AN INTEGRATED OF CORPUS-BASED AND PRE-TRAINED WORD EMBEDDING MODEL FOR ARABIC NAMED ENTITY RECOGNITION

Nihad Mahmood Adnan, Lailatul Qadri Zakaria

Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia
43600 Bangi, Selangor Darul Ehsan, Malaysia.

Email: nihad_albairuty@yahoo.com, lailatul.qadri@ukm.edu.my

ABSTRACT

Named Entity Recognition (NER) refers to the task of extracting entities' names such as persons, companies and countries. Arabic is one of the languages that has been addressed adequately for the process of NER. In particular, most of the recent related work on Arabic NER have utilized the word embedding technique. This technique aims at exploiting the neural network architecture in order to generate unique embedding for each term. The related work has been divided into studies that have built the word embedding on specific corpus, this approach is known as corpus-based. While other studies have utilized a pre-trained model of word embedding. The first approach suffers from the parameter tuning problem where numerous parameters must be adjusted in order to get precise embedding. Whereas, the pre-trained model which has been proposed to solve the parameter tuning, but still suffers from the 'out-of-vocabulary' problem where some words would not have an embedding within the model. In this manner, this research proposes an integration of corpus-based and pre-trained word embedding to trade-off the limitations of the two approaches. ANERcorp dataset has been used with some normalization tasks such as unwanted characters removal. On the other hand, a corpus-based word embedding model has been built using the ANERcorp corpus. In addition, a pre-trained model that has been trained on Arabic Wikipedia data, has been utilized. Finally, a K-nearest Neighbour (KNN) classifier has been used as a classifier. Results showed that the proposed integration has outperformed the individual implementation of corpus-based and pre-trained models by achieving a weighted average f-measure of 0.93. This result demonstrates the effectiveness of the proposed integration for limiting the out-of-vocabulary problem.

Key words: Arabic Named Entity Recognition, Word Embedding, Corpus-based, Pre-trained, K-Nearest Neighbour

INTRODUCTION

Text processing is a research study that caught the researchers' attentions in the last two decades. Text processing contains various applications like question answering, sentiment analysis and named entity recognition (NER). NER is considered the fundamental task in text processing where the proper nouns are being addressed to be detected (Shaalan, 2014). The earliest methods used to detect proper nouns were depending on rule-based approach, dictionary-based approach or a combination between them (Abdallah *et al.*, 2012). Obviously, to detect proper noun such as a geographical location, there are wide range of keywords that could be followed or before such location. For example, keywords such as 'city', 'state', 'kingdom', 'republic' and others would probably occur with any geographical location. Hence, using a dictionary that contains vast amount of these keywords would be a sufficient solution to detect the named entities. Besides the dictionary, a group of rules can be used to supervise the dictionary where sometimes the keywords would occur before or after the entities. Therefore, the rules will organize and guide the extraction using keywords.

Since the keywords are tremendous and everyday would witness newly keywords, as well as, the variety of cases regarding keyword occurrence, the rule and dictionary techniques have been

abandoned and other methods have taken their place. The most competent methods used for NER were based on machine learning technique. This is because such methods have the ability to learn from historical data and statically. When using the machine learning techniques wide range of features can be examined such as morphological features (e.g. capitalization, uppercase and lowercase) along with semantic features such as the keywords (Al-Shoukry and Omar, 2015).

Recently, several concerning cases have been arisen when using the machine learning techniques for NER task. One of these cases is treating languages like the Arabic language where the entities could take various forms with different dialectical forms. Another challenging issue is the modern representations such as the word embedding and character embedding which are known as deep learning. Deep learning has posed various challenging issues in terms of representing the words. This study will examine some of the aforementioned challenges.

The state-of-the-art in Arabic NER task was depicted by the use of deep learning methods in which word embedding revealed significant improvement on the classification accuracy compared to the conventional methods (Attia et al., 2018; Helwe and Elbassuoni, 2019). Nonetheless, these studies have built their word embedding model using a corpus-based topology where the model is being trained on the terms inside any dataset, corpus or corpora (Levy and Goldberg, 2014). The problem of corpus-based is that it trained on limited number of words and contexts for example, the ANERCorp that has been used to initiate the corpus-based model in the literature is containing around 150 thousand terms that have numerous duplications. In addition, due to the nature of word embedding which depends on neural network architecture, the corpus-based would require massive parameter tuning where size of input, hidden and output layers along with number of epochs should be carefully chosen. Otherwise, the embedding produced by the corpus-based would be imprecise and does not have the ability to differentiate the context of the words.

To overcome the aforementioned problem some studies have exploit a pre-trained model where a fine-tune model that has been trained on large number of terms is being used to give accurate embedding (Ali et al. 2019; Khalifa & Shaalan 2019). However, according to Levy & Goldberg (2014) there are numerous cases where the pre-trained model would not have an embedding for particular terms which known as ‘out-of-vocabulary’ problem. Assume a pre-trained model that has been trained on numerous domain of interests terms and contexts; there will be domain-specific terms that might not be encountered by the model. Therefore, these terms would have not any embedding inside the model.

In this manner, this research proposes a combination of corpus-based and pre-trained model of word embedding in order to overcome the “out of vocabulary” limitation which might improve the classification accuracy.

RELATED WORK

Since the current study focuses on proposing a KNN classifier for the extraction of NE, it is important to highlight some previous studies that have employed the use of such classifier for the same task. Based on the review of literature, it was found that only few researches have employed the use of such classifier, in which the popularly used classifiers include NB, SVM, and NN. However, other researchers Shabat & Omar (2015) have taken leveraged the KNN in terms of examining the similarities between the data instances. More so, they have extracted Arabic NEs from crime documents by combining the KNN, with SVM and NB. The experimental results revealed that the accuracy of classification has been improved by such combination, with an f-measure of 93.36%.

Recently, the modern representations of word embedding introduced by Google in 2013 have been investigated by various scholars. The objective of this representation is to provide embedding value that is made up of a variety of attributes for each word. The meaning of a word, its grammatical tag, as well as its relations to others can be simulated by means of such attributes. The Word2Vec, which is a two-layer Neural Network architecture is used to generate the embedding. With this Word2Vec, the one-hot encoding vector of the word is used as the input, while the distinct encoding is the output.

In a study conducted by Awad et al. (2018) they focused on the generating word embedding by proposing a deep learning technique for Arabic named entity recognition, where the Long Short Term Memory (LSTM) was united with a Conditional Random Fields (CRF). They were able to build and train their model through the use of a given corpus called ANERCorp. Such kind of embedding model is referred to as corpus-based, which means that specific corpus is used in generating embedding. The proposed method has achieved an f-measure of 75.68%. in their study, they entities used were the names of Persons.

In the same way, a Deep Neural Network (DNN) was proposed by Attia et al. (2018) based on word embedding for Arabic NER task. They also employed the use ANERCorp to generate the embedding, and also perform a corpus-based training. The results of their experiments revealed that with their

method, an f-measure of 70.09% was achieved. The focus of this study in terms of the entities was dedicated for the Person names.

Additionally, in a study by Helwe & Elbassuoni (2019), a word embedding method based on LSTM was presented for the recognition of Arabic named entity. Here, they also ANERCorp dataset was used to generate the embedding as well as to test the classification. Based on their experimental results, an f-measure of 83% was achieved. The concentration on this study in terms of the entities was dedicated for the Person names.

The 'out-of-vocabulary' problem was investigated by Khalifa and Shaalan (2019). This problem arises from word embedding. In their study, they presented an LSTM with Convolutional Neural Network (CNN) that is capable of performing both word embedding and character embedding. Nonetheless, they did not use the ANERcorp for word embedding, rather, they used a pre-trained model of embedding that has been previously trained on Arabic Wikipedia words. Through the use of ANERcorp, they were able to achieve an f-measure of 86.96%. The concentration on this study in terms of the entities was dedicated for the Person names.

Lastly, Ali et al. (2019) focused on addressing character and word embedding in the task of Arabic NER through a bi-directional LSTM which they proposed. Likewise, the use of the ANERcorp corpus has been employed in the experiments, the words embedding was done using a pre-trained model of Arabic Wikipedia. Their experimental results revealed that they attained an f-measure of 86.43%. The concentration on this study in terms of the entities was dedicated for the Person names.

MATERIALS AND METHODS

The research design of this study has been set to articulate the research objectives in which the proposed combination embedding method can be applied. For this purpose, the research design contains the required process to apply the proposed method as depicted in Figure 3.1.

The first phase of the design contains the dataset where a set of Arabic named entities dataset is being used for testing the proposed method. The second phase of the design contains the normalization tasks which has been meant to facilitate the text processing and eliminate the unnecessary data.

The third phase contains the two main approaches of the word embedding including corpus-based and pre-trained models. The third phase also contains the proposed combination embedding method where the two approaches are being integrated. Finally, the fourth phase includes the classification using K-nearest neighbor classifier based on the proposed embedding method. As well as, such phase contains the evaluation mechanism. The phases are illustrated as follows:

- **Phase 1 (Arabic NEs Dataset):** contains the details of the dataset used by the proposed method.
- **Phase 2 (Normalization):** contains the preprocessing tasks that intended to prepare the data.
- **Phase 3 (Word Embedding):** contains the word embedding models used in this study along with the proposed embedding.
- **Phase 4 (Classification & Evaluation):** contains the explanation of the utilized classifier along with the evaluation mechanism.

Figure 1 represents the phases.

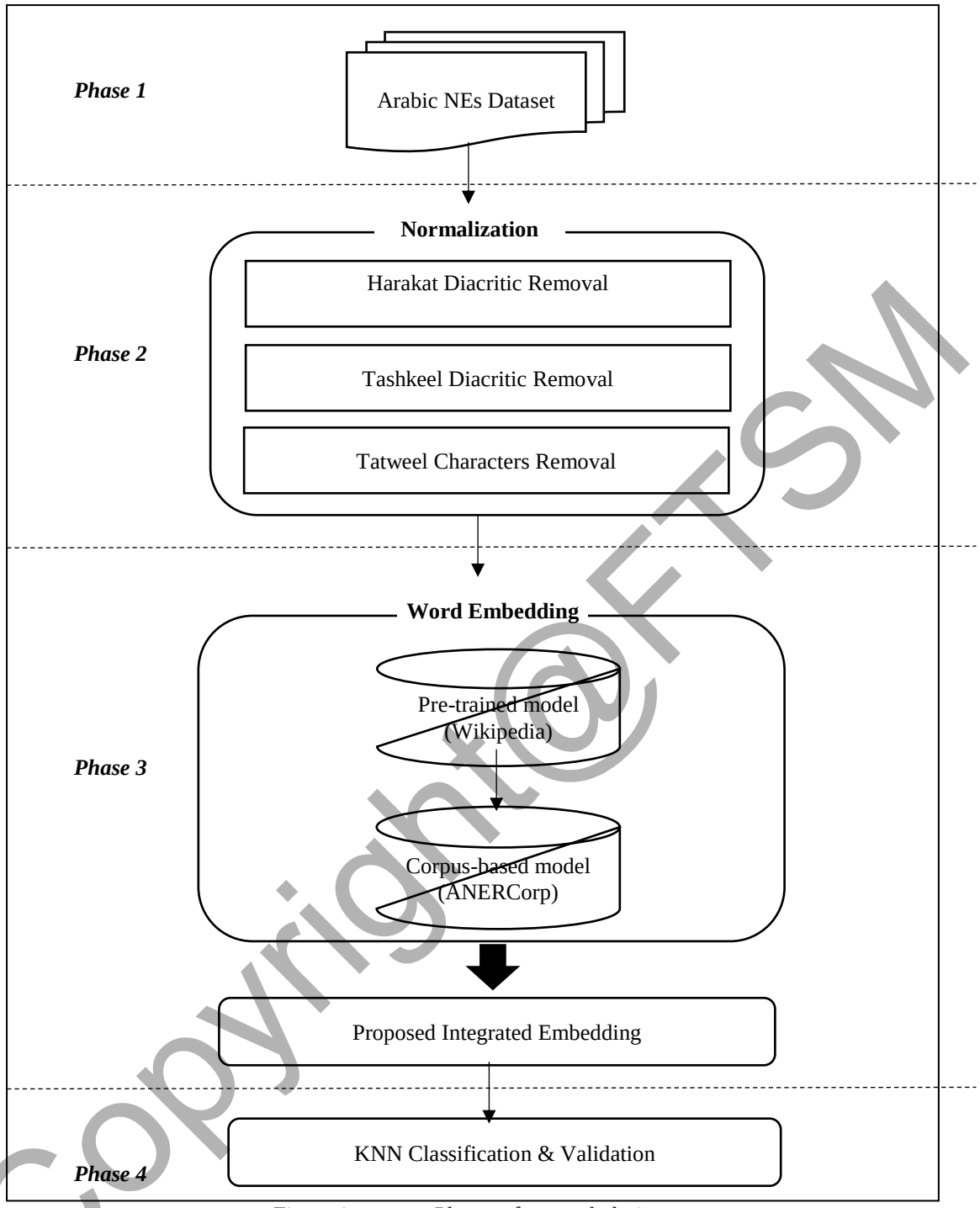

Figure 1. Phases of research design

Arabic Named Entities Dataset

To apply the proposed combination embedding, it is important first to find an annotated corpus that contains Arabic NEs with their class labels. One of the popular annotated dataset for this purpose is the one that proposed by Benajiba et al. (2007) which was called ANERcorp. Such corpus is composed of nearly 150,000 tokens in which each token is being annotated as NE or not. Even the NE class label is divided into person names, location names, organization names and currencies. Table 3.1 depicts the information about ANERcorp dataset.

Table 1 ANERcrop dataset information

Attribute	Training	Testing	Total
Documents	3821	1106	4927
Person	4728	1707	6435
Location	4189	844	5033
Organization	2513	895	3408
Miscellaneous	1223	434	1657
Total entities	12,653	3880	16,533
Regular words (O)	109,606	25,833	135,439
Total tokens	118,851	29,713	~150k

As shown in Table 1, the highest quantity of entities are the person names followed by location, organization and currencies. Table 2 depicts a snippet of the dataset.

Table 2 Snippet of the dataset

Arabic word	English Translation	Tag
دعا	call	O
مندوب	Representative	O
فرنسا	France	LOC
لدى	To	O
الأمم	Nations	ORG
المتحدة	United	ORG
جان	Jan	PERS
مارك	Mark	PERS
مجلس	Council	ORG
الأمن	Security	ORG
الى	To	O
اعلان	Announce	O
هدنة	Truce	O
انسانية	Humanitarian	O

Normalization

In contrast to the classical text analysis problems, NER task has special normalization tasks. For example, in document classification, the stopwords are removed because they have no significant impact on classifying the document. However, in NER task, the stopwords are important for training the classifier on non-NEs. Yet, there are still other normalization tasks that NER requires for better classification accuracy. Following subsections illustrate these tasks in further detail.

Harakat Diacritic Removal

The first type of unnecessary data that need to be eliminated are the diacritics. Diacritics in Arabic are various and have several forms. It means to be used for facilitate pronouncing the word. For instance, a word like ‘جزر’ can be interpreted as ‘carrot’ or ‘islands’. Therefore, the Harakat diacritic is used to guide the reader whether it refers to ‘carrot’ if the diacritics were as ‘جَزَر’ or ‘islands’ if the diacritics were as ‘جُزُر’.

Although the diacritics would be helpful for the pronunciation however, it would mis-lead the machine learning where sometimes a single term would be considered two different words if the diacritics have been formed differently. For instance, the two terms ‘الأمَمُ’ and ‘الأمَمَ’ refers to the same meaning which is ‘nations’, but the machine would consider them as different terms because they have different diacritics.

Hence, it is better to get rid of these diacritics in order to unify the embedding of words that have the same meaning. Table 3 depicts an example of Harakat elimination.

Table 3 Example of Harakat elimination

Word with Harakat	Word without Harakat
دَعَا	دعا
مَندوبُ	مندوب
فرنسَا	فرنسا
لِدى	لدى
الأمَمُ	الأمم
المتحِدَة	المتحدة
جَان	جان

مَارك	مارك
مَجلِسَ	مجلس
الأمنِ	الأمن
الى	الى
اِعْلانِ	اعلان
هُدنة	هدنة
اِنسانية	انسانية

To apply Harakat removal, a python library known as PyArabic introduced by (Zerrouki, 2010) was utilized.

Tashkeel Diacritic Removal

Another type of Arabic diacritics is the Tashkeel which has been used to indicate the word structure within a sentence such as subject, object or verb. Such Tashkeel is also mis-leading for the machine learning in which two identical terms in meaning but with different Tashkeel would be considered as different terms. Hence, it is better to get rid of these Tashkeel diacritics. Table 4 represents such process.

Table 4 Tashkeel elimination example

Tashkeel Woord	**Word without Tashkeel**
دعا	دعا
مندوبٌ	مندوب
فرنسا	فرنسا
لدى	لدى
الأممٌ	الأمم
المتحدةً	المتحدة
جان	جان
مارك	مارك
مجلساً	مجلس
الأمنٌ	الأمن
الى	الى
اعلانِ	اعلان
هدنةٌ	هدنة
انسانيةٌ	انسانية

To apply Tashkeel removal, a python library known as PyArabic introduced by (Zerrouki, 2010) was utilized.

Tatweel Characters Removal

Arabic words can be written in various ways. One of the ways that Arabic word can be written with is the Tatweel characters which usually indicate an exaggeration within the context. For instance, the word 'فرنسا' which means 'France' can be written as 'فرنســـا' in order to give an exaggeration within the context. Such exaggeration can be represented by giving the extra characters of 'ـــ'. However, adding these characters would differentiate the treatment of the same word but with such extra characters. The machine would give different embedding for the same words with different Tatweel characters. Hence, it is necessary to get rid of these characters to unify the treatment of terms. Table 5 represents this process.

Table 5 Example of Tatweel characters elimination

Word with Tatweel Characters	**Word without Tatweel Characters**
دعـــا	دعا
منـــدوب	مندوب
فرنســا	فرنسا
لـــدى	لدى
الأمـــم	الأمم
المتحـــدة	المتحدة
جـــان	جان
مـــارك	مارك
مجـــلس	مجلس
الأمـــن	الأمن
الـــى	الى
اعـــلان	اعلان

هدنـــــة	هدنة
انســانية	انسانية

To apply Tatweel removal, a python library known as PyArabic introduced by (Zerrouki, 2010) was utilized.

Word Embedding

In recent years, the text analysis field has witnessed a revolutionary progress in terms the semantic techniques. One of these techniques is the word embedding which aims to input words from a particular context through a specific Neural Network to give each word a distinct embedding that refers to its meaning within the context (Xing *et al.*, 2015). Such neural network is called Word2Vec which composed of two main topologies; Skipgram and Continuous Bag of Words (CBOW). The former topology works by inputting a word and the aim is to predict its surrounding words. In contrast, the latter topology works by inputting the surrounding words to predict the center word.

In fact, both topologies have their own pros and cons which make them suitable applications. For instance, skipgram has shown superior performance in applications such as named entities recognition and relation extraction according to some authors (Chiu *et al.*, 2016). This is because in skipgram, the representation of a focus word is learnt by predicting every other context word in the window independently, with the prediction error of each context word back-propagated to the target word. This might provide better vectors to be learnt as a focus word is trained over more data, but with less smoothing over contexts. Another factor that makes the skip-gram is much suitable is that the state of the art in Arabic NER have used it (Attia, *et al.*, 2018; Helwe and Elbassuoni, 2019). Therefore, in this study, the skipgram will be considered for the word embedding.

However, the word embedding model can be created using two main approaches including corpus-based and pre-trained. Such approaches can be described in the next subsections.

Corpus-based Embedding Model

This approach aims at implementing the word embedding model from the scratch where a specific corpus that is related to a particular domain of interest is being used within the training. Creating the model can be represented by preparing a corpus of text in which the terms of such corpus will processed through the Word2Vec architecture. Assume a corpus C text that contains several words as follow:

C = "دعا مندوب فرنسا لدى الأمم المتحدة"

The first task before processing the words through the Word2Vec architecture is to get the unique terms of the corpus and represent them via one-hot encoding paradigm. Such paradigm aims to articulate the distinctive terms as attributes and instances where the correspondences among terms will be depicted as '1' and the mis-match terms are depicted as '0'. Table 6 depicts the one-hot encoding representation of the corpus C.

Table 6 One-hot encoding

Terms/ Terms	**دعا**	**مندوب**	**فرنسا**	**لدى**	**الأمم**	**المتحدة**
دعا	1	0	0	0	0	0
مندوب	0	1	0	0	0	0
فرنسا	0	0	1	0	0	0
لدى	0	0	0	1	0	0
الأمم	0	0	0	0	1	0
المتحدة	0	0	0	0	0	1

As shown in Table 6 each word has a vector with a length equivalent to the number of distinctive terms inside the corpus. For instance, the word 'دعا' would have a vector of:

دعا → 1 0 0 0 0 0

To obtain the embedding of 'دعا' word, the one hot vector of such word will be processed through the Word2Vec as depicted in Figure 2.

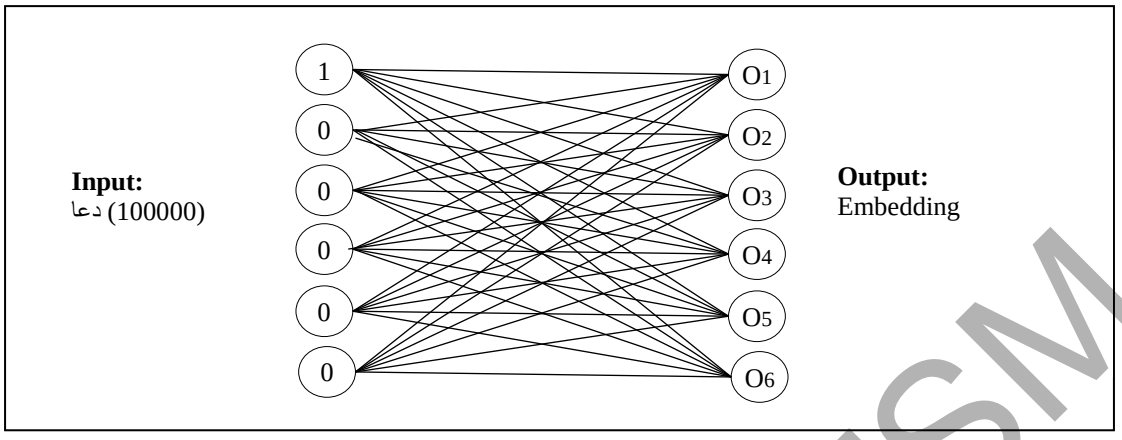

Figure 2. Inputting the word 'دعا' via the Word2vec architecture

As shown in Figure 2, every neuron from the input layer is attached with all the neurons from the output layer. Every attachment is associated with a weight that randomly generated. To compute the output neurons' values, Equation 1 depicts the general formula of neural network weights multiplication (Srivastava *et al.*, 2014):

$$O_k = \sum I_j \times W_i \quad (1)$$

Where I_j is an input neuron and W_i is the corresponding weights that linked it with the output O_k. Assuming that the output neurons are computed using Equation 1, Figure 3 depicts the results of embedding.

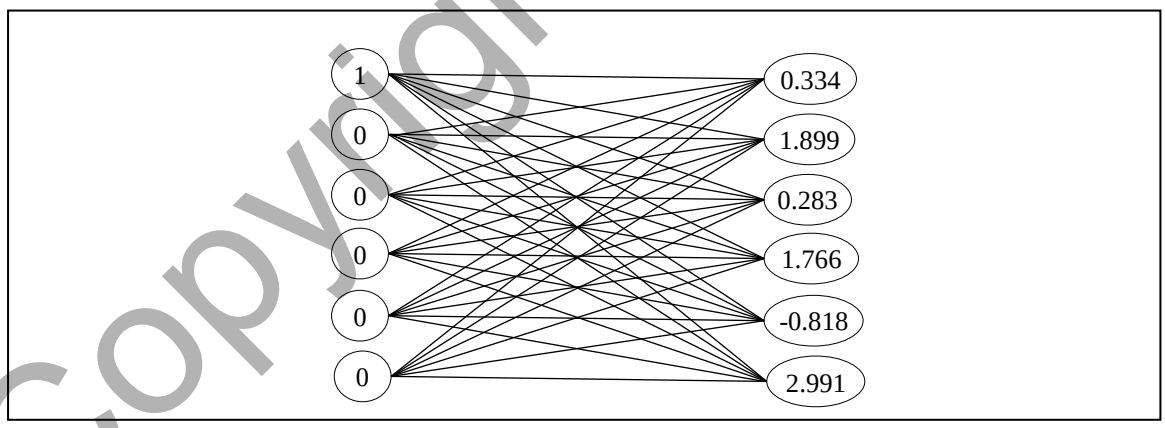

Figure 3. Output neurons computation

As depicted in Figure 3, based on the results of the output neurons' values, the embedding for the word 'دعا' can be represented as:

'دعا' 0.334 1.899 0.283 1.766 -0.818 2.991

Every word embedding model has its own parameter setting. Therefore, it is necessary to highlight the parameter values used in this paradigm. Table 7 displays the parameter values and their description.

Table 7 Corpus-based parameter settings

Parameter	Description	Value
Documents No.	The number of sentences included in ANERcorp dataset	4,929
Token No.	The number of all words inside the ANERcorp dataset	148,564
Dimension	The number of distinctive tokens within the corpus	300
Window size	The number of surrounding words considered	5
Minimum count	Minimum time of occurrence for the terms considered in the dimension	2
Epochs	Number of iterations of the training	10

As depicted in Table 7, the parameters have been adjusted to build the model. The values shown in the table is the standard ones used by most of the literature (Rong, 2014).

The corpus-based model has some advantages such as the domain-specific learning of terms in which the corpus used to build the embedding would have terms that are related to specific domain of interest. This would make the model is accurate to predict terms from the same domain. Yet, this would be a limitation when the model is asked to predict a term related to another domain. Furthermore, the corpus-based model suffers of the parameter tuning required to get the best values of parameters (Goldberg and Levy, 2014).

Pre-Trained Embedding

The corpus-based model has multiple limitations thus, researchers have attempted to overcome these limitations. One of the successful attempts to solve the problem of parameter tuning is to experiment various parameter values in order to determine the most accurate parameters. Therefore, tremendous efforts have been dedicated to provide a model that is well-trained and fine-tuned and save such model for future uses, this approach is called pre-trained model.

In this same manner and unlike the corpus-based, the pre-trained model can be trained on millions of terms that are related to various domain of interests in order to solve the problem of limited contexts lies in the corpus-based. Hence, such pre-trained model would be exploited easily by many researchers and for several applications without bothering building the model from the scratch.

Unlike English language where tremendous pre-trained embedding models have been introduced in recent years, few studies have addressed the pre-trained model for Arabic language. However, recently, the study of Soliman et al. (2017) has shown a great effort to release a pre-trained model for Arabic language. The model has been trained on Wikipedia Arabic webpages in which 1.8 million documents with 2225.3 million tokens have been considered. Table 8 displays the values of the parameters used by such model.

Table 8. Pre-trained parameter settings

Parameter	Value
Source	Arabic Wikipedia
Document No.	1.8 million
Token No.	2225.3 million
Dimension	300
Window size	5
Minimum count	500
Epochs	1000

As depicted in Table 8, the pre-trained model's parameters demonstrate a large number of trained token and documents. This can generate a robust embedding that take into the account numerous contexts. In addition, the model parameters have been accurately adjusted. According to Soliman et al. (2017), the value of 300 for the dimension was demonstrated higher accuracy in text classification compared to other values such as 100 and 200. Whereas, the value 5 of window words is proven widely to be semantically efficient in which the context can be captured from at least five words. Moreover, in terms of the minimum count, Soliman et al. (2017) concluded that a value of 500 would contribute toward avoiding insignificant terms that have occurrences less than 500. Eventually, a large value such as 1000 for epoch numbers would help the neural network to be error-tuned. However, according to Levy et al. (2015), the pre-trained model suffers of a drawback called out-of-vocabulary when the model encounter a word in the testing that has no embedding stored in the model.

Proposed Combination Embedding Model

Due to the limitations existed in both the corpus-based and the pre-trained models, this research proposes a combination between the two models where the incorporation between them would overcome some limitations. Such idea of embedding combination has been inspired by some studies that showed superior performance when combining different embedding models. For instance, the study of Amiri & Shobi (2017) has shown an improvement in classifying tweets when combining Word2Vec with Doc2Vec. Therefore, this study will overcome the out-of-vocabulary drawback by using the corpus-based model, while overcoming the problem of parameter-tuning by taking the advantage of the pre-trained model. Figure 4 represents the workflow of the combined embedding where several check conditions are being considered.

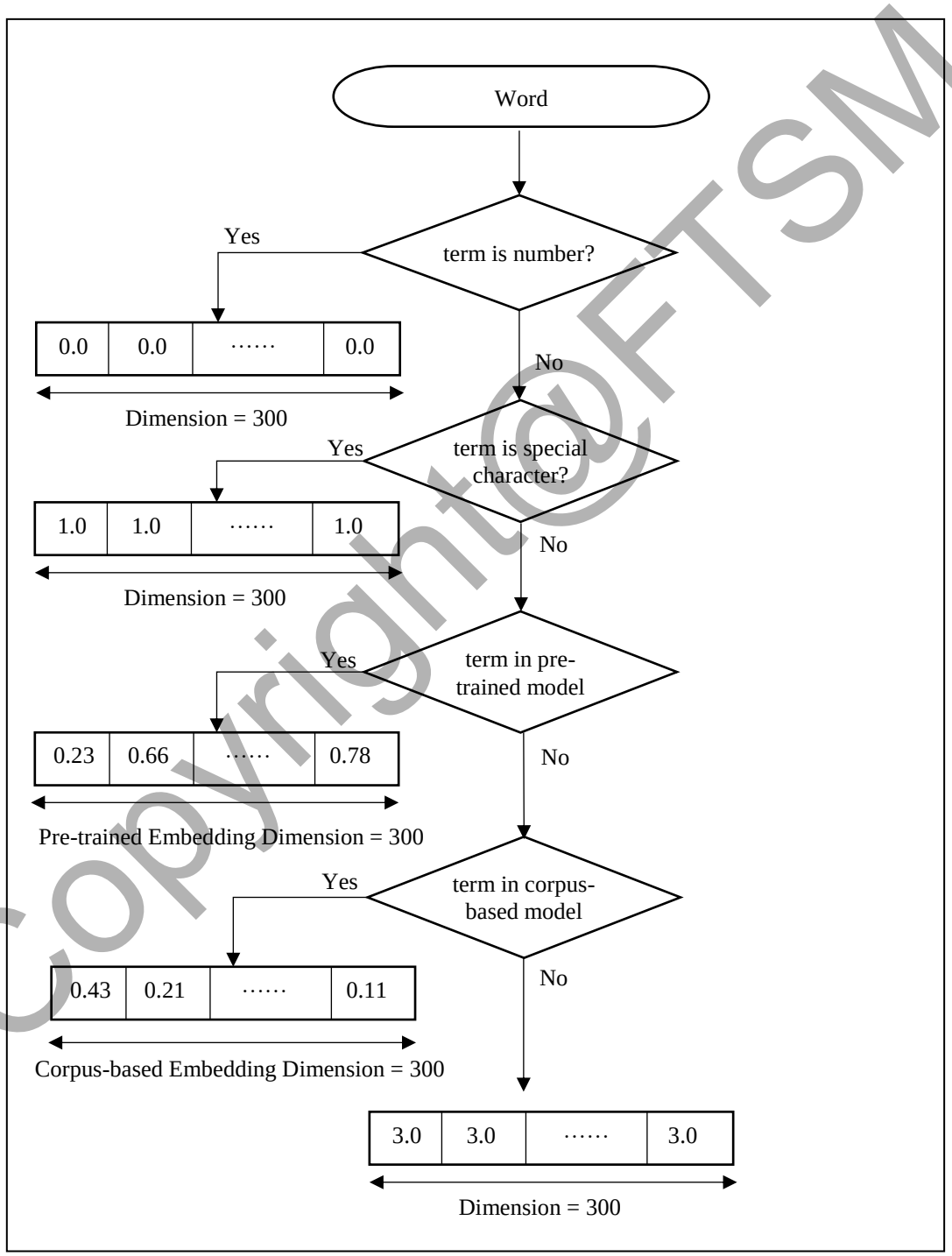

Figure 4. The Integrated embedding workflow

As depicted in Figure 4, the flowchart starts by checking whether the word is a number or not. If yes, a vector of 300 dimension is being initiated where all its values are set to 0.0 in order to provide a vector with real numbers. The reason behind unifying all the digit-words with a specific embedding is that the separate treatment of digit-words would not facilitate determining NEs.

After that, if the word was not number, another check-up condition is being used to check whether the word is a special character or not. If yes, a vector of 300 dimension is being initiated where all of its values will be set to 1.0 in order to provide a vector with real numbers. The reason behind unifying all the special characters with a single embedding is that it would facilitate the machine to recognize such word as not NE.

Then, if the word was not a special character, another check-up condition will be applied to check whether the word is existing in the pre-trained model or not. If yes, the embedding of such word will be brought from the pre-trained model where its vector length would be 300.

Yet, it is worth mentioning that the pre-trained model is suffering from the out-of-vocabulary problem where the word would have no embedding inside the model. Therefore, this study will utilize the corpus-based model in order to get the embedding of the word with a vector of 300 dimension.

Eventually, if the word is not existing inside the corpus-based model (which is unlikely probability), a vector of 300 dimension will be initiated and its values will be set to 3.0.

Classification

To accommodate the classification, the K-Nearest Neighbor (KNN) is used to train on the produced embedding and tested by its ability to predict a subset of the data. This study has used KNN because it is based on identifying the most similar instance from the training to the testing instance (Khamar, 2013). Since there are different embedding used (i.e. corpus-based and pre-trained) KNN will suite the variations of embedding between training and testing portions.

RESULT

The results of classification have been evaluated using precision, recall and f-measure. Every model has been evaluated separately with KNN classification. Table 9 shows the results.

Table 9. Weighted F-measure comparison

	F-measure		
Entity	**Corpus-based**	**Pre-trained**	**Proposed model**
PERS	0.5	0.69	0.73
LOC	0.68	0.68	0.75
ORG	0.4	0.39	0.53
MISC	0.35	0.29	0.33
O	0.95	0.96	0.97
Weighted Average	**0.89**	**0.91**	**0.93**

Greatest f-measure obtained for classifying person entity was got by the proposed model where the f-measure was 0.73 compared to 0.69 obtained by the pre-trained and 0.5 obtained by the corpus-based model. Apparently, the use of pre-trained model has outperformed the corpus-based because it has much sophisticated embedding since it has been training on huge number of words and context. Yet, the proposed model has outperformed both the pre-trained and the corpus-based because it has took the advantage of both models. It exploited the capability of pre-trained model in terms of generating sophisticated embedding for the word, as well as, it exploited the capability of corpus-based in terms of 'out-of-vocabulary' problem that exists in the pre-trained model.

Similarly, the highest f-measure obtained for classifying location entity was got by the proposed model where the f-measure was 0.75 compared to 0.68 obtained by the pre-trained and 0.68 obtained by the corpus-based model. Apparently, the proposed model has outperformed both the pre-trained and the corpus-based because it has took the advantage of both models. It exploited the capability of pre-trained model in terms of generating

sophisticated embedding for the word, as well as, it exploited the capability of corpus-based in terms of 'out-of-vocabulary' problem that exists in the pre-trained model.

For the organization entity, the corpus-based model has outperformed the pre-trained model where it got 0.40 compared to 0.39. The reason behind this outperformance is that the corpus-based trained on the ANERcorp dataset where the organization entities inside such dataset were associated with specific fields like politics. While the pre-trained model has been trained on various fields therefore, it revealed weaker performance regarding organization entities classification. Yet, the proposed combined model has outperformed the two models by gaining 0.53 of f-measure. This is because it gained the advantages of both models.

For the MISC entity, the highest f-measure was obtained by the corpus-based model where the f-measure was 0.35 compared to 0.29 achieved by the pre-trained model and 0.33 achieved by the proposed model. The reason behind this is that the pre-trained model has been trained on wide range of domains, while the ANERcorp contains specific currencies. Since the proposed model is prioritized the pre-trained model firstly and then the corpus-based thus, the proposed model has been impacted by the poor performance of the pre-trained model.

For the non-NEs, the proposed combined model obtained the highest f-measure of 0.97 compared to 0.95 obtained by the corpus-based and 0.96 obtained by the pre-trained model.

The outperformance that the proposed combined model has made for all classes (excepts the MISC), has led to superior results regarding weighted average f-measure of the combined model. This has been depicted where the proposed model obtained a weighted average f-measure of 0.93 compared to 0.91 obtained by the pre-trained model and 0.89 obtained by the corpus-based model. Figure 5 represents the results of the three models based on all the classes.

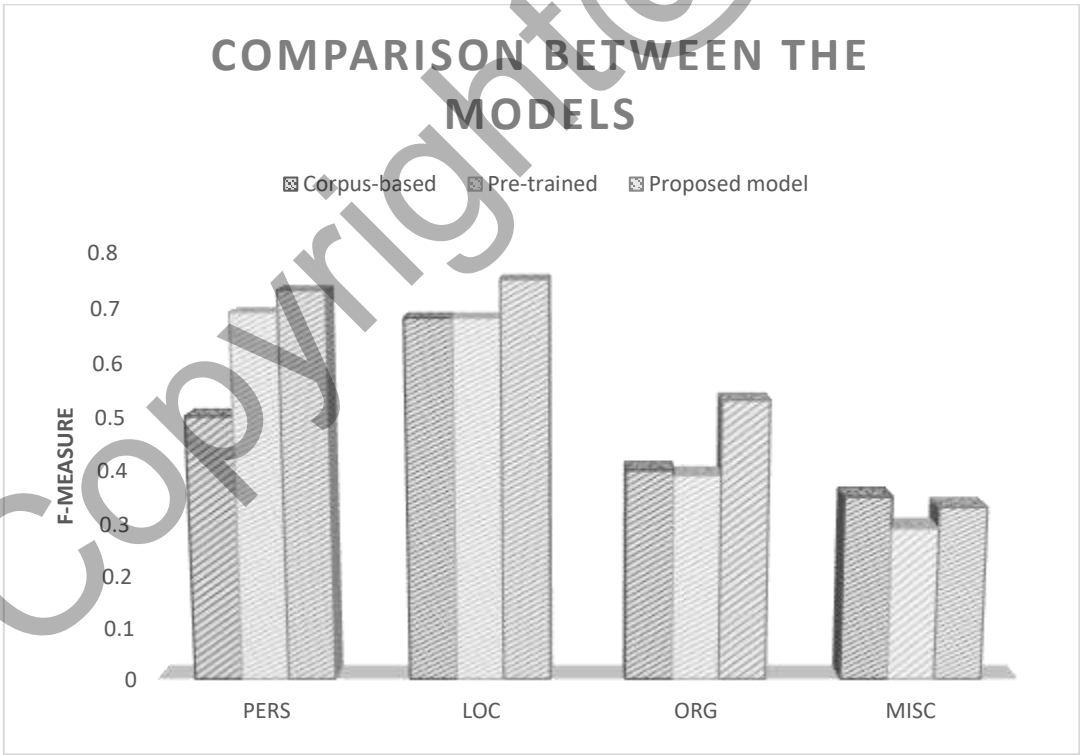

Figure 5. Representation of entities for the three models

CONCLUSION

This study proposed an integrated of corpus-based and pre-trained word embedding model for Arabic NER task. The proposed integrated model has outperformed the individual application of corpus-based and pre-trained model.

Examining further embedding topologies such as character embedding and document embedding would be a useful research. Although these topologies are expensive regarding time and resources usage yet, they might give better results.

REFERENCES

Abdallah, et al. 2012. 'Integrating rule-based system with classification for Arabic named entity recognition', *Computational Linguistics and Intelligent Text Processing*, Springer, pp. 311-322.

Al-Shoukry and Omar. 2015. Arabic named entity recognition for crime documents using classifiers combination, International Review on Computers and Software, Vol. 10 No. 6, pp. 628-634 (Access 2015

Ali, et al. 2019. Boosting Arabic Named-Entity Recognition With Multi-Attention Layer, IEEE Access, Vol. 7, pp. 46575-46582 (Access 2019

Amiri and Shobi. 2017. A link prediction strategy for personalized tweet recommendation through doc2vec approach, Research in economics and management, Vol. 2 No. 4, pp. 63-76 (Access 2017

Attia, et al. 2018. Ghht at calcs 2018: Named entity recognition for dialectal arabic using neural networks, *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pp. 98-102.

Awad, et al. 2018. Arabic Name Entity Recognition Using Deep Learning, Springer International Publishing, Cham, pp. 105-116.

Benajiba, et al. 2007. 'Anersys: An arabic named entity recognition system based on maximum entropy', *Computational Linguistics and Intelligent Text Processing*, Springer, pp. 143-153.

Chiu, et al. 2016. How to train good word embeddings for biomedical NLP, *Proceedings of the 15th workshop on biomedical natural language processing*, pp. 166-174.

Goldberg and Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method, arXiv preprint arXiv:1402.3722, (Access 2014

Helwe and Elbassuoni. 2019. Arabic named entity recognition via deep co-learning, Artificial Intelligence Review, Vol. 52 No. 1, pp. 197-215 (Access 2019

Khalifa and Shaalan. 2019. Character convolutions for Arabic Named Entity Recognition with Long Short-Term Memory Networks, Computer Speech & Language, Vol. 58, pp. 335-346 http://www.sciencedirect.com/science/article/pii/S0885230818301657 (Access 2019

Khamar. 2013. Short text classification using kNN based on distance function, IJARCCE International Journal of Advanced Research in Computer and Communication Engineering. Government Engineering College, Modasa.(ISSN Print: 2319-5940 ISSN Online: 2278-1021), (Access 2013

Levy and Goldberg. 2014. Dependency-based word embeddings, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 302-308.

Levy, et al. 2015. Improving distributional similarity with lessons learned from word embeddings, Transactions of the Association for Computational Linguistics, Vol. 3, pp. 211-225 (Access 2015

Rong. 2014. word2vec parameter learning explained, arXiv preprint arXiv:1411.2738, (Access 2014

Shaalan. 2014. A survey of Arabic named entity recognition and classification, Computational Linguistics, Vol. 40 No. 2, pp. 469-510 (Access 2014

Shabat and Omar. 2015. Named Entity Recognition in Crime News Documents Using Classifiers Combination, Middle-East Journal of Scientific Research, Vol. 23 No. 6, pp. 1215-1221 (Access 2015

Soliman, et al. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp, Procedia Computer Science, Vol. 117, pp. 256-265 (Access 2017

Srivastava, et al. 2014. Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research, Vol. 15 No. 1, pp. 1929-1958 (Access 2014

Xing, et al. 2015. Normalized word embedding and orthogonal transform for bilingual word translation, *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1006-1011.

Zerrouki. 2010. Pyarabic, an arabic language library for python.