Automatic Configuration of Deep Learning Algorithms for an Arabic Named Entity Recognition System

AZROUMAHLI Chaimae[1], MOUHIB Ibtihal[2], El YOUNOUSSI Yacine[3], BADIR Hassan[4]
Laboratory of Intelligent Systems and Applications (LSIA),
Moroccan School of Engineering Sciences (EMSI), Tangier, Morrocco[1, 2]
SIGL Laboratory, ENSA Tetuan, Abdel Malek Essaadi University, Tetuan, Morrocco[3]
IDS-Team, ENSA Tangier, Abdel Malek Essaadi University, Tangier, Morrocco[4]

***Abstract*—Word embedding models have been widely used by many researchers to extract linguistic features for Natural Language Processing (NLP) tasks. However, the creation of an adequate Word embedding model depends on choosing the right language model method and architecture, in addition to fine-tuning the various parameters of the language model. Each parameter combination could result in a different model, and each model can behave differently according to the targeted NLP task. In this paper, we present an approach that combines a range of Word embedding models, multiple clustering and classification methods, and Irace for automatic algorithm configuration. The goal is to facilitate the construction of the most accurate Arabic Named Entity Recognition (NER) model for our dataset. Our approach involves the creation of different Word embedding models, the implementation of these models in different classification and clustering methods, and fine-tuning these implementations with different parameter combinations to create an Arabic NER System with the highest accuracy rate.**

Keywords*—Algorithm automatic configuration; natural language processing; named entity recognition; word embeddings; finetuning; irace

I. Introduction

For NLP applications like machine translation, information retrieval and sentiment analysis, it is crucial to have high-quality systems for lower tasks that return necessary features for machine learning systems [1] [2]. NER is an essential component for such tasks, its most important aspect is information extraction, and it can be carried out in two steps; the detection of the Named Entities, and the classification of these entities into a predefined set of categories (e.g., organizations, places, people, …). The term "Named Entity" was introduced during the sixth Message understanding conference [3]. The NER task was limited to the recognition of the people's names, organizations, places, temporal expressions and certain types of numerical expression [4]. These classification tags were divided afterwards into these categories: ENAMEX for people names, organizations and places, TIMEX for temporal expressions, NUMEX for numerical expression, and MISC for proper names that are not in the ENAMEX category.

NER systems utilize several linguistic features, in fact detecting these features is considered more important than the used model itself, especially when handling languages with a complex morphology like Arabic [5]. Lately, there has been a hype on using unlabeled data to learn word representation or Word Embeddings that can capture morphological, semantic and syntactic features of words, which consequently can be helpful in many learning algorithms of NLP including NER. However, there are various methods for learning Word Embeddings (e.g. Word2Vec [6], GloVe [7], FastText [8], BERT [9], ELMO [10]), and each method has many parameters that can be adjusted to create different models. Further, the machine learning algorithm that will use these Word embedding models can also have a major effect on the performance of the resulting NER system. In addition, for each model and machine learning algorithm, several training parameters can be tuned and adjusted to get more accurate results.

In this work, we adopt Irace [11] as a finetuning tool to find the most accurate NER system for our dataset. Our dataset includes two Arabic varieties: Modern Standard Arabic (MSA) and Arabic Dialects (AD) [12]. The Objective is to choose automatically one Word Embedding model from several models created using four methods (i.e., Skip-Gram, CBOW, GloVe, FastText) and one machine learning algorithm with its suitable hyperparameter combination. We use Irace to fine-tune between different created Arabic Word Embeddings, and different classification algorithms (i.e., LSTM [13], GRU [14]) and clustering algorithms (i.e., K-Mean [15], Mean-shift [16], DBSCAN [17], and Agglomerative [18]) to get the most accurate system possible for the NER task.

The rest of this paper follows this structure: Section II provides an overview of the key concepts and introduces the Irace package, a crucial component of the proposed methodology. Section III outlines the step-by-step process employed to develop the NER system. Section IV presents and analyzes the results obtained from the experiments. Finally, Section 5 concludes the work described in this paper.

II. The Proposed Method

In prior work [19], we have created word Embedding models with four methods (i.e. Skip-Gram, CBOW, Glove, FastText). These models were trained on three different datasets (i.e., Wikipedia, Facebook, Twitter) containing two

different Arabic varieties (i.e., MSA and Arabic Dialects). Then, we investigated the quality of the trained models using 12 hyperparameter combinations. We used different intrinsic evaluations (i.e., Word Analogy Task, Concept categorization) and different extrinsic evaluations (i.e., POS tagging, NER, Text Classification, Sentiment Analysis). In conclusion, our study raised three outcomes; The different stylistic properties of the datasets and the tuned hyper-parameters had an impact on the semantic and syntactic properties of the generated word representations and, subsequently, an impact on the NLP tasks. Further, even though the hyperparameters have a major impact on the accuracy of different NLP tasks, these changes are inconsistent and random [19].

As a consequence, we propose the solution of fine-tuning between the different hyperparameters to get the most accurate system for a specific application. Nevertheless, fine-tuning between 12 combinations to create the Word Embeddings model, and the different classification or clustering methods will be computationally expensive and time-consuming. Thus, we needed an approach that would automatically fine-tune between these hyperparameters and deduce the most accurate hyperparameter combination for a specific NLP task.

In this section, we present a detailed description of the tools that we used, afterwards in Section 3, we explain in depth the algorithms of our approach.

A. The Irace Package

In NLP, creating an efficient system is relevant to the selection and fine-tuning of the training algorithms' parameters. In Machine Learning, this is known as the automatic algorithm's configuration. The goal is to find beneficial parameter settings to solve unseen problem instances by trying automatic learning on a set of training problem instances [11]. We opted to use the Irace package to facilitate the combined use of Word Embeddings' models and machine learning methods. Irace executes an automatic tuning of a set of parameter combinations, consequently, we avoid the manual adjusting of these parameters.

The Irace package is an R software tool that implements iterated racing procedures. It was created for the automatic configuration of optimization and decision algorithms, thus, its goal is to find the most accurate settings of an algorithm where a set of probabilities' instances is given [20],[21]. This package is suitable for our application since it can automatically configure the training algorithm where their performances depend greatly on parameter settings.

Irace deduces the most accurate algorithm configuration by implementing an elite principle on the iterated racing algorithm [20]; In the first iteration, initial algorithm configurations are randomly generated, and the best configuration is determined by a race [11]. Each configuration is evaluated on the training problem instances to set the "elite" configurations from the prior configuration iterations. Afterwards, a statistical test is used to determine the eliminated configurations once they perform worse than the other configurations. The remaining configurations will be known as the surviving configurations that will run on the next instance.

B. The Word Embeddings Models

We opted for several Word Embeddings models using different architectures and different hyper-parameters, Table I shows the Dataset's sources and the hyperparameters used to train the models for every architecture. In prior work [19], we used a Python implementation of Word2Vec to create Skip-Gram and CBOW Word Embeddings models using Word2Vec's both architectures (i.e., Hierarchical SoftMax - HS) and Negative Sampling (NS) [22])[1]. The models were trained on four different training Arabic Datasets containing both MSA and Arabic Dialects content. We used the Glove-Python implementation to create Glove Word Embeddings models. We opted to use pre-trained word vectors created by the Facebook Artificial Intelligence Research team using Fast Text's CBOW and Skip-Gram architectures [23].

The used datasets were collected and pre-processed in prior work from different sources [24]. The first source is the online encyclopedia; Wikipedia. This corpus presents the two varieties Classical and Modern Standard Arabic. The second source is social media; Twitter and Facebook. This corpus presents the various Arabic Dialects content. The datasets were pre-processed afterwards; non-Arabic characters and diacritical marks were removed, several characters were normalized to unify the shape of some Arabic letters, and several Arabic stop words were disregarded as well [24].

C. Machine Learning Methods

Different works have shown that the combined use of both the supervised and the unsupervised methods has a positive impact on the performance of several NLP applications [25], [26],[27]. These methods are called semi-supervised methods, where labelled and unlabeled data are used to perform certain learning tasks [28]. In our case, we harnessed large amounts of unlabeled Arabic text data and created word representations that have the potential of carrying semantic and syntactic word properties as explained in the previous section, we used relatively smaller sets of labelled data to perform the NER task as it will be explained in the next section. There is a wide range of classification and clustering algorithms that can be used as the supervised part of our application, and since we can automatically configure and fine-tune between different decision algorithms using Irace, we can utilize several algorithms. For the clustering algorithms, we chose K-mean [29], Mean-shift [16], DBSCAN [30], and Agglomerative [31]. For the classification algorithms, we chose LSTM [32] and GRU [33]. Each method had its parameters and learning activation and optimization layers that can be fine-tuned using Irace. We chose these specific machine learning algorithms since they prove to be useful for many machine learning fields other than the NLP applications like the works cited in [34], [35] and [36].

[1] These implementations are available at: https://github.com/AzChaimae/NLP-applications-with-Word-Embeddings-models-Extrinsic-Evaluation.git

TABLE I. MODEL'S HYPERPARAMETER CONFIGURATIONS

	CBOW (HS)	CBOW (NS)	Skip-Gram (HS)	Skip-Gram (NS)	Glove	FastText
Dataset source	Wikipedia, Facebook, Twitter	Wikipedia, Facebook, Twitter	Wikipedia, Facebook, Twitter	Wikipedia, Facebook, Twitter	Wikipedia, Facebook, Twitter	Wikipedia
Contextual window	3,5,7,9	3,5,7,9	3,5,7,9	3,5,7,9	3,5,7,9	10
Vectors' dimension	200,300, 400	200,300, 400	200,300, 400	200,300, 400	200,300, 400	300

III. METHODOLOGY

A. NER Template

To create an NER System using Word Embeddings, we opted to use annotated existing corpora along with classification and clustering algorithms. Our NER application is performed on an annotated corpus provided by the AQMAR project [37]. This dataset was preprocessed following the steps described in Section II (B). The AQMAR dataset version that we used contains 28 articles hand-annotated to nine named entities, using the BIO system tags i.e., O (outside), B-PER (Beginning of person's entity), B-MIS (Beginning of miscellaneous' entity), B-ORG (Beginning of an organization's entity), B-LOC (Beginning of location's entity), I-PER (Inside of person's entity), I-MIS (Inside of miscellaneous' entity), I-ORG (Inside of an organization's entity), I-LOC (Inside of location's entity). Table II illustrates the statistics of the NER annotated dataset and an example of the annotated tokens.

Our NER approach is illustrated in Fig. 1. The Auto NER configuration class fine-tunes between Word Embeddings models and different machine learning algorithms. The Word Embeddings models were created using Word2Vec, Glove and FastText models. The machine learning algorithms include classification and clustering methods (i.e., LSTM, GRU, K-Means, Mean-Shift, DBSCAN, and Agglomerative).

The Autoconfiguration class calls either an implementation of LSTM networks or GRU networks as classification algorithms or an implementation of K-Means, Agglomerative, DBSCAN or Mean-Shift as clustering algorithms. These algorithms were used to compute for each word representation a score of the considered classes using the BIO system adopted in the annotated corpus. Furthermore, the dataset was split into training, validation and testing sets. The word representations of the training and validation sets were fed to the chosen classification or clustering layer, followed by a sigmoid activation layer if it was set, and complied by Adam, a stochastic Optimization layer again if they were chosen by the Auto NER configuration class of Irace. The optimization of each classification and clustering hyper-parameters is achieved by creating several hyperparameter combinations using different Word Embeddings models, then ranking the results of each NER classifier or cluster with Irace.

To start building an NER system using one of several classifications and clustering algorithms in addition to one of the Word embedding models, we created an Auto NER Configuration class. Fig. 2 illustrates the created methods for our NER Auto parametrization executable Python script[2]. This script is the input of Irace Target Runner.

The Auto parameterization class contains two major methods: 1) The first method is the prediction method. It gets the clusters' prediction using the inputs set by Irace. These inputs are the Word Embedding models and the training algorithm. The NER models are created by preparing the tokens, splitting the training document from the AQMAR dataset into validation and testing sets, and then creating the weight matrix for the words in the sets after loading the chosen Word Embedding model. After defining the training algorithm, the method calls the chosen classification, clustering or activation layer into the compiler in addition to the stochastic optimizers' epochs and batch sizes. 2) The second method calculates the accuracies of the created NER model, by predicting the probabilities of the model, reducing the probability array to speed up the compilation time, and then returning the accuracy value that will be ranked using Irace.

The main objective of Irace is to minimize the cost value returned by the target algorithm, to find the best solution. In our case, the returned cost value of the target algorithm runner is the NER model accuracies multiplied by -1 before returning it to Irace. To calculate the accuracy of each NER model which is the cost value used to find the best solution, we opted for the F1 score if it's a classification method as it was used in [38], and the purity score if it's a clustering method.

Due to the accuracy paradox, which asserts that the accuracy score is unclear when evaluating classification models, the F1 score was utilized to evaluate classification models rather than accuracy. To measure a model's performance, the F1 score uses Precision P and Recall R as shown in Eq. (1).

$$F_1 = \frac{2}{R^{-1}+P^{-1}} = 2 \cdot \frac{P \cdot R}{P+R} \quad (1)$$

$$P = \frac{number\ of\ true\ postives}{number\ of\ true\ positives + number\ of\ false\ positives} \quad (2)$$

[2] Our Auto parametrization python implementation is available at: https://github.com/AzChaimae/Auto-parametrization-for-NLP-application-.git

TABLE II. STATISTICS OF THE NER ANNOTATED CORPUS

Number of tokens	Examples of the annotated tokens
57858	… . الفائقة الأوتار بنظرية انتهاء و ، الكبرى التوحيد بنظريات مرورا …
	… O B-MIS I-MIS I-MIS O O O B-MIS I-MIS I-MIS O …

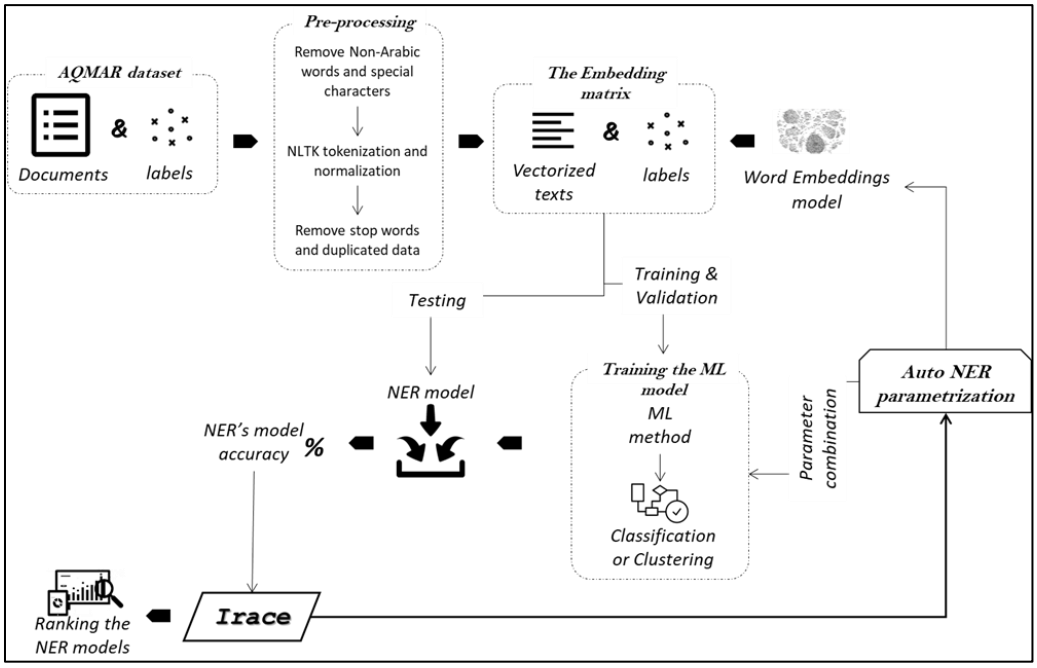

Fig. 1. NER approach using word embeddings.

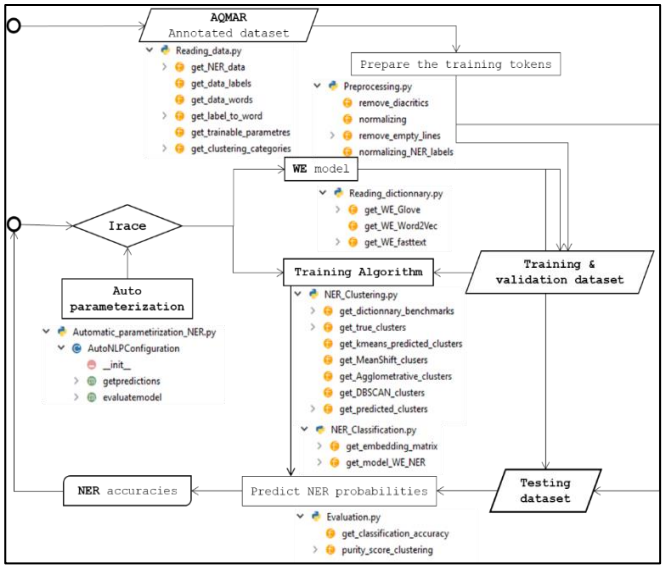

Fig. 2. Auto NER configuration.

$$R = \frac{number\ of\ true\ postives}{number\ of\ true\ positives + number\ of\ false\ negatives} \quad (3)$$

The purity score $purity(\Omega, \mathbb{C})$ is calculated by assigning for each cluster the most frequent class in this cluster, then the accuracy of the assigned class is measured by counting the number of the correct assigned classes and dividing it by N the number of clusters in Eq. (4). Ω presents the set of clusters, and $\mathbb{C}$ is the set of classes.

$$purity(\Omega,\mathbb{C}) = \frac{1}{N}\sum_k \max_j |\omega_k \cap c_j| \quad \Omega=\{\omega_1,\ldots,\omega_K\},\ \mathbb{C}=\{c_1,\ldots,c_J\} \quad (4)$$

B. Autoconfiguring NER

1) Parameter's file. To configure an algorithm, Irace requires a text file that contains all parameters to be tuned, including their type (categorical, ordinal, real, and integer) and sets of their possible values. The algorithm must be prepared to be configured externally with any valid parameter combination from the parameters' file. This was taken into account while creating the Auto NER Configuration class. Our parameters file contains all the parameters that can be taken by the NER auto parametrization class: Embedding file, Method, Flatten Layer, Global Average Pooling Layer, LSTM Layer, LSTM Dropout Layer, GRU Layer, GRU Dropout, GRU recurrent Dropout, Dense activation, Compile optimizer, The Number of batch size, The Number of epochs, K-Means, Mean-shift, DBSCAN or Agglomerative.

2) Forbidden parameter combinations. In addition to the parameters file, Irace provides the option of declaring sets of forbidden parameter combinations. For the Automatic NER parametrization class, several parameter combinations will result in a compilation error or will not be logical to compile.

There is a total of 15 forbidden parameter combinations. If the clustering method was chosen by Irace, the parameters related to the classification methods would not be finetuned, and vice versa for a classification method. Furthermore, one classification or clustering method has to be chosen to train the training weight matrix, the different dense activation and compile optimizer layers can be overlayered and fine-tuned. The rest of the forbidden parameter combinations were set to avoid error compilations.

The total number of possible parameter combinations is around 10 648 combinations, while the mean compilation time is 1.5 to 2 hours depending on the Embedding file the training method and the machine performance. To minimize the compilation time, we had to use the parallelization option offered by Python and Irace to use 23 parallel multiprocessing sessions.

IV. Results and Discussion

The objective of this experimentation is to rank the performance of the different Word embedding models, classification layers and clustering methods in an NER system. The different parameter combinations produced some impressive accuracies, while the higher accuracy being 0,9536 was achieved by training a CBOW model using an LSTM neural network, a SoftMax dense activation, and an Adam compile optimizer. Irace tuned between 1024 parameter combinations to get the best result.

To properly rank all the accuracies, we divided them into four quartiles. The first one contains the best 25% of the accuracies, the second one contains the next 25% best accuracies and so forth. Table III shows the intervals of the quartiles.

Fig. 3 illustrates the different training algorithms that achieved accuracies in the four quartiles. The first observation we can conduct from these results is that the classification algorithms outperformed the clustering ones since the best 25% of the accuracies were obtained by GRU and LSTM neural networks. In the second and third quartiles, the majority of the accuracies refer to the clustering algorithms. While the fourth quartile represents the lowest accuracies obtained by classification algorithms. From Fig. 3 (b), it's obvious that the accuracy performances of the algorithms that belong to the same category (classification or clustering) are relatively similar. So, besides the machine learning approach, the training hyperparameters could highly affect the accuracy of an algorithm.

In the previous graphs, we noticed that even though the best results ranked by Irace were obtained by the classification methods, several mediocre accuracies were also obtained by these classification methods, this can be explained by the fact that we used several layers to create the classification models. Fig. 4 and Fig. 5 illustrate the performance of LSTM and GRU layers in the four accuracy quartiles. LSTM and GRU layers performed relatively differently for both algorithms. The maximum batch size had better results for LSTM, and a minimum batch size gave better results for GRU. However, a minimum number of epochs performed better for LSTM and GRU. Furthermore, SoftMax and Sigmoid dense activations outperformed the ReLU dense activation, and the Adam optimizer architecture resulted in the majority of accuracies in the first quartile for both LSTM and GRU algorithms.

TABLE III. The Quartiles of Accuracies

	Quartile 1	Quartile 2	Quartile 3	Quartile 4
Max	0,9536773	0,90623109	0,86445103	0,4216347
Min	0,90640394	0,86495761	0,42272924	0,3184785

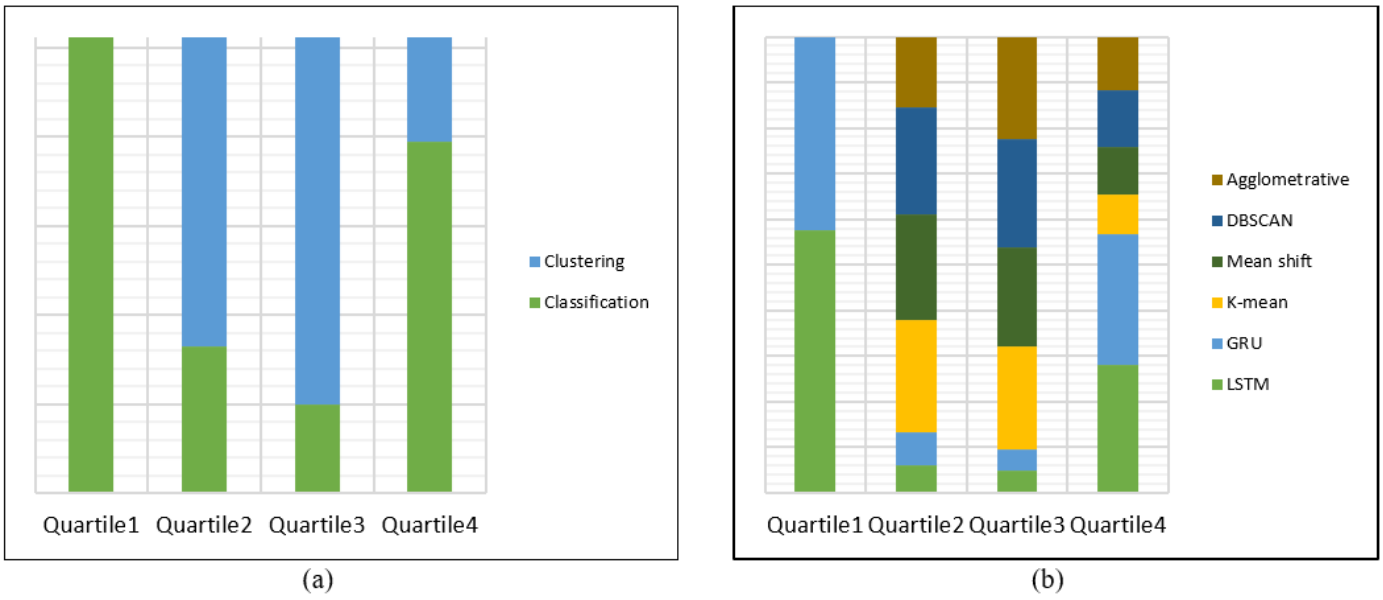

Fig. 3. (a) Classification vs. clustering performances. (b) The performance of all the different classification and clustering methods.

Each training algorithm resulted in both good and mediocre accuracies. Fig 6 illustrates the scale accuracies of the different training algorithms. This illustration enables us to study the distributional characteristics of the accuracy level of each algorithm. Consistently, the median of each box plot of each training algorithm is relatively the same; around 0.86 for clustering algorithms, and 0.89 for classification algorithms. The upper quartile of the clustering algorithms is around 0.93, but the lower quartile of the classification algorithms is around 0.41. The clustering algorithms, however, had an upper quartile of 0.87 but a lower quartile of just around 0.83. For GRU and LSTM, the box plots are comparatively long which means that there is a range of results that behaved differently as a result of different factors. On the contrary, for Agglomerative, K-Means, mean-shift, and DBSCAN, the box plots were significantly shorter with few outliers, which means that even though they were different factors including the Word Embeddings' models, the accuracies for clustering methods had a high level of agreement.

All of the parameters and factors that were used affect the accuracy of a model and its performance in an NLP application, thus, Irace and the use of Word Embeddings was the ideal solution to find the parameters combination that will result in the most accurate model for a NER system or any other NLP applications.

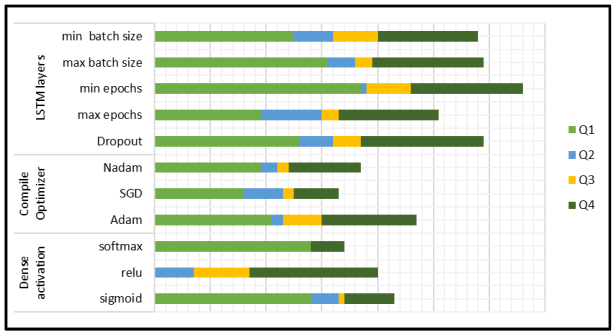

Fig. 4. The performance of the LSTM parameters.

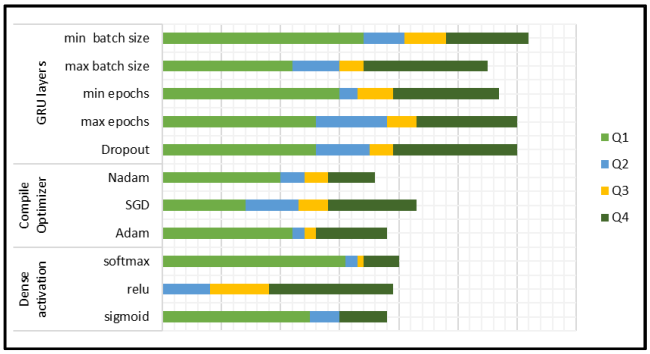

Fig. 5. The performance of the GRU parameters.

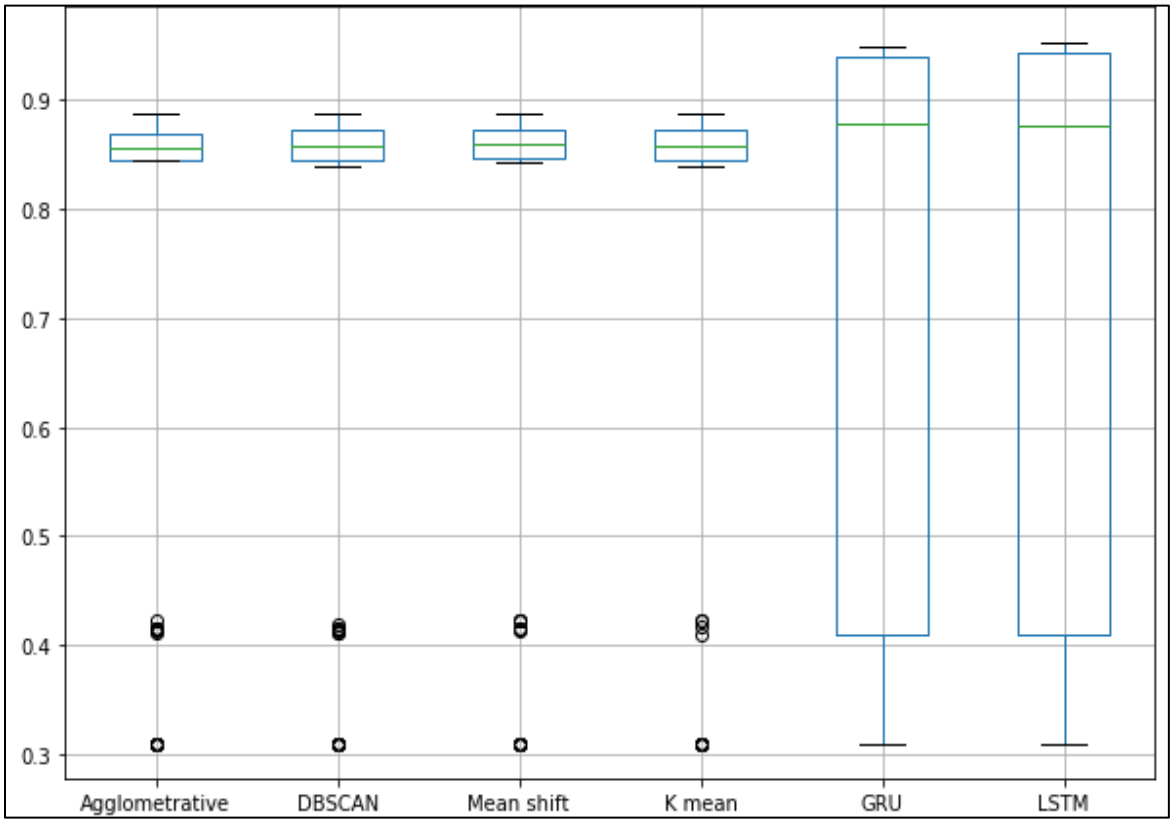

Fig. 6. The distributional scale scores of the training algorithms.

V. CONCLUSION

In this paper, we have proposed a NER approach that fine-tunes between different Word Embedding models and different hyperparameters of a different machine learning algorithm to obtain the most accurate NER system. The Word Embedding models were trained on different Arabic datasets that we collected from different sources to obtain all the Arabic varieties. The NER models were obtained by training the word Embedding representations of annotated Named Entities datasets, the training algorithms included several classification and clustering methods. The Irace package served as an automatic parameter configuration and optimization solution for an Arabic NER system. We have developed a program that was fed to the Irace package, it trains the Word Embeddings representation of the annotated Named Entities according to the selected parameters combination and returns the model accuracy that was ranked by Irace. The most accurate NER model achieved an accuracy of 0.9536.

Several promising directions remain to be explored using Irace and Word Embeddings models; other uprisings Word Embeddings models like Elmo, Bert and XLNET could result in more accurate systems for NER and other NLP applications.

REFERENCES

[1] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," Multimed. Tools Appl., vol. 82, no. 3, pp. 3713–3744, 2023, doi: 10.1007/s11042-022-13428-4.

[2] A. Shoufan and S. Al-Ameri, "Natural Language Processing for Dialectical Arabic: A Survey," pp. 36–48, 2015, [Online]. Available: http://www.aclweb.org/anthology/W15-3205

[3] Y. Benajiba and P. Rosso, "ANERsys 2 . 0 : Conquering the NER Task for the Arabic Language by Combining the Maximum Entropy with POS-tag Information," 3rd Indian Int. Conf. Artif. Intell., pp. 1814–1823, 2007.

[4] Y. Benajiba, P. Rosso, and J. M. Bened Ruiz, "ANERsys : An Arabic Named Entity Recognition System Based on Maximum Entropy," CICLing 2007, pp. 143–153, 2007.

[5] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference, 2016, pp. 260–270. doi: 10.18653/v1/n16-1030.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," CrossRef List. Deleted DOIs, vol. 1, pp. 4069–4076, Jan. 2013, doi: https://doi.org/10.48550/arXiv.1301.3781.

[7] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.

[8] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," Anal. Methods, vol. 5, no. 3, pp. 729–734, Jul. 2016, doi: https://doi.org/10.48550/arXiv.1607.04606.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv Prepr. arXiv1810.04805, 2018, doi: arXiv:1811.03600v2.

[10] M. E. Peters et al., "Deep contextualized word representations," in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2018, pp. 2227–2237. doi: 10.18653/v1/N18-1202.

[11] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," Oper. Res. Perspect., vol. 3, pp. 43–58, 2016, doi: 10.1016/j.orp.2016.09.002.

[12] B. Mouaz, B. H. Abderrahim, and E. Abdelmajid, "Speech recognition of Moroccan dialect using hidden Markov models," IAES Int. J. Artif. Intell., vol. 8, no. 1, pp. 7–13, 2019, doi: 10.11591/ijai.v8.i1.pp7-13.

[13] R. Siddalingappa and K. Sekar, "Bi-directional long short term memory using recurrent neural network for biological entity recognition," IAES Int. J. Artif. Intell., vol. 11, no. 1, p. 89, 2022, doi: 10.11591/ijai.v11.i1.pp89-101.

[14] H. Elzayady, M. S. Mohamed, K. M. Badran, and G. I. Salama, "Detecting Arabic textual threats in social media using artificial intelligence: An overview," Indones. J. Electr. Eng. Comput. Sci., vol. 25, no. 3, p. 1712, 2022, doi 10.11591/ijeecs.v25.i3.pp1712-1722.

[15] W. Yang, H. Long, L. Ma, and H. Sun, "Research on clustering method based on weighted distance density and k-means," Procedia Comput. Sci., vol. 166, pp. 507–511, 2020, doi: 10.1016/j.procs.2020.02.056.

[16] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 5, pp. 603–619, 2002, doi: 10.1109/34.1000236.

[17] R. Zhang, J. Qiu, M. Guo, H. Cui, and X. Chen, "An Adjusting Strategy after DBSCAN," IFAC-PapersOnLine, vol. 55, no. 3, pp. 219–222, 2022, doi: 10.1016/j.ifacol.2022.05.038.

[18] T. Li, A. Rezaeipanah, and E. S. M. Tag El Din, "An ensemble agglomerative hierarchical clustering algorithm based on clusters clustering technique and the novel similarity measurement," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 6, pp. 3828–3842, 2022, doi: 10.1016/j.jksuci.2022.04.010.

[19] A. Chaimae, M. Rybinski, E. Y. Yacine, and J. F. A. Montes, "Comparative study of Arabic Word Embeddings : Evaluation and Application," Int. J. Comput. Inf. Syst. Ind. Manag. Appl. ISSN 2150-7988, vol. 12, pp. 349–362, 2020, [Online]. Available: http://www.mirlabs.org/ijcisim/volume_12.html

[20] A. J. Nebro, C. Barba-González, M. López-Ibáñez, and J. García-Nieto, "Automatic configuration of NSGA-II with jMetal and irace," GECCO 2019 Companion - Proc. 2019 Genet. Evol. Comput. Conf. Companion, pp. 1374–1381, 2019, doi: 10.1145/3319619.3326832.

[21] M. López-ibáñez, L. P. Cáceres, J. Dubois-lacoste, T. Stützle, and M. Birattari, The irace Package : User Guide. 2018. doi: 10.1080/19463138.2012.694818.

[22] R. Adipradana, B. P. Nayoga, R. Suryadi, and D. Suhartono, "Hoax analyzer for Indonesian news using rnns with fasttext and glove embeddings," Bull. Electr. Eng. Informatics, vol. 10, no. 4, pp. 2130–2136, 2021, doi: 10.11591/eei.v10i4.2956.

[23] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," in Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, 2017, pp. 427–431. doi: 1511.09249v1.

[24] A. Chaimae, Y. El Younoussi, O. Moussaoui, and Y. Zahidi, "An Arabic Dialects Dictionary Using Word Embeddings," Int. J. Rough Sets Data Anal., vol. 6, no. 3, pp. 18–31, Jul. 2019, doi: 10.4018/IJRSDA.2019070102.

[25] M. A. S. Md Afendi and M. Yusoff, "A sound event detection based on hybrid convolution neural network and random forest," IAES Int. J. Artif. Intell., vol. 11, no. 1, p. 121, 2022, doi: 10.11591/ijai.v11.i1.pp121-128.

[26] M. Hadni, S. A. Ouatik, and A. Lachkar, "Effective Arabic Stemmer Based Hybrid Approach for Arabic Text Categorization," Int. J. Data Min. Knowl. Manag. Process, vol. 3, no. 4, pp. 1–14, 2013, doi: 10.5121/ijdkp.2013.3401.

[27] R. Boujelbane, M. E. Khemakhem, S. Ben Ayed, and L. H. Belguith, "Building bilingual lexicon to create Dialect Tunisian corpora and adapt language model," Proc. Second Work. Hybrid Approaches to Transl. pages 88–93, Sofia, Bulg. August 8, 2013. c 2013 Assoc. Comput. Linguist., pp. 88–93, 2013.

[28] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," Mach. Learn. 109, pp. 373–440, 2019, doi: 10.1007/s10994-019-05855-6.

[29] J. Yadav and M. Sharma, "A Review of K-mean Algorithm," Int. J. Eng. Trends Technol., vol. 4, no. 7, pp. 2972–2976, 2013.

[30] B. Ma, C. Yang, A. Li, Y. Chi, and L. Chen, "A Faster DBSCAN Algorithm Based on Self-Adaptive Determination of Parameters," Procedia Comput. Sci., vol. 221, pp. 113–120, Jan. 2023, doi: 10.1016/j.procs.2023.07.017.

[31] B. Walter, K. Bala, M. Kulkarni, and K. Pingali, "Fast agglomerative clustering for rendering," in RT'08 - IEEE/EG Symposium on Interactive Ray Tracing 2008, Proceedings, 2008, pp. 81–86. doi: 10.1109/RT.2008.4634626.

[32] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modelling," in 13th Annual International Speech Communication Association, 2012, pp. 194–197.

[33] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of Recurrent Network architectures," in 32nd International Conference on Machine Learning, ICML 2015, 2015, pp. 2342–2350.

[34] [34] S. Patankar, M. Phadke, and S. Devane, "Wiki sense bag creation using multilingual word sense disambiguation," IAES Int. J. Artif. Intell., vol. 11, no. 1, p. 319, 2022, doi: 10.11591/ijai.v11.i1.pp319-326.

[35] C. Huang and G. Qin, "Low-rank matrix optimization for video segmentation research," Indones. J. Electr. Eng. Comput. Sci., vol. 6, no. 1, pp. 36–41, 2017, doi: 10.11591/ijeecs.v6.i1.pp36-41.

[36] A. S. Abdalkafor and S. A. Aliesawi, "Applying of (SOM, HAC, and RBF) algorithms for data aggregation in wireless sensors networks," Bull. Electr. Eng. Informatics, vol. 11, no. 1, pp. 354–363, 2022, doi: 10.11591/eei.v11i1.3462.

[37] B. Mohit, N. Schneider, R. Bhowmick, K. Oflazer, and N. A. Smith, "Recall-oriented learning of named entities in Arabicwikipedia," EACL 2012 - 13th Conf. Eur. Chapter Assoc. Comput. Linguist. Proc., pp. 162–173, 2012.

[38] A. S. M. Afendi and M. Yusoff, "Review of anomalous sound event detection approaches," IAES Int. J. Artif. Intell., vol. 8, no. 3, pp. 264–269, 2019, doi: 10.11591/ijai.v8.i3.pp264-269.