Original Research Paper

Boosting Arabic Named Entity Recognition with K-Fold Cross Validation on LSTM and Bi-LSTM Models

[1,2]Hamid Sadeq Mahdi Alsultani and [2]Ahmed H. Aliwy

[1]*Department of Computer Science, College of Basic Education, University of Diyala, Iraq*
[2]*Department of Computer Science, Faculty of Computer Science and Mathematics, University of Kufa, Iraq*

Article history
Received:23-03-2022
Revised: 01-07-2022
Accepted: 13-08-2022

Corresponding Author:
Hamid Sadeq Mahdi Alsultani
Department of Computer Science, College of Basic Education, University of Diyala, Iraq
Email: hamedsultani@uodiyala.edu.iq

Abstract: Named-Entity-Recognition (NER) is one of the most important Information-Extraction (IE) use cases, which is used to improve the performance of Natural Languages Processing (NLP) tasks, such as Relation-Extraction (RE), Question-Answering (QA). Recently, Arabic NER is tackled in different ways by researchers. In this study, we assess the performance of two widely used models, namely, LSTM and Bi-LSTM on the NER task in the Arabic language and perform a comparative study between these models. In contrast to the traditional data partition technique widely used during the training, we employ the technique of k-fold cross-validation to improve the performance of each model. The experimental results reveal that the performance of all models is improved when k-fold cross-validation is applied. Additionally, according to our experiment results, the Bi-LSTM model outperforms the LSTM model in terms of our evaluation metric. We achieve the best F1 score of 94.17% with CNN-Bi-LSTM-CRF. An ablation study on k-fold cross-validation demonstrates that the F1 score increased from 87.28 to 94.17%.

Keywords: Arabic Named Entity Recognition, LSTM, BiLSTM, K-Fold Cross Validation

Introduction

Arabic is one of the most commonly spoken languages and has become a United Nations (UN's) official language. Arabic is the main language of twenty-two nations and About 360 million people speak the Arabic language in more than 25 countries around the world (Ali *et al*., 2019). Arabic uses twenty-eight basic alphabets and is always written from right to left in contrast to the English language. It is highly inflected, morphologically rich, and syntactically complicated, and these factors complicate the development of NLP tools for this language (Shaalan and Oudah, 2014).

"Named Entity Recognition" (NER) is considered as a subtask of Information Extraction, that identifies and classifies textual elements according to a pre-defined set of classes called Named-Entities (NEs), which include names of people, organizations, locations, quantities, etc. (Gorla *et al*., 2020). NER is a tool used to pre-process a wide variety of applications, including "Relation Extraction", "Question Answering", "Information Retrieval", etc. (Chen *et al*., 2019).

In past decades, the Arabic NER models have been developed employing either handcrafted rules (Shaalan and Raza, 2008) or statistical learning (Benajiba and Rosso, 2007). More specifically, the handcrafted rules rely on manually handcrafted grammatical rules acquired from linguists. The drawbacks of such systems are the maintenance cost and their labor-intensive characteristics, particularly in cases where the knowledge and background of the linguists are poor (Shahina *et al*., 2019). In contrast, statistical learning based on Machine Learning (ML) relies on a training dataset to extract model patterns that are pertinent to the NER task. ML-based models do not require in-depth knowledge of the Arabic language. By providing sufficient corpus, ML-based models are adaptable and updatable with minimum cost and time (Patil *et al*., 2020).

Although their significant benefits, the traditional ML-based methods are not able to be used on large-scale Arabic datasets available on the web. Therefore, it becomes crucial to look for an alternative solution to build powerful and durable processing tools for the Arabic language. To mitigate that, neural networks have attracted researchers' attention and different deep neural network approaches have been proposed in recent years. Especially, several research works have combined supervised learning algorithms and deep neural networks to solve NLP problems such as NER, machine translation, etc. (Melis *et al*., 2017). In these deep learning approaches, researchers have employed word embedding to capture the similarities between words. Because only word embedding leads often to the problem of Out-of-Vocabulary (OoV), some research works employed both word-level embedding and character-level embedding (Shahina *et al*., 2019).

In addition to the embedding process, the encoding layer is often added on top of the embedding layer to encode the input sequence. After the encoding phase, the prediction of the entity of each word in the input sequence is performed through softmax or other classification methods. When tackling the NER problem or Part-of-Speech (PoS) tagging, it is very important to exploit the dependencies between subsequent tags and not decide on tags locally. For instance, a PER tag is more likely followed by another PER tag than by an ORG tag (Patil *et al*., 2020). Conditional Random Fields (CRF) can in exploiting the dependencies between subsequent tags. Therefore, CRF is widely used as a prediction layer in many NER models. When it comes to the encoding part, RNNs, LSTMs, and Bi-LSTMs can model the data such that the context of all words is relevant (Lee, 2017).

In most NER models, the traditional 80% for training and 20% for testing is widely used. In this study, our main contribution is to shift to the K-fold Cross Validation (K-fold CV) technique and compare various NER models to show the advantages of such a technique. The following models are compared in this study: LSTM, Bi-LSTM, LSTM-CRF, Bi-LSTM-CRF, CNN-LSTM, CNN-Bi-LSTM, CNN-LSTM-CRF, and CNN-Bi-LSTM-CRF. We show that the (K-fold CV) not only prevents the model from overfitting but it also increases the performance of all models.

Literature Review

Building strong NER models has become a crucial study field for decades, and numerous models have been tried and created to achieve substantial results. In the past decades, ANER systems have been developed using one of the following three techniques: rule-based, machine-learning-based, or hybrid.

Zaghouani (2012) presented a rule-based system called RENAR. It morphologically pre-processes data, retrieves known NEs, and extracts unknown NEs utilizing localized grammar.

Elsayed and Elghazaly (2015) developed a rule-based model to extract Arabic entities using the grammar rules of Arabic. These rules identified Arabic nouns that are not included in existing gazetteers.

Benajiba *et al*. (2008) created SVM-CRF classifiers. They evaluated the ACE dataset. Their findings indicate that CRF is not better than SVM in ANER or vice versa. Each type of NEs had a sensitivity to distinct features.

Benajiba and Rosso (2008) improved the performance of their model by replacing "Maximum Entropy" (ME) with CRF. They utilized many features in the model, such as POS-tagging, Base-Phrase-Chunks (BPC), nationalities as well as gazetteers. The model measures were: Recall, precision, and F-Score with 72.77, 86.90, and 79.21% respectively.

AbdelRahman *et al*. (2010) integrated two ML models to handle Arabic-NER utilizing CRF, plus bootstrapping. Many features were implemented such as Word features, morphological features, POS tagging, and gazetteers. The model detected several NEs (Persons, Locations, Devices, Cell-Phones, Organizations, Cars, Dates, and Times).

Elarnaoty *et al*. (2012) offered another notable machine learning-based study. The ANER challenge was solved by identifying ten NE classes. CRF and bootstrapping were seamlessly combined to improve performance.

Mohammed and Omar (2012) built an ANER model utilizing neural networks. Two approaches were used to extract persons, organizations, places, and miscellaneous named entities. The experiment compared Decision-Tree (DT) with Neural Network (NN) on the exact dataset. The used measure was the precision with 92% for the NN and 87% for the DT.

Dahan *et al*. (2015) suggested a "Hidden Markov Model" HMM-based ANER. The model addressed Arabic inflection and ambiguity using stemming. Their NE-recognition model was entirely automated and tested using "Al -Hayat newspaper".

Al-Shoukry and Omar (2015) suggested a Decision Tree-based ML model. Their technology can extract NEs of persons, places, criminal activities, times, as well as the date. The F-score for the model was 81.35%.

Alsayadi and ElKorany (2016) used machine learning to propose a model for ANER that was semantically integrated. Multiple language features and syntax dependencies were used by the model to predict how named entities are semantically related. The model assisted in overcoming some of Arabic's orthographic as well as morphological limitations.

Recently, deep learning techniques have demonstrated their efficiency in several tasks from computer vision to healthcare. On NLP tasks, including NER, deep learning outperforms handcrafted features with a large margin. Therefore, current NER systems employ deep learning. For instance, Helwe and Elbassuoni (2019) proposed a revolutionary deep co-learning approach for recognizing and classifying NEs. This approach uses only word embedding but no other designed features. This approach surpassed other Arabic approaches, including ones that incorporate well-engineered NLP features. It also surpassed many supervised and semi-supervised approaches in deep learning.

An encoder-decoder Arabic NER model was proposed by Ali and Tan (2019). More specifically, they used an attention layer to concatenate the characters embedding and word embedding at the first layer. The output of this layer is fed to a Bi-LSTM for encoding. Then, another attention is added on top of the encoder and the output of this attention layer is given to the decoder which is another Bi-LSTM. On ANERCorp and AQMAR datasets, the authors demonstrated significant improvement in their model.

For Arabic NER, El Bazi and Laachfoubi (2019) used a BiLSTM-CRF neural network model. The total efficiency of the model was evaluated using a different set of hyperparameters that are commonly used. Both character and word embeddings are used to give word information to the model, removing the need for designed features.

Al-Smadi *et al*. (2020) proposed a transfer-learning NER model for Arabic. They used a pre-trained Universal Sentence Encoder (USE) for embedding and fed the embedding output to a Bi-GRU. Then, the authors applied a Global-Average-Pooling (GAP) and a Global-Max-Pooling (GMP). The results of these operations are concatenated and fed to a feed-forward network for prediction. They obtained a 91.20% of F1-score on the WikiFANE_Gold dataset.

Materials and Methods

The task of NER is used to identify each token in the sequence and to assign it a suitable label. Many models have been used to efficiently predict the correct labels for tokens. In this section, some models that are commonly used in the task of NER in general and in ANER, in particular, are mentioned and briefly explained. These models are as follows:

Long Short-Term Memory (LSTM)

Scientists created the LSTM to handle the long-term dependencies and overcome the issues of vanishing gradients and exploding gradients that exist in the traditional RNN (Thomas and Sangeetha, 2020). LSTM memory cells substituted previously hidden layer updates in RNN. A self-recurrent neuron (SRN) as well as three gates (input, forget, and output) make up an LSTM memory cell (Kompalli *et al*., (2021). The SRN ensures that the memory cell state is maintained over time. The input gate controls the effects of the input signal on the cell's state, whereas the output gate controls the effects of the cell's state on the further neurons. Finally, the forget-gate manages the SRN (Fan *et al*., 2020). The basic design of the LSTM cell is shown in Fig. 1.

Figure 2 shows an LSTM model for ANER that uses the above-mentioned LSTM's memory cells (grey-colored boxes).

CNN-LSTM

Convolutional Neural Networks (CNNs) were firstly used in computer vision for pattern recognition. CNN has several layers, an input layer, hidden layers, and an output layer. Hidden layers include convolutional-layer, pooling-layer, fully-connected-layer, and normalizing-layer (Thomas and Sangeetha, 2020). Lastly, CNN includes a SoftMax layer which is used to get output classes. In NLP, CNN is often used to obtain character-level features that can be employed in NER tasks (Maslej-Krešňáková *et al*., 2020). The basic design of CNN is shown in Fig. 3.

A CNN-LSTM model is formed by combining a CNN-layer with an LSTM-layer as in Fig. 4. The purpose of using the CNN-layer (blue colored boxes) is to get character-level features from the input data.

LSTM-CRF

An LSTM-CRF model is formed by combining an LSTM-layer with a CRF-layer as in Fig. 5. In this model, previous input features through an LSTM-layer as well as label information at the sentence level through a CRF-layer can be used efficiently. A CRF-layer is denoted by green dashed lines that connect successive outputs. A CRF layer uses parameters called the state-transition matrix. Through a CRF layer, it is efficient to predict the present label because previous and future labels can be used to do this task, and this is the same as using previous and future input features through a BiLSTM model.

CNN-LSTM-CRF

A CNN-LSTM-CRF model is formed by combining a CNN-layer with an LSTM-layer and a CRF-layer as in Fig. 6. A CRF-layer is denoted by green dashed lines that connect successive outputs.

Bidirectional Long Short-Term Memory (BiLSTM)

Sequence labeling can be made more efficient by providing both future and previous input context for a certain period. An integrated LSTM in two directions (forward and backward) solves the problem of only having previous inputs for all the hidden states (Manur *et al*., 2020). Every time a sequence is given, it is checked in the forward direction (left to right), and in the backward direction (right to left) once more (Cai *et al*.,2021). The basic design of BiLSTM is shown in Fig. 7.

Figure 8 shows a BiLSTM model for ANER that uses both forward and backward LSTM layers (grey-colored boxes).

CNN-BiLSTM

A CNN-BiLSTM model is formed by combining a CNN layer with a BiLSTM-layer as in Fig. 9.

BiLSTM-CRF

A BiLSTM-CRF model is formed by combining a BiLSTM-layer with a CRF-layer as in Fig. 10. This model uses the future input features as well as the previous input features and label information at the sentence level that is used in the LSTM-CRF model.

CNN-BiLSTM-CRF

A CNN-BiLSTM-CRF model is formed by combining a CNN-layer with a BiLSTM-layer and a CRF-layer as in Fig. 11.

Experiments

Dataset

All models are trained, validated, and tested using a public Arabic dataset called (WikiFANE_Selective). The dataset has (57126) sentences and (2,021,177) tokens. It also has (9) main labels: FAC (Facility), GPE (Geopolitical), LOC (Location), ORG (Organization), PER (Person), PRO (Product), VEH (Vehicle), WEA (Weapon), and O (Other). The annotation standard used for the ANER task was (BIO) where (B, I, and O) mean (Begin, Inside, and Other) respectively. The annotation standard was used for all labels except the (O) label. The link for the dataset is: (https://fsalotaibi.kau.edu.sa/Pages-Arabic-NE-Corpora.aspx).

Metrics

The F1 score was the main metric used to evaluate the performance of the used models. This metric depends on two other metrics: Precision (P) and recall (R). The formulas for precision, recall, and F1-score are given in Eq. 1, 2, and 3.

$$P = \frac{TP}{TP + FP} * 100 \tag{1}$$

$$R = \frac{TP}{TP + FP} * 100 \tag{2}$$

$$F1 = \frac{2 * P * R}{P + R} * 100 \tag{3}$$

where, (*TP*), (*FP*), and (*FN*) denote true-positive, false-positive, and false-negative, respectively.

Pre-Processing

The Arabic language consists of (28) basic alphabets plus "hamza" (ء), "alif maqsoora" (ى), and "taa Marboota" (ـة). Some alphabets are written in many forms, for example, "alif" (ا) may be written as follows: (ا, آ, أ, إ). Also "hamza" may be written as follows: (ء, ؤ, ئ). To deal with these various forms fast and easily, some pre-processing may be applied using two main processes: Normalization and transliteration.

Normalization

In this process, some rules are applied as follows:

1- Normalize "*alif*": آ, أ, إ → ا
2- Normalize "*hamza*": ؤ, ئ → ء

Transliteration

In this process, each Arabic alphabet is converted to its equivalent alphabet in English. This process is important when there is a mix of Arabic and English alphabets in a dataset or text. In this study, a special transliteration was used. The applied rules are as follows:

1- "*hamza*": ء --------- i
2- "*alif*": ا --------- a
3- "*baa*": ب --------- b
4- "*taa*": ت --------- t
5- " *taa marboota* ": ـة --------- p
6- "*thaa*": ث --------- P
7- "*jeem*": ج --------- j
8- "*Haa*": ح --------- H
9- "*Khaa*": خ --------- K
10- "*daal*": د --------- d
11- "*thal*": ذ --------- V
12- "*raa*": ر --------- r
13- "*zaay*": ز --------- z
14- "*seen*": س --------- c
15- "*sheen*": ش --------- C
16- "*saad*": ص --------- s
17- "*dhaad*": ض --------- S
18- "*Taa*": ط --------- T
19- "*Zaa*": ظ --------- Z
20- "*ain*": ع --------- E
21- "*ghain*": غ --------- g
22- "*faa*": ف --------- f
23- "*qaaf*": ق --------- q
24- "*kaaf*": ك --------- k
25- "*laam*": ل --------- l
26- "*meem*": م --------- m
27- "*noon*": ن --------- n
28- "*haa*": ه --------- h
29- "*waaw*": و --------- w
30- "*yaa*": ي --------- y
31- "*alif* maqsoora": ى --------- Y

Tables 1 and 2 show examples of the normalization and the transliteration processes on the used dataset respectively:

Table 1: Examples of the normalization process

Words in English	Words in Arabic before normalization	Words in Arabic after normalization
Song	أغنية	اغنية
Spain	إسبانا	اسبانيا
Last	آخر	اخر
Responsible	مسؤول	مسءول
Disadvantages	مساوئ	مساوء

Table 2: Examples of the transliteration process

Words in English	Words in Arabic before Transliteration	Words after Transliteration
Ancient	القديم	alqdym
Program	برنامج	brnamj
Company	شركة	Crkp
When	عندما	Endma
Photography	فوتوغرافيا	fwtwgrafya
Principles	مبادء	mbadi

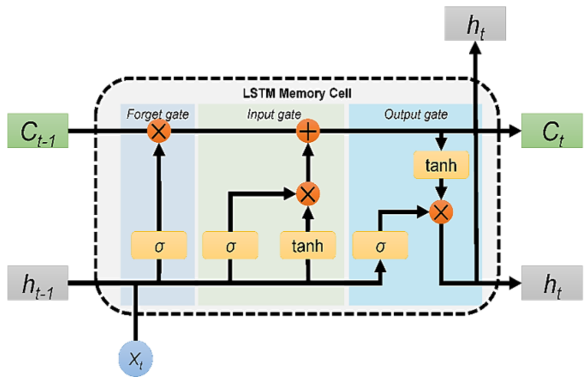

Fig. 1: The basic design of LSTM cell (Fan *et al*., 2020)

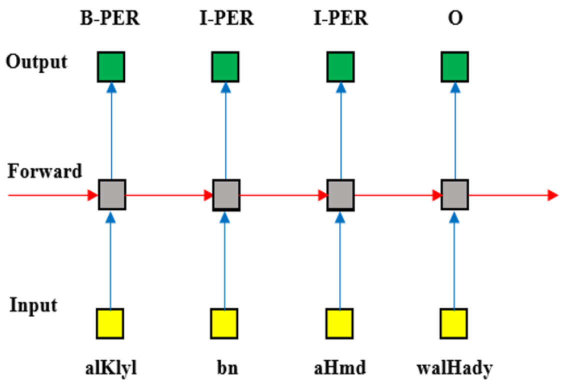

Fig. 2: An LSTM model for ANER

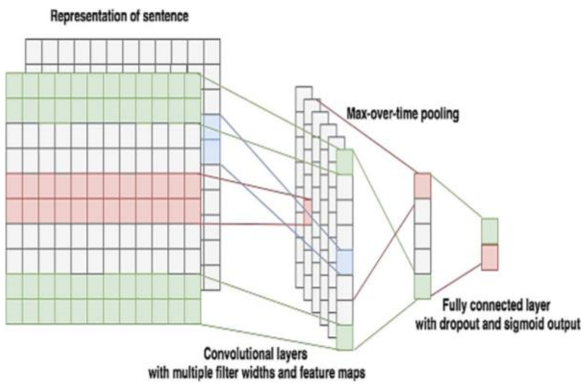
Representation of sentence
Max-over-time pooling
Fully connected layer with dropout and sigmoid output
Convolutional layers with multiple filter widths and feature maps

Fig. 3: The basic design of CNN [28]

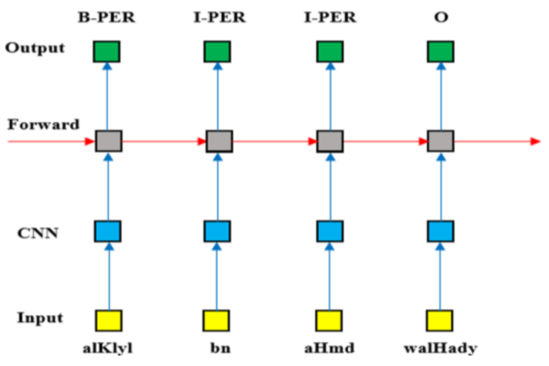

Fig. 4: A CNN-LSTM model for ANER

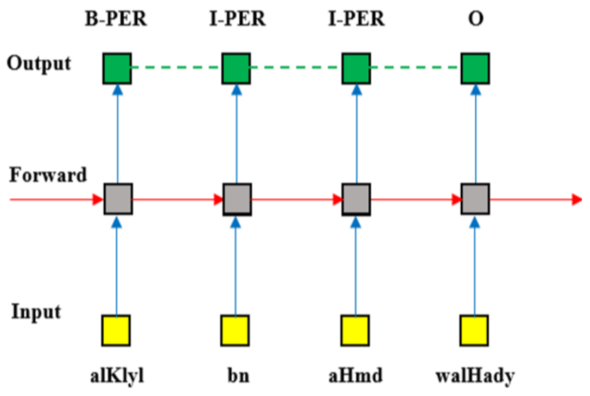

Fig. 5: An LSTM-CRF model for ANERc

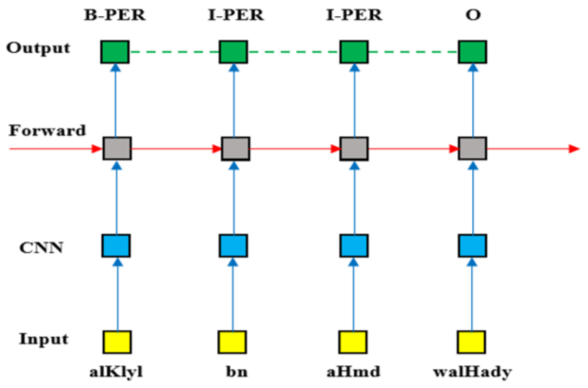

Fig. 6: A CNN-LSTM-CRF model for ANER

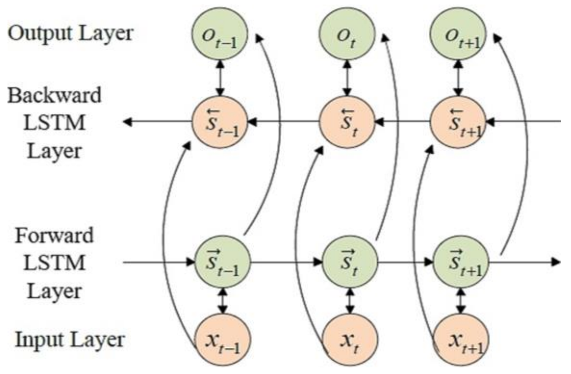

Fig. 7: The basic design of BiLSTM (Cai *et al*., 2021)

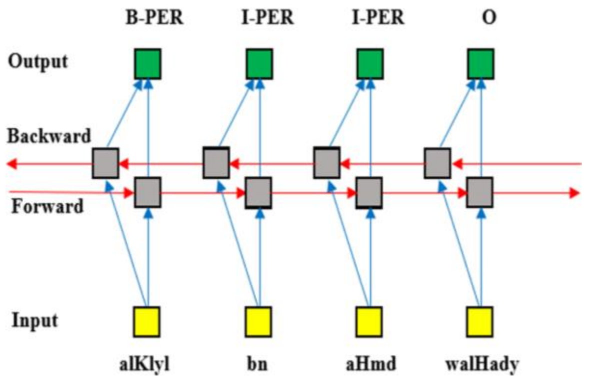

Fig. 8: A BiLSTM model for ANER

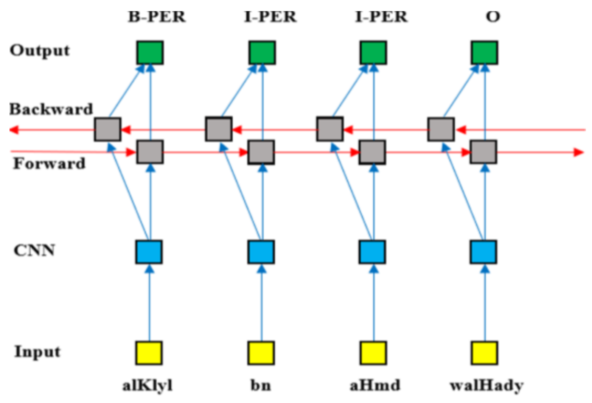

Fig. 9: A CNN-BiLSTM model for ANER

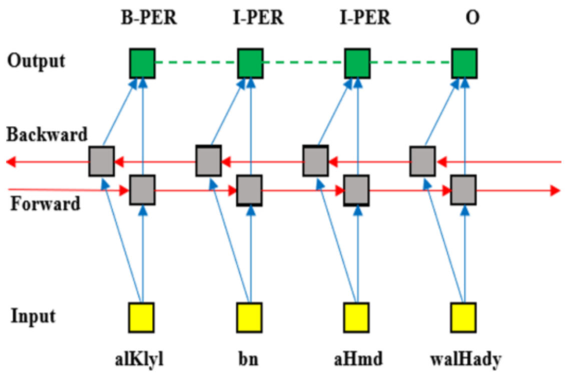

Fig. 10: A BiLSTM-CRF model for ANER

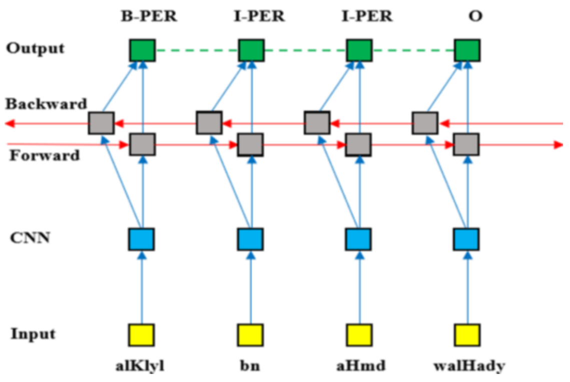

Fig. 11: A CNN-BiLSTM-CRF model for ANER

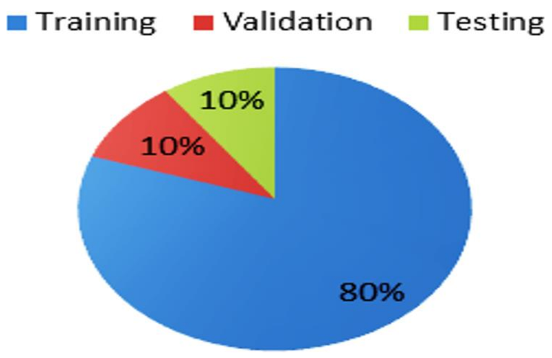

Fig. 12: Traditional partitioning

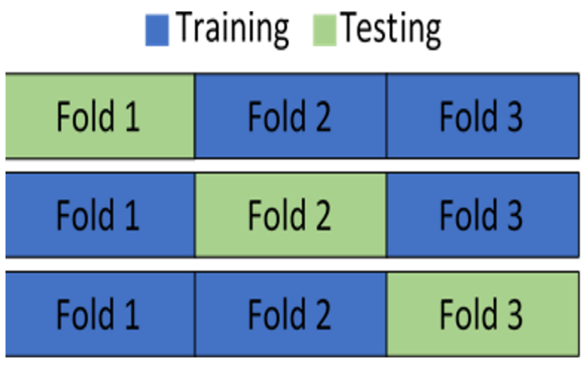

Fig. 13: K-fold CV (k=3)

Dataset Partitioning

In this study, two types of partitioning were applied to the dataset: Traditional partitioning and partitioning by k-fold cross validation.

Traditional Partitioning

The dataset is partitioned by a specific ratio into three parts: Training, validation, and testing. In this study, the dataset was partitioned by 80, 10, and 10% for training, validation, and testing, respectively, as shown in Fig. 12.

Partitioning by K-Fold Cross Validation

The "K-fold cross-validation" or "K-fold CV" technique can be used to evaluate learning models (Jiang and Wang, 2017). With this technique, the dataset can be partitioned into k-folds. Each iteration uses one-fold for testing and the rest for training. Thus, this procedure repeats until the entire dataset has been evaluated (Nti *et al*., 2021). The final model scores are calculated by averaging the scores obtained from all iterations (Yadav and Shukla, 2016). In this study, the number of folds (k) was (3) as shown in Fig. 13.

Results and Discussion

Since no previous paper used the WikiFane_Selective dataset for the task of ANER to use it for comparison, two baseline models were used and improved in this study: LSTM and BiLSTM. The F1 scores of the baseline models and their improvements are presented in Table 3. Note that each model in the table has two F1 scores, the first one when the traditional partitioning is used and the second when the partitioning by k-fold CV is used.

The LSTM baseline model achieves the lowest F1 scores (68.94) and (81.27). Adding the CNN layer to the LSTM makes the model run better when it uses character features. The F1 scores become (71.08) and

(84.68). If a CRF layer is added to the LSTM, the F1 scores will be (80.87) and (92.34). While adding both the CNN-layer and the CRF-layer to the LSTM makes the model get the highest F1 scores (84.69) and (92.46). The results of the LSTM variations are shown in Fig. 14.

For the BiLSTM baseline model, the initial F1 scores are (79.44) and (88.65). After adding the CNN layer, the F1 scores become (81.89) and (90.34). If a CRF-layer is added the F1 scores will be (83.61) and (94.06). Finally, adding both the CNN-layer and the CRF-layer raises F1 scores to (87.28) and (94.17). The results of the BiLSTM variations are shown in Fig. 15.

A close look at Fig. 16 shows many important issues. First, the BiLSTM models always overcome their analogs in the LSTM models when using the CNN layer, the CRF layer, or both layers. Second, using the CNN-layer (character features) gives a minor improvement for both the LSTM and BiLSTM baseline models when used with ANER for the used dataset (WikiFane_Selective). Third, when the CRF-layer is used, both the LSTM and CNN-LSTM models get a big boost from it. The BiLSTM and CNN-BiLSTM models, on the other hand, get a small boost from them. Finally, the results of the partitioning by K-fold CV are always better than the results of the traditional partitioning for all models.

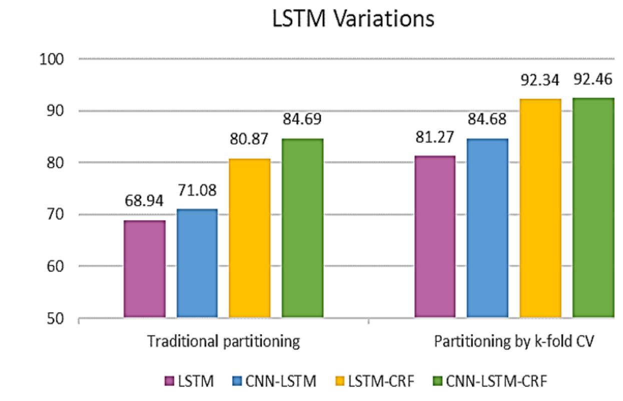

Fig. 14: Results of the LSTM variations

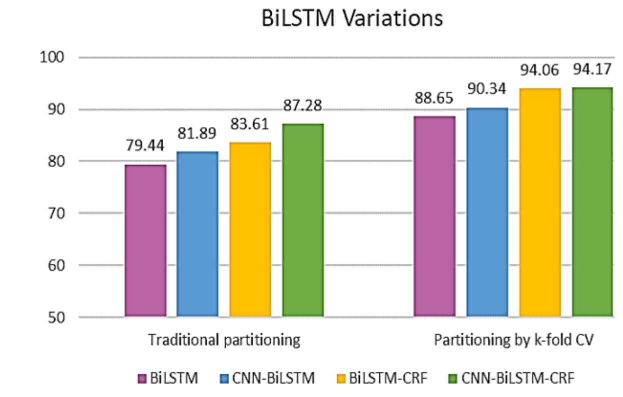

Fig. 15: Results of the BiLSTM variations

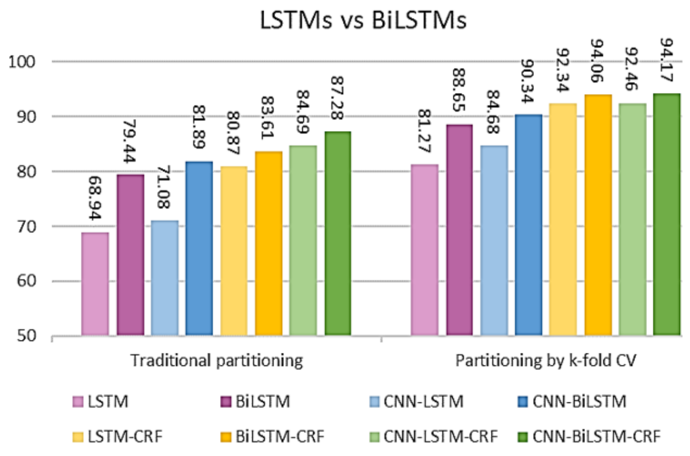

Fig. 16: Comparison of the LSTM and BiLSTM models

Table 3: F1-scores of different ANER models using traditional partitioning and partitioning by k-fold CV

Models	Traditional partitioning	Partitioning by k-fold CV
LSTM	68.94	81.27
CNN-LSTM	71.08	84.68
LSTM-CRF	80.87	92.34
CNN-LSTM-CRF	84.69	92.46
BiLSTM	79.44	88.65
CNN-BiLSTM	81.89	90.34
BiLSTM-CRF	83.61	94.06
CNN-BiLSTM-CRF	87.28	94.17

Conclusion and Future Works

This study presented a comparison among variants of the LSTM and BiLSTM baseline models using a public Arabic dataset to extract Arabic-named entities. Each baseline model was combined with either a CNN layer, a CRF layer, or both layers. Two types of dataset partitioning were used: The traditional portioning and the partitioning by k-fold CV. The BiLSTM models got better results if they were compared with their analog LSTM models in both types of dataset partitioning. Also, dataset partitioning by the K-fold CV is more useful than traditional partitioning. The best F1 score when using the K-fold CV was for the CNN-BiLSTM-CRF model, with a score of 94.17. Based on the results and what was mentioned above, it is concluded that using the K-fold CV in dataset partitioning and combining the CRF layer with both the BiLSTM and CNN-BiLSTM models will get the best results.

In the future, we aim to apply K-fold CV to advanced architectures, like sequence-2-sequence models or architecture that includes attention mechanisms.

Acknowledgment

This research paper was not funded by any institution.

Author's Contributions

Hamid Sadeq Mahdi Alsultani: The paper background work, conceptualization, methodology, dataset pre-processing, implementation, result analysis and comparison, preparing and editing draft, and visualization.

Ahmed H. Aliwy: The supervision, review of work and project administration.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

AbdelRahman, S., Elarnaoty, M., Magdy, M., & Fahmy, A. (2010). Integrated machine learning techniques for Arabic named entity recognition. *IJCSI*, 7(4), 27-36.

Ali, M. N. A., Tan, G., & Hussain, A. (2019). Boosting Arabic named-entity recognition with multi-attention layer. *IEEE Access*, 7, 46575-46582. https://doi.org/10.1109/ACCESS.2019.2909641

Ali, M. N., & Tan, G. (2019). Bidirectional encoder-decoder model for Arabic named entity recognition. *Arabian Journal for Science and Engineering*, 44(11), 9693-9701. https://doi.org/10.1007/s13369-019-04068-2

Alsayadi, H. A., & ElKorany, A. M. (2016). Integrating semantic features for enhancing Arabic named entity recognition. International *Journal OF Advanced Computer Science And Applications*, 7(3).

Al-Shoukry, S., & Omar, N. (2015). Proper noun recognition in Arabic crime text using machine learning approach. *Journal of Theoretical & Applied Information Technology*, *79*(3).

Al-Smadi, M., Al-Zboon, S., Jararweh, Y., & Juola, P. (2020). Transfer learning for Arabic named entity recognition with deep neural networks. *IEEE Access*, *8*, 37736-37745. https://doi.org/10.1109/ACCESS.2020.2973319

Benajiba, Y., & Rosso, P. (2007, December). ANERsys 2.0: Conquering the NER task for the Arabic language by combining the maximum entropy with POS-tag information. In *IICAI* (pp. 1814-1823).

Benajiba, Y., & Rosso, P. (2008, May). Arabic named entity recognition using conditional random fields. In *Proc. of Workshop on HLT & NLP within the Arabic World, LREC* (Vol. 8, pp. 143-153).

Benajiba, Y., Diab, M., & Rosso, P. (2008). Arabic named entity recognition: An SVM-based approach. In *Proceedings of 2008 Arab International Conference on Information Technology (ACIT)* (pp. 16-18). Amman, Jordan: Association of Arab Universities.

Cai, C., Tao, Y., Zhu, T., & Deng, Z. (2021). Short-Term Load Forecasting Based on Deep Learning Bidirectional LSTM Neural Network. *Applied Sciences*, *11*(17), 8129. https://doi.org/10.3390/app11178129

Chen, H., Lin, Z., Ding, G., Lou, J., Zhang, Y., & Karlsson, B. (2019, July). GRN: Gated relation network to enhance convolutional neural network for named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 6236-6243). https://doi.org/10.1609/aaai.v33i01.33016236

Dahan, F., Touir, A., & Mathkour, H. (2015). First-order hidden Markov model for automatic Arabic name entity recognition. *International Journal of Computer Applications*, *123*(7).

El Bazi, I., & Laachfoubi, N. (2019). Arabic named entity recognition using a deep learning approach. *International Journal of Electrical & Computer Engineering (2088-8708)*, *9*(3). https://10.11591/IJECE.V9I3.PP2025-2032

Elarnaoty, M., AbdelRahman, S., & Fahmy, A. (2012). A machine learning approach for opinion holder extraction in the Arabic language. *ARXIV preprint arXiv:1206.1011*. https://doi.org/10.48550/arXiv.1206.1011

Elsayed, H., & Elghazaly, T. (2015, April). A named entities recognition system for modern standard Arabic using a rule-based approach. In *2015 First International Conference on Arabic Computational Linguistics (ACLing)* (pp. 51-54). IEEE. https://doi.org/10.1109/ACLing.2015.14

Fan, H., Jiang, M., Xu, L., Zhu, H., Cheng, J., & Jiang, J. (2020). Comparison of long short-term memory networks and the hydrological model in runoff simulation. *Water*, *12*(1), 175.

Gorla, S., Neti, L. B. M., & Malapati, A. (2020). Enhancing the performance of Telugu named entity recognition using Gazetteer features. *Information*, *11*(2), 82. https://10.3390/info11020082

Helwe, C., & Elbassuoni, S. (2019). Arabic named entity recognition via deep co-learning. *Artificial Intelligence Review*, *52*(1), 197-215. https://doi.org/10.1007/s10462-019-09688-6

Jiang, G., & Wang, W. (2017). Error estimation based on variance analysis of k-fold cross-validation. *Pattern Recognition*, *69*, 94-106. https://doi.org/10.1016/j.patcog.2017.03.025

Kompalli, N. S. J., & Murthy, D. S. R. C (2021). Elite Firefly and Long Short Term Memory Based Model for Texture Classification.

Lee, C. (2017). LSTM-CRF models for named entity recognition. *IEICE Transactions on Information and Systems*, *100*(4), 882-887. https://doi.org/10.1587/transinf.2016EDP7179

Manur, M., Pani, A. K., & Kumar, P. (2020). A prediction technique for heart disease based on long short-term memory recurrent neural network. *International Journal of Intelligent Engineering and Systems*, *13*(2), 31-33.

Maslej-Krešňáková, V., Sarnovský, M., Butka, P., & Machová, K. (2020). Comparison of deep learning models and various text pre-processing techniques for the toxic comments classification. *Applied Sciences*, *10*(23), 8631. https://doi.org/10.3390/app10238631

Melis, G., Dyer, C., & Blunsom, P. (2017). On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*. https://doi.org/10.48550/arXiv.1707.05589

Mohammed, N. F., & Omar, N. (2012). Arabic named entity recognition using artificial neural network. *Journal of Computer Science*, *8*(8), 1285.

Nti, I. K., Nyarko-Boateng, O., & Aning, J. (2021). Performance of Machine Learning Algorithms with Different K Values in K-fold Cross-Validation. *Inter. J. Info. Technol. Comp. Sci.*, *13*, 61-71. https://10.5815/ijitcs.2021.06.05

Patil, N., Patil, A., & Pawar, B. V. (2020). Named entity recognition using conditional random fields. *Procedia Computer Science*, *167*, 1181-1188.

Shaalan, K., & Oudah, M. (2014). A hybrid approach to Arabic named entity recognition. *Journal of Information Science*, *40*(1), 67-87. https://doi.org/10.1177/0165551513502417

Shaalan, K., & Raza, H. (2008, August). Arabic named entity recognition from diverse text types. In *International Conference on Natural Language Processing* (pp. 440-451). Springer, Berlin, Heidelberg.

Shahina, K. K., Jyothsna, P. V., Prabha, G., Premjith, B., & Soman, K. P. (2019, March). A sequential labeling approach for the named entity recognition in the Arabic language using deep learning algorithms. In *2019 International Conference on Data Science and Communication (IconDSC)* (pp. 1-6). IEEE. https://doi.org/10.1109/IconDSC.2019.8817039

Thomas, A., & Sangeetha, S. (2020). Deep learning architectures for named entity recognition: A survey. In *Advanced Computing and Intelligent Engineering* (pp. 215-225). Springer, Singapore. https://doi.org/10.1007/978-981-15-1081-6_18

Yadav, S., & Shukla, S. (2016, February). Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In *2016 IEEE 6th International Conference on Advanced Computing (IACC)* (pp. 78-83). IEEE. https://doi.org/10.1109/IACC.2016.25

Zaghouani, W. (2012). RENAR: A rule-based Arabic named entity recognition system. *ACM Transactions on Asian Language Information Processing (TALIP)*, *11*(1), 1-13. https://doi.org/10.1145/2090176.2090178