A Combination of TFIDF and Pre-Trained Word Embedding Model for Arabic Named Entity Recognition

Nadia Baqer Hassoon, Lailatul Qadri Zakaria

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

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

ABSTRACT

Arabic Named Entity Recognition (NER) refers to the extraction of original nouns in Arabic language like the names of people, organiztrion, location and others. The state of the art in Arabic NER were based on the modern technology of word embedding. This technology utilizes a feed-forward neural network in order to give words within different contexts a distinctive embedding. Such technology showed remarkable performance in terms of extracting NEs. However, it suffers from a serious limitation which is the 'out-of-vocabulary' problem where some terms within the testing might not have an embedding within the model. Therefore, this study aims to incorporate additional information to the word embedding in order to solve the aforementioned problem. Such additional information can be obtained by the Term Frequency Inverse Document Frequency (TFIDF). A benchmark dataset has been considered in the experiments along with some pre-processing tasks including diacritics and unnecessary characters removal. In addition, a pre-trained model has been utilized to get large repository for Arabic words' embedding. Such model has been trained on Arabic Twitter data. Lastly, a Decision Tree (DT) classifier has been utilized to accommodate the classification. Experimental results showed that the integration of TFIDF and word embedding has outperformed the word embedding only by obtaining a weighted f-measure of 0.91. This result proves the usefulness of the proposed incorporation of TFIDF and word embedding in terms of overcoming the problem of 'out-of-vocabulary'.

Key words: Arabic Named Entity Recognition, Word Embedding, TFIDF, Pre-trained, Decision Tree

INTRODUCTION

Natural language processing (NLP) is an interesting research area where several text tasks have been addressed. One of the significant tasks was the Named Entity Recognition (NER), this task aims to train the machine in order to be able to detect the original nouns like the names of people, organization, location and other entities (Abdallah *et al.*, 2012). In the past, the studies were mainly relying on predefined rules that specify the occurrence of named entities (Abdallah, *et al.*, 2012). For example, utilizing specific terms that would likely occur with named entities can help the machine to recognize them. Let a frequent term as 'doctor' such term would possibly occur vastly with people's name such as 'Doctor John' or 'Doctor Adam'. Such frequent terms can be organized in a dictionary or so-called Gazetteer and once the machine detect an occurrence for one of these terms, the machine would extract either its preceding or following terms as entities based on predefined rules. The rules will guide the machine whether to extract the terms before or after the frequent terms based on its type such as location, organization or person.

Nonetheless, due to the drawbacks of rule-based approach like the difficulty of building the rules and the wide range of customizations required to deal with different languages, researchers tended to utilize other techniques such as the machine learning (Shaalan, 2014). These techniques do not require building a set of rules, but rather, it builds a model statistically to examine the appearance of named instances (Al-Shoukry and Omar, 2015). Apparently, the frequent terms can be examined as individual features where the single term would be tested if it is a frequent term or not, and if it is a frequent term, the probability of followed or preceding word to be a named entity would be arisen.

Machine learning technique have shown substantial capabilities for identifying the named entities with high classification accuracy. However, there are still numerous challenges that would face the task of NER like encountering sophisticated languages as Arabic, examining sophisticated representations such as the word and character embedding, and other issues. This study intends to tackle the task of NER specifically in Arabic and explore the challenges for optimizing the classification accuracy.

Recent studies in Arabic NER are exploiting word and character embedding. Character embedding has shown the highest f-measure values but it suffers from the expensive computation where numerous parameters need to be tuned. Word embedding is still having fair results but it is composed of two approaches corpus-based and pre-trained. The corpus-based refers to the model building upon the terms and contexts of ANERCorp dataset (Attia *et al.*, 2018; Helwe and Elbassuoni, 2019). While, pre-trained refers to the exploitation of a previously trained model on large Arabic text collection (Ali *et al.*, 2019; Khalifa and Shaalan, 2019). In fact, the pre-trained model tends to be more effective compared to the corpus-based since it has been fine-tuned and trained on larger number of terms and contexts. However, it suffers from the 'out-of-vocabulary' problem where some terms in ANERCorp might not exist in the pre-trained model.

Recently, some studies have suggested the use of additional information along with the embedding for limiting the case of 'out-of-vocabulary'. De Boom et al. (2016) have incorporated the term frequency of terms along with the embedding for short text classification. The incorporation of TFIDF will contribute toward decreasing the problem of 'out-of-vocabulary'. This can be represented when the pre-trained model would not have an embedding for a given term. In this regard, the TFIDF of such term can be used to initiate an embedding vector which might help the machine learning to configure its context.

Hence, this study aims to exploit such idea of incorporation TFIDF with the pre-trained embedding model to limit the 'out-of-vocabulary' cases in Arabic NER.

RELATED WORK

Past studies were reviewed for the purpose of this research since the study is proposing the DT classifier for the NE extraction just like some past studies. To extract Arabic NEs from crime documents, Al-Shoukry & Omar (2015) used the DT classifier. The experiment results revealed that integrating DT with NB has helped improve the accuracy of classification where the f-measure gotten was 94.19%.

Recently, modern representations of word embedding introduced by Google in 2013 was investigated by various studies. This representation aims at providing a distinct embedding value that is made up of a wide range of characteristics for each word. Such characteristics simulate the meaning of the word, its grammatical tag and how it is related to other words. Word2Vec which is a two-layer Neural System architecture is used to generate the embedding. It inputs one-hot encoding vector of the expression and yield the distinct embedding.

Profound acquiring tactic for Arabic termed unit perception where a Long Short-Term Memory (LSTM) is incorporated into a Conditional Random Fields (CRF) to create the word embedding was presented by Awad et al., (2018). The authors have built and trained their model in order to create the embedding for each word using a particular corpus called ANERCorp. This type of embedding model is known as corpus-based which means a generation of embedding using particular corpus. An f-measure of 75.68% has been acquired using the proposed method. The concentration on this study in terms of the entities was dedicated to the names of the people.

Similarly, a Deep Neural Network (DNN) based on word embedding for Arabic NER task was revealed by Attia et al., (2018). The same corpus of ANERCorp was investigated and the same method of creating the embedding was used where a corpus-based training was done. An f-measure of 70.09% for the proposed method was revealed from the experiment carried out. The concentration on this study in terms of the entities was dedicated to the Person's name.

Furthermore, a word embedding approach based on LSTM for Arabic named entity recognition was revealed. For both generating the embedding and testing the classification, the ANERCorp dataset was investigated. Results revealed 83% f-measure of the proposed method. The concentration on this study in terms of the entities was dedicated to the Person's name. Helwe & Elbassuoni (2019).

Khalifa and Shaalan (2019) investigated the problem of 'out-of-vocabulary' that resulted from word embedding. A LSTM with Convolutional Neural Network (CNN) to incorporate mutual charisma embedding and expression embedding was proposed by the authors. However, the authors have not generated the embedding from the corpus of ANERcorp for word embedding. Instead, a pre-trained model of embedding that was previously trained on Arabic Wikipedia words was exploited. 86.96% was the f-measure result using ANERCorp. The concentration on this study in terms of the entities was dedicated to the Person's name.

Lastly, a bi-directional LSTM for addressing word and character embedding in Arabic NER was presented by Ali et al., (2019). Similarly, the ANERcorp corpus has been used in the experiments. However, the embedding of the words has been brought from a pre-trained model of Arabic Wikipedia. The f-measure was 86.43%. The concentration on this study in terms of the entities was dedicated to the Person's name.

MATERIALS AND METHODS

This section intends to explain the framework of the proposed embedding by tackling the components of such framework. From Figure 1, it is obvious that the framework includes four stages; dataset, preprocessing, embedding, and classification. First stage will discuss the details of the dataset used within the experiments. While the second stage discusses the required preprocessing tasks to turn the data into much suitable form. Third stage elaborates on the core contribution of this study where the proposed embedding method is being illustrated in detail. Fourth stage discusses the classification method that has been applied through the baseline and the proposed embedding. The stages can be summarized as follows:

- **Stage 1 (Dataset):** tackles the dataset used within the experiments.
- **Stage 2 (Preprocessing):** tackles the preparation tasks required to clean the data.
- **Stage 3 (Word Embedding):** tackles the proposed embedding method.
- **Stage 4 (Classification):** tackles the classifier used to predict the NEs and the validation method.

Figure 1 represents the stages.

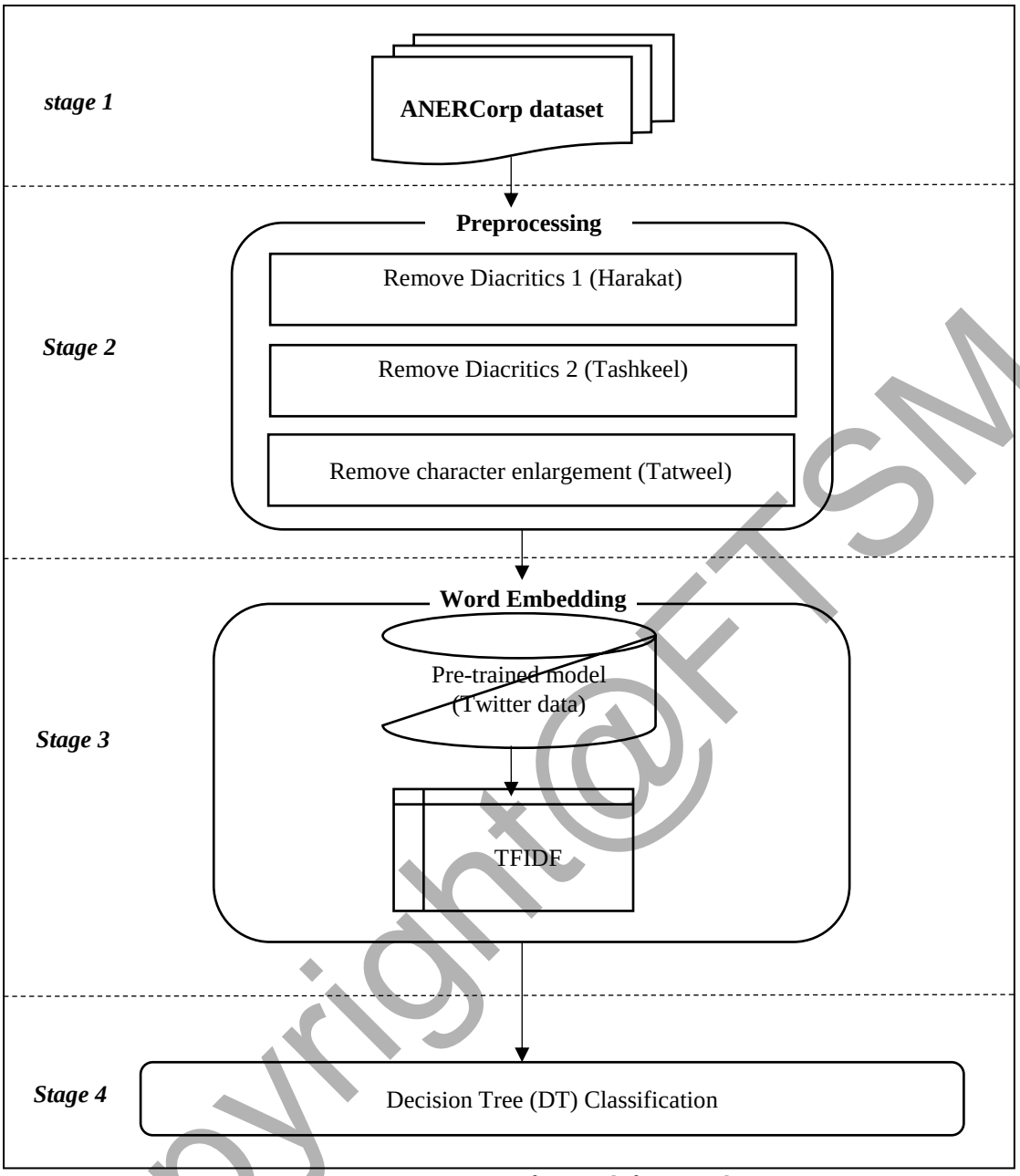

Figure 1. Stage of research framework

Dataset

This stage includes the details of the dataset used within the experiments. The dataset used is known as ANERcorp which is an annotated dataset for Arabic named entities introduced by Benajiba et al. (2007). The data contains approximately 150 thousand tokens. It contains different labels ranging from person into currencies. Table 1 highlights the statistics of the dataset.

Table 1 Dataset details

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

Preprocessing

Unlike the traditional text analysis, the named entity recognition task has a unique preprocessing scheme in which the stopword removal or stemming is not considered. The reason behind this is that the overall assessment of any NER system is to recognize any word whether it is a named entity or not. Therefore, removing the stopwords or performing the stemming would not help the recognition task. However, for Arabic language, a single term would be written with various forms. Thus, the preprocessing in this study aims to unify such forms for facilitating the discrimination of words.

Remove Diacritic 1

The first factor that might cause the variations in writing a word in Arabic is the diacritic. Such diacritic is meant to help the reader pronouncing the word. For example, a single term in Arabic such as 'ذهب' would have multiple meaning based on the pronunciation, if it is formed as 'ذهبَ' it would mean 'gone', but it is formed as 'ذهبٌ' it would mean 'gold'. Hence, the use of diacritics would help to differentiate the meaning of the word, but at the same time, since the diacritics are considered as independent characters thus, within the analysis some words with the exact meaning would gain different embedding. For example, the word 'صناعةٌ' and 'صناعةُ' have the same meaning of 'industry', yet they have different characters based on the variations of diacritics.

In this regard, this study will eliminate the diacritics in order to unify the word embedding within the dataset. Table 2 highlights a snippet of discarding these diacritics.

Table 2 Example of removing diacritics 1

Term with diacritic 1	**Term without diacritic 1**
أعلنَ	أعلن
اِتحادُ	اتحاد
صِناعةِ	صناعة
السّياراتُ	السيارات
في	في
المٰانيا	المانيا

To remove the Harakat, a python package called PyArabic (Zerrouki, 2010) has been used.

Remove Diacritic 2

Similar to the previous section, this task is also meant to remove another type of diacritics which are also used for helping the reader to pronounce a word in Arabic. However, these diacritics have different types of characters which are usually applied on the last letter in any Arabic word. Table 3 highlights a snippet of removing these diacritics.

Table 3 Snippet of removing diacritics 2

Term with diacritic 2	Term without diacritic 2
أعلنَ	أعلن
اتحاداً	اتحاد
صناعةٌ	صناعة
السّياراتٍ	السيارات
في	في
المانيا	المانيا

To remove the Tashkeel, a python package called PyArabic (Zerrouki, 2010) has been used.

Remove Characters Enlargement

In Arabic language, there is a mechanism used sometimes to highlight specific words to catch the attention of the reader. Such mechanism aims at enlarging some letters. For example, the word 'الحياة' can be written as 'الحيـــاة' in order to grab the attention. Although both words have the same meaning yet, they would have different embedding within the text analysis. Therefore, it is important to remove the enlargement in order to unify both words. Table 4 highlights a snippet of removing the character enlargement.

Table 4 Snippet of removing character enlargement

Word with Tatweel	Word without Tatweel
أعلـــن	أعلن
اتحـــاد	اتحاد
صـــناعة	صناعة
الســـيارات	السيارات
فـــي	في
المـــانيا	المانيا

To remove the Tatweel, a python package called PyArabic (Zerrouki, 2010) has been used.

Word Embedding

Recently, the word embedding has been used for text analysis due to its powerful performance in terms of identifying word similarity based on the context. Word embedding is being built using a neural network architecture known as Word2Vec. Such architecture consists of two main paradigms including Skip-gram and Continuous Bag of Word (CBOW). The first paradigm aims at processing a single term for the sake of predicting its context or surrounding terms. While the second paradigm is processing multiple context terms in order to predict the target term. According to Chiu et al. (2016) the skip-gram is usually preferred for tasks such as entity mention tagging and relation extraction.

For getting an embedding for a given term, Word2vec model is initiated by determining a target corpus that contains respective number of words and contexts. However, the main challenging issue faces the model building is to identify robust parameters. In fact, there are many parameters that involved in the model building of embedding such as the dimension of embedding, window words, minimum count and number of epochs. The dimension refers to the length of embedding which usually determined in when identifying the unique terms in any corpus. While, the window words are the number of surrounding words selected by the model. Minimum count refers to the least occurrence number of terms to be selected within the dimension. Finally, the epochs are the error-tuning iterations within a neural network architecture.

Pre-Trained Embedding

As mentioned earlier, building a word embedding model would suffer from the limited contexts that exist in the selected corpus, as well as, the problem of parameter tuning. Therefore, the research community in Natural Language Processing (NLP) field has tended to dedicate a huge effort for providing a model that can be trained on vast amount of text with millions of words and tens of thousands of contexts. In this regard, much effort can be given for tuning the parameters in order to get the most accurate embedding. Hence, many researchers would have the ability to take the advantage of such pre-trained model to accommodate different text analysis tasks.

One of the popular pre-trained models in Arabic language is the one introduced by Soliman et al. (2017). Such model has been trained on Twitter Arabic data where 66.9 million documents along with

1090 million tokens have been considered. Table 5 highlights parameter settings used by the pre-trained model.

Table 5. Pre-trained parameter settings

Parameter	Value
Text source	Twitter
Number of documents	66.9 million
Number of tokens	1090 million
Dimension	300
Window size	5
Minimum count	500
Epochs	1000

As shown in Table 5, the pre-trained model showed large number of trained token and documents. This would lead to produce a sophisticated embedding that consider wide range of contexts. On the other hand, the model parameters have been accurately adjusted. According to Soliman et al. (2017), the 300 dimension has shown better performance in terms of text analysis compared to other parameters such as 100 and 200. While, 5 window words showed better performance in text analysis compared to other values. The reason behind such better is that 5 window words would lead to consider two words before and two words after a target term, which is enough for learning the context of the target term. In addition, choosing a value of 500 for a minimum count was showing fair performance because it would avoid the terms with lesser occurrences of 500 which might be insignificant terms. Finally, a value of 1000 of epochs was showing good text classification accuracy due to the possible iterations of error-tuning.

Yet, there is still a serious limitation behind the pre-trained model, such limitation is known as 'out-of-vocabulary' (Levy *et al.*, 2015). This problem occurred when the pre-trained model is used to predict a specific text data that might contain terms that have no embedding inside the pre-trained model.

Term Frequency Inverse Document Frequency (TIDF)

As shown within previous sections, the pre-trained model suffers from several limitations. Therefore, some authors have suggested to utilize additional information with the embedding. For example, De Boom et al. (2016) have incorporated the TFIDF information along with the embedding for short text analysis. This study has been motivated by this hypothesis in which the TFIDF of a term can be incorporated with the pre-trained embedding for limiting the case of 'out-of-vocabulary'.

In fact, TFIDF is composed of two parts; TF which is the simple notion computing appearances of a given term, and IDF is the ratio of term occurrence in terms of a specific document. To understand the mechanism of TFIDF, Table 6 highlights a snippet of words taken from the ANERcorp dataset.

Table 6 Dataset snippet of words

Sentence	Word	Translation	Class
Sentence 1	قدر	announces	O
	خبير	Expert	O
	اقتصادي	economic	O
	اسرائيلي	Israeli	O
	إن	That	O
	تكلفة	Cost	O
	العدوان	invasion	O
	على	On	O

Sentence 2	لبنان	Lebanon	LOC
	بلغت	Reached	O
	مليار	Billion	O
	دولار	Dollar	MISC
	.	.	O
	فيما	however	O
	بلغ	estimated	O
	الضرر	destruction	O
	الاقتصادي	economical	O
	داخل	Inside	O
	البلدات	provenances	O
	الاسرائيلية	Israeli	O
	مليار	Billion	O
	شيكل	Shekel	MISC
	.	.	O

As shown in Table 6, words in the ANERcorp dataset are separated independently in every line. These words are forming sentences in which the sentences are being separated by period (i.e. '.'). Therefore, the table showed two sentences. First step to get TFIDF is to consider the distinctive terms without duplication. Table 7 highlights these distinctive terms.

Table 7 Term frequency

	Distinctive Terms																			
Sentences	قدر announces	خبير expert	اقتصادي Economic	اسرائيلي Israeli	إن That	تكلفة cost	العدوان Invasion	على on	لبنان Lebanon	بلغت reached	مليار billion	دولار Dollar	فيما however	بلغ estimated	الضرر destruction	الاقتصادي economical	داخل inside	البلدات Provenances	الإسرائيلية Israeli	شيكل Shekel
S_1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
S_2	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1

As shown in Table 7, the words have either occurred in S1 or in S2 where the occurrence has been depicted as '1' and the absence has been depicted as '0'. However, only the word 'مليار / billion' has occurred in both sentences. Hence, examining only the TF would not be sufficient in terms of determining the importance of a term because some terms might occur in a sentence while disappear from other sentences. Therefore, it is important to examine the IDF which measure the ratio of a term occurrence in accordance to all the documents or sentences. It can be calculated as follow:

$$IDF = \log N/Nt \quad (1)$$

N indicates the sentences quantity, while Nt indicates the number of sentences contain the term t.

In respect the above formulas, all the words within Table 7 have a single occurrence in either sentence 1 or sentence 2 excepts the word 'مليار / billion' which had occurrences in both sentences. Therefore, all the terms would have an IDF of 0.3, whereas the word 'مليار / billion' would have an IDF of 0. Table 8 highlights the gain of IDF.

Table 8 IDF values

Terrm	**IDF computation**
قدر	$IDF = \log N/N_t = \log 2/1 = 0.3$
خبير	$IDF = \log N/N_t = \log 2/1 = 0.3$
اقتصادي	$IDF = \log N/N_t = \log 2/1 = 0.3$
اسرائيلي	$IDF = \log N/N_t = \log 2/1 = 0.3$
إن	$IDF = \log N/N_t = \log 2/1 = 0.3$
تكلفة	$IDF = \log N/N_t = \log 2/1 = 0.3$
العدوان	$IDF = \log N/N_t = \log 2/1 = 0.3$
على	$IDF = \log N/N_t = \log 2/1 = 0.3$

لبنان	IDF = log N/N_t = log 2/1 = 0.3
بلغت	IDF = log N/N_t = log 2/1 = 0.3
مليار	IDF = log N/N_t = log 2/2 = 0
دولار	IDF = log N/N_t = log 2/1 = 0.3
فيما	IDF = log N/N_t = log 2/1 = 0.3
بلغ	IDF = log N/N_t = log 2/1 = 0.3
الضرر	IDF = log N/N_t = log 2/1 = 0.3
الاقتصادي	IDF = log N/N_t = log 2/1 = 0.3
داخل	IDF = log N/N_t = log 2/1 = 0.3
البلدات	IDF = log N/N_t = log 2/1 = 0.3
الاسرائيلية	IDF = log N/N_t = log 2/1 = 0.3
شيكل	IDF = log N/N_t = log 2/1 = 0.3

Now, to represent the TFIDF values for all terms, both Table 7 and Table 8 will be multiplied. The results of such multiplication can be depicted in Table 9.

Table 9 Term frequency

	Unique Terms																			
Sentences	قدر	خبير	اقتصادي	اسرائيلي	ان	تكلفة	العنوان	على	لبنان	بلغت	مليار	دولار	فيما	بلغ	الضرر	الاقتصادي	داخل	البلدات	الاسرائيلية	شيكل
S_1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0	0.3	0	0	0	0	0	0	0	0
S_2	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3

Proposed Combination Embedding Model

After explaining both the pre-trained and TFIDF, this section will discuss the proposed combination embedding method using the two aforementioned approaches. Figure 2 depicts the flowchart of the proposed embedding where several check conditions are being considered. As shown in Figure 3.2, the flowchart begins with a checkup whether the term is digit or not. If the term is digit, a vector with a length of 300 dimension will be created and populated with the value of '0.0'. This is because giving each digit a distinct embedding would not help identifying the NEs. Therefore, all the digits would have a unified embedding.

The checkup continues if the term was not digit by accommodating another checkup whether the term is a punctuation or not. If the term is a punctuation, a vector with length of 300 dimension will be created and populated with the value of '1.0'. Similar to the digits, giving different embedding for every punctuation would not help the identification of NEs.

If the term was not a punctuation, the flowchart continues to check whether the term has an embedding inside the pre-trained model. If it has embedding, it would simply bring the embedding and assigned to the term.

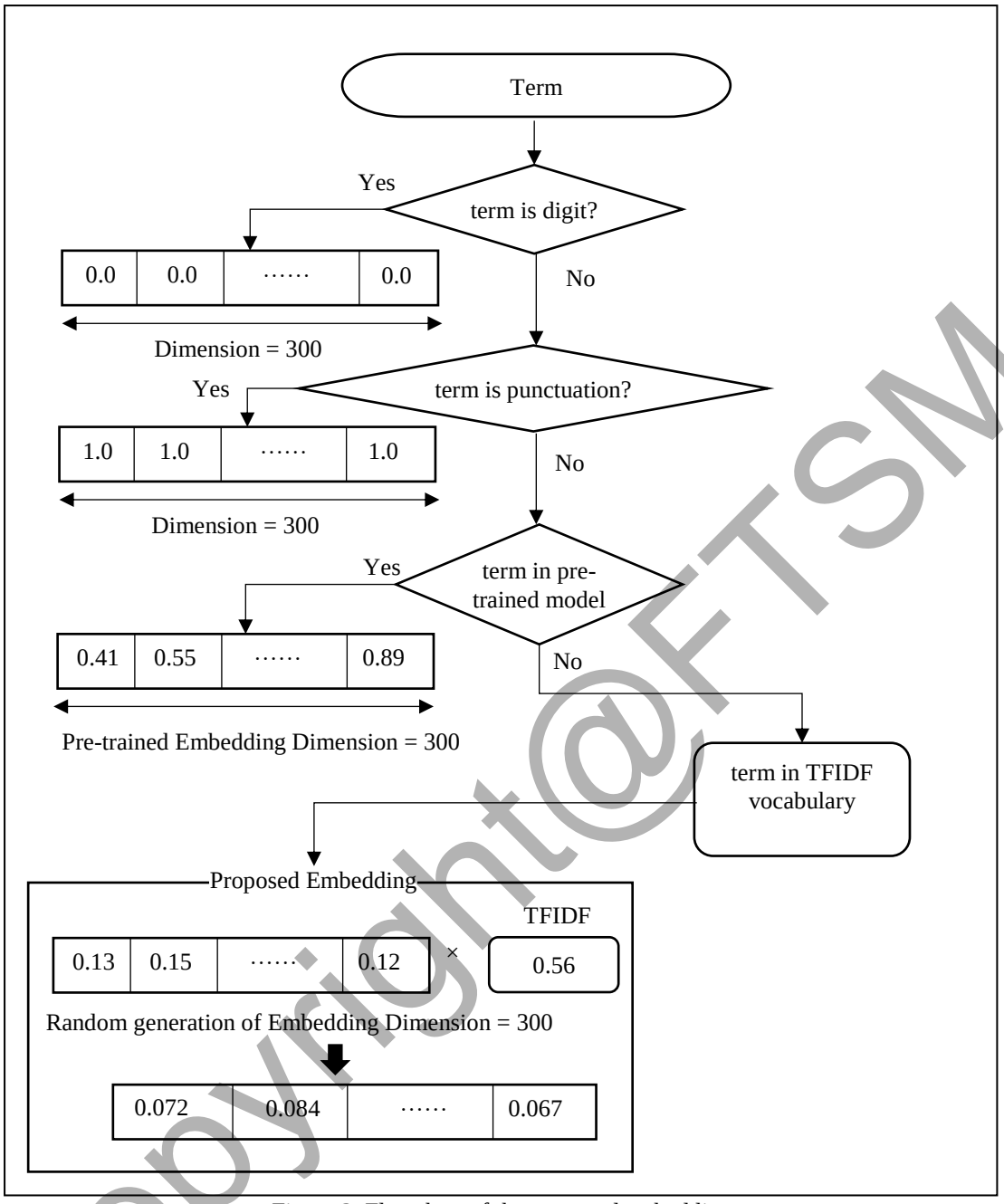

Figure 2. Flowchart of the proposed embedding

However, it is possible to encounter the problem of 'out-of-vocabulary' where a term would not have an embedding inside the pre-trained model. Therefore, the proposed embedding will utilize the TFIDF for clarifying the term's significance based on its occurrence. Then, a vector with length of 300 will be created and populated with random values that will be later multiplied by the TFIDF value of the term.

Classification

In order to perform the classification, Decision Tree (DT) classifier has been used in the experiments. The reason behind using DT is that it has effective performance in terms of addressing embedding features in a tree manner (Guo and Berkhahn, 2016).

RESULT

The results of classification have been evaluated using precision, recall and f-measure. Every model has been evaluated separately with DT classification. Table 10 shows the results.

Table 10. Weighted F-measure comparison

Class label	Pre-trained	Pre-trained + TFIDF
Person	0.55	0.61
Location	0.6	0.71
Organization	0.37	0.47
Currency (MISC)	0.31	0.35
Non-Named Entity (O)	0.95	0.96
Weighted Average	**0.89**	**0.91**

As stated in the comparison, first, the person class label has been classified accurately by the combined model where the f-measure was 0.61 compared to the pre-trained model where the f-measure was 0.55. Apparently, the proposed combined model has outperformed the pre-trained model because the proposed model would have additional information to the pre-trained embedding which is the TFIDF. Such additional information has helped the model to reduce the problem of 'out-of-vocabulary'.

Similarly, for the location class label, the outperformance was dedicated for the proposed combined model where the f-measure was 0.71 compared to 0.6 acquired by the pre-trained model. As mentioned earlier, the pre-trained model has been targeting Twitter data which has a lot of informal Arabic location names in which the single location would be written differently. Therefore, the pre-trained model would not be able to classify the location names inside the ANERcorp dataset correctly. In contrast, the incorporation of TFIDF has contributed to solve this problem by improving the classification accuracy. This is because the TFIDF would examine the occurrence of location names within ANERcorp which helps the classification.

Furthermore, for the organization class label, the outperformance was dedicated for the proposed combined model where the f-measure was 0.47 compared to 0.37 acquired by the pre-trained model. As mentioned earlier, the pre-trained model has been targeting Twitter data which has a lot of informal Arabic organization names in which the single location would be written differently. Therefore, the pre-trained model would not be able to classify the organization names inside the ANERcorp dataset correctly. In contrast, the incorporation of TFIDF has contributed to solve this problem by improving the classification accuracy. This is because the TFIDF would examine the occurrence of organization names within ANERcorp which helps the classification.

Yet, the currency class label (MISC) detection has been improved when incorporating the TFIDF where the proposed model acquired an f-measure of 0.35 compared to 0.31 acquired by the pre-trained model. This is because the pre-trained model was built based on Twitter data which might not contain many currency mentions, while the combined model would have the ability to take the advantage of TFIDF.

For the non-named entities, the pre-trained model showed an f-measure of 0.95 which can be justified due to its capability of detecting wide range of words. Finally, the incorporation of TFIDF has contributed toward improving the classification of such entity where the f-measure was 0.96.

To sum up, the incorporation of TFIDF with pre-trained model has improved all the classes. This leads to an overall average of f-measure 0.91 for the proposed combined model compared to 0.89 for the pre-trained model. This can demonstrate the usefulness of the proposed incorporation of TFIDF.

Figure 3 displays the f-measure of each class label for both the pre-trained model and the proposed combined model.

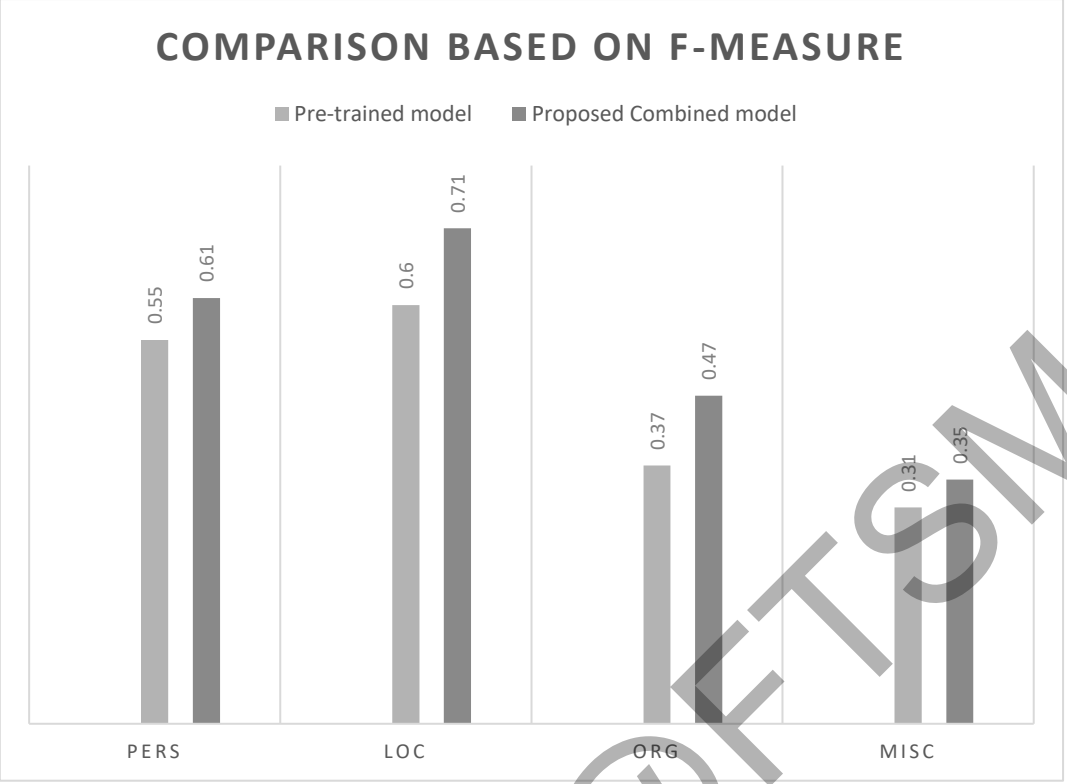

Figure 3. F-measure results for the two models

CONCLUSION

This study proposed a combination of pre-trained word embedding model and TFIDF for Arabic NER task. The proposed combined model has outperformed the individual application pre-trained model. Examining new embedding architectures like document and character embedding can be a great opportunity for examining their capabilities in terms of NER task.

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

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.

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.

De Boom, et al. 2016. Representation learning for very short texts using weighted word embedding aggregation, Pattern Recognition Letters, Vol. 80, pp. 150-156 (Access 2016

Guo and Berkhahn. 2016. Entity embeddings of categorical variables, arXiv preprint arXiv:1604.06737, (Access 2016

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

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

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

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

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