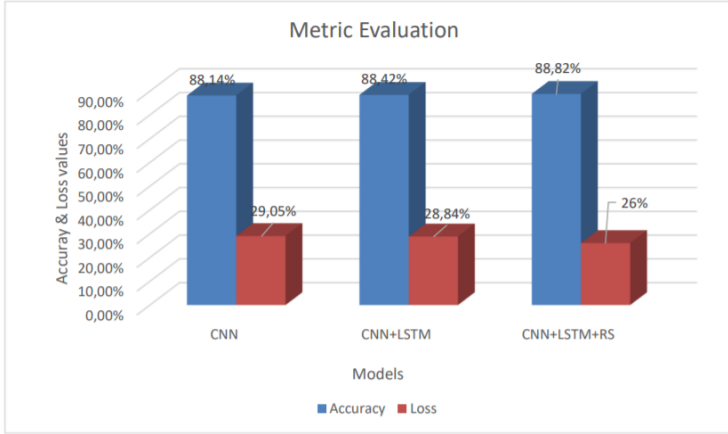

Figure 7. Metrics variation results on CNN and CNN-LSTM

It is evident that the LSTM model possesses a superior ability to capture contextual features in natural language compared to the standard CNN model when analyzing textual data.

This can be observed through the performance of the CNN-LSTM model, which surpasses the CNN model on both the test and validation sets, with an improvement of 0.28 % and 1.79 % respectively, indicating that the classification accuracy of the LSTM output is better. Consequently, it can be concluded that the performance of the CNN-LSTM model in the experiment is superior to that of the CNN model.

It is also noteworthy that the accuracy of the models was further improved by implementing random search hyper-parameter tuning, which resulted in a modest improvement of 0.40 %. Although this increase may appear insignificant, it is essential to recognize that it may be partially attributed to the fortuitous selection of hyper-parameters.

To assess the fitting appropriateness of the models, we present the training history of the CNN and CNN-LSTM models in Figures 8 and 9, respectively, which were captured by the History callback in Keras during training.

Based on the analysis of the accuracy plots, it appears that the model could benefit from further training, given that the trend for accuracy continues to rise over the last few epochs across all models.

Additionally, it can be inferred that the models have not reached a state of overfitting to the training dataset.

In contrast, the loss plot reveals that the model's performance is somewhat inconsistent across both the training and validation datasets. The parallel plots initially show a similar trend, but they begin to diverge as training progresses. This discrepancy in performance could explain why the training was stopped at an earlier epoch.

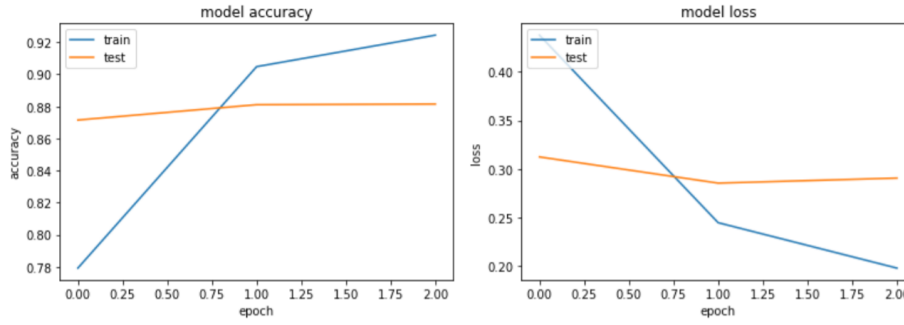

Figure 8. CNN Model training history

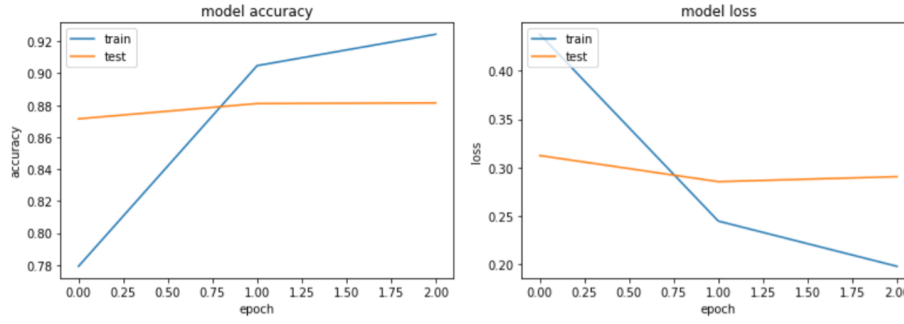

Figure 9. CNN-LSTM Model training history

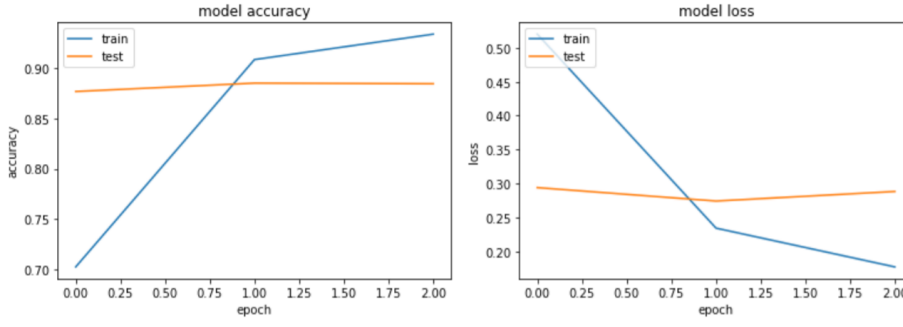

Figure 10. CNN-LSTM-RS Model training history

8 CONCLUSION

As the length and complexity of text increase, the task of text classification becomes more challenging. To address this issue, we propose a novel LSTM-CNN hybrid model for text classification. Our model combines the strengths of LSTM and CNN tasks to create a more effective deep-learning model. Our proposed model outperforms existing models in terms of accuracy and efficiency.

One advantage of our proposed model is that it can address the long-term dependency problem commonly encountered in existing models. Additionally, our model can mitigate the data loss problem, which is a common issue in traditional text classification models. To further improve our model's performance, we propose a method to tune its hyper-parameters using Random Search.

In future work, we plan to explore different architectures to incorporate into our model to further enhance its prediction performance. By continuously refining and optimizing our proposed LSTM-CNN hybrid model, we aim to provide a robust solution for text classification tasks, even for lengthy and complex texts.

REFERENCES

[1] Lakhotia, S.—Bresson, X.: An Experimental Comparison of Text Classification Techniques. 2018 International Conference on Cyberworlds (CW), 2018, pp. 58–65, doi: 10.1109/CW.2018.00022.

[2] Lewis, D. D.: Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In: Nédellec, C., Rouveirol, C. (Eds.): Machine Learning: ECML-98. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1398, 1998, pp. 4–15, doi: 10.1007/BFb0026666.

[3] Suykens, J. A. K.—Vandewalle, J.: Least Squares Support Vector Machine Classifiers. Neural Processing Letters, Vol. 9, 1999, No. 3, pp. 293–300, doi: 10.1023/A:1018628609742.

[4] Safavian, S. R.—Landgrebe, D.: A Survey of Decision Tree Classifier Methodology. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, 1991, No. 3, pp. 660–674, doi: 10.1109/21.97458.

[5] Cai, J.—Li, J.—Li, W.—Wang, J.: Deeplearning Model Used in Text Classification. 2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2018, pp. 123–126, doi: 10.1109/ICCWAMTIP.2018.8632592.

[6] Kim, Y.: Convolutional Neural Networks for Sentence Classification. CoRR, 2014, doi: 10.48550/arXiv.1408.5882.

[7] Zhou, P.—Qi, Z.—Zheng, S.—Xu, J.—Bao, H.—Xu, B.: Text Classification Improved by Integrating Bidirectional LSTM with Two-Dimensional Max Pooling. CoRR, 2016, doi: 10.48550/arXiv.1611.06639.

[8] Zhang, J.—Li, Y.—Tian, J.—Li, T.: LSTM-CNN Hybrid Model for Text Classification. 2018 IEEE 3rd Advanced Information Technology, Elec-

tronic and Automation Control Conference (IAEAC), 2018, pp. 1675–1680, doi: 10.1109/IAEAC.2018.8577620.

[9] Liang, D.—Zhang, Y.: AC-BLSTM: Asymmetric Convolutional Bidirectional LSTM Networks for Text Classification. CoRR, 2016, doi: 10.48550/arXiv.1611.01884.

[10] Zhou, C.—Sun, C.—Liu, Z.—Lau, F. C. M.: A C-LSTM Neural Network for Text Classification. CoRR, 2015, doi: 10.48550/arXiv.1511.08630.

[11] Rehman, A. U.—Malik, A. K.—Raza, B.—Ali, W.: A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis. Multimedia Tools and Applications, Vol. 78, 2019, No. 18, pp. 26597–26613, doi: 10.1007/s11042-019-07788-7.

[12] She, X.—Zhang, D.: Text Classification Based on Hybrid CNN-LSTM Hybrid Model. Vol. 2, 2018, pp. 185–189, doi: 10.1109/ISCID.2018.10144.

[13] Bergstra, J.—Bengio, Y.: Random Search for Hyper-Parameter Optimization. Vol. 13, 2012, pp. 281–305.

[14] Bergstra, J.—Bardenet, R.—Bengio, Y.—Kégl, B.: Algorithms for Hyper-Parameter Optimization. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 24 (NIPS 2011). Curran Associates, Inc., 2011, pp. 2546–2554, https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[15] Bergstra, J.—Yamins, D.—Cox, D. D.: Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In: Dasgupta, S., McAllester, D. (Eds.): Proceedings of the 30th International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research (PMLR), Vol. 28, 2013, pp. 115–123, http://proceedings.mlr.press/v28/bergstra13.pdf.

[16] Fan, X.—Runa, A.—Pei, Z.—Jiang, M.: An Improved Convolutional Neural Network for Text Classification. Journal of Physics: Conference Series, Vol. 2066, 2021, No. 1, Art. No. 012091, doi: 10.1088/1742-6596/2066/1/012091.

Chaima Ahle Touate is a doctoral candidate at the Faculty of Science and Technology, University Sultan Moulay Slimane. Specializing in artificial intelligence and text classification, she possesses a strong academic background with a Master's degree in business intelligence from the same institution. She is an active member of the Information Processing and Decision Support Laboratory TIAD. Her research interests revolve around data management, natural language processing, machine learning, and artificial intelligence, among others.

Rachid El Ayachi is an Esteemed Professor of higher education at the Faculty of Sciences and Techniques of Beni Mellal, specifically within the Computer Science Department. He has been serving in this position since 2013. He obtained his Master's degree in computer science, telecom and multimedia (ITM) from the Faculty of Sciences of Rabat, Mohammed V University, in 2006, and later went on to earn his Ph.D. in computer science from the Faculty of Sciences and Techniques of Beni Mellal, Sultan Moulay Slimane University, in 2012. Currently, he is an active member of the Information Processing and Decision Support Laboratory (TIAD). His research focuses on a wide range of areas including image processing, pattern recognition, machine learning, and natural language processing (NLP), among others.

Mohamed Biniz is a highly accomplished individual who completed his Master's degree in business intelligence in 2014 and his Ph.D. in computer science in 2018 at the Faculty of Science and Technology, University Sultan Moulay Sliman Beni Mellal. He currently holds the position of a Professor of computer science at the Polydisciplinary Faculty of the University Sultan Moulay Slimane Beni Mellal in Morocco. His research primarily centers around semantic web engineering and deep learning, with a specific interest in studying the evolution of ontology, Big Data, natural language processing, machine learning, and dynamic programming. His contributions in these fields have significantly advanced our understanding and application of cutting-edge technologies.