A Comparative Study of Word Embedding Models for Arabic Text Processing

Fatmah Assiri and Nuha Alghamdi
fyassiri@uj.edu.sa nalghamdi@uj.edu.sa
University of Jeddah, College of Computer Science and Engineering, Jeddah, Saudi Arabia

Summary
Natural texts are analyzed to obtain their intended meaning to be classified depending on the problem under study. One way to represent words is by generating vectors of real values to encode the meaning; this is called *word embedding*. Similarities between word representations are measured to identify text class. Word embeddings can be created using word2vec technique. However, recently fastText was implemented to provide better results when it is used with classifiers. In this paper, we will study the performance of well-known classifiers when using both techniques for word embedding with Arabic dataset. We applied them to real data collected from Wikipedia, and we found that both word2vec and fastText had similar accuracy with all used classifiers.

Keywords:
Word Embeddings, Arabic Text Processing, FastText, Word2Vec, NLP.

1. Introduction

Natural language processing (NLP) includes text analysis to obtain its intended meaning. It has been widely used to solve many problems, such as distinguishing fake news, finding users' interests in recommender systems, and much more. E-commerce expanded recently, especially with the outbreak of COVID-19. To improve users' experience, e-commerce software uses user demographics, shopping history, social media, etc. Collected data can be analyzed and classified to determine users' interests, and then services and products are provided based on these [1][2]. To build a better recommender system, text analysis was used to gain insights into individuals' interests. The first step of this process is word embeddings, which generates a vector of real values to encode the meaning of the text. Then, to build the classifier, word embeddings were used, and the similarities between vectors are measured to classify the text.

There are different techniques to generate word embeddings [3, 4, 5]. Word2vec was presented as a better approach than feature extraction according to Altowayan et al. [4]. In this method, each word is represented in a list of numbers called a vector. A mathematical equation, such as cosine similarity, measures the semantic similarities between words. Word2vec is limited to the set of words in the training set.

Manuscript received August 5, 2022
Manuscript revised August 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.8.50

Recently, a word embedding model, called fastText, was implemented by Facebook to generate better embeddings. It represents words as bags of n-gram characters; then, the n-gram's vectors are summed [6]. It supports many languages, and it overcomes the issues of word2vec, as it can generate embeddings to new words that are not part of the training set.

The research community has claimed that the performance of fastText is better than that of word2vec; however, to the best of our knowledge, there is no study comparing the accuracy of classifier algorithms when each technique was applied using Arabic text. Thus, in this work, we conducted a controlled experiment to compare the accuracy of different classifiers when different word embedding techniques where used: word2vec and fastText.

Our previous study used fastText to generate word embeddings [1]. Then, eight classification algorithms were used to classify Arabic text. In our experiment, we will use the same dataset and classifiers used in the previous study to compare the results of using word2vec to those generated using factText. The dataset consists of 1,909 articles belonging to five categories: health, sport, beauty, technology, and work.

This paper is organized as follows: section 2 covers the literature review. Then, section 3 explains the overall methodology we will follow in our experiment. Last, in section 4 is the experimental design and results.

2. Literature Review

Arabic text is used in many applications under the field of natural language processing [7]. Text is classified to determine its actual meanings; word embeddings, which are word representations in low-dimensional space, are produced to capture the syntax and semantic meanings of the text. They can be generated using different techniques. Word2vec is one of the techniques proposed by Mikolov et al. [6], which generates a vector for each word with higher accuracy and lower computational costs than previous techniques [8].

Semantic natural language processing uses the cosine similarity to measure the similarity between two vectors. Word2vec was applied because it is recommended more than the legacy method TF-IDF. However, word2vec embedding must be trained, which requires a big data sample to generate accurate results [9].

Word2vec technique was also used with neural network model to identify similarities between user profiles and to determine the symmetries in word vectors [10]. Word2vec was used along with Metadata as inputs in movie recommendation systems [11]; it shows a better performance than the Singular Value Decomposition (SVD) and item2Vec methods.

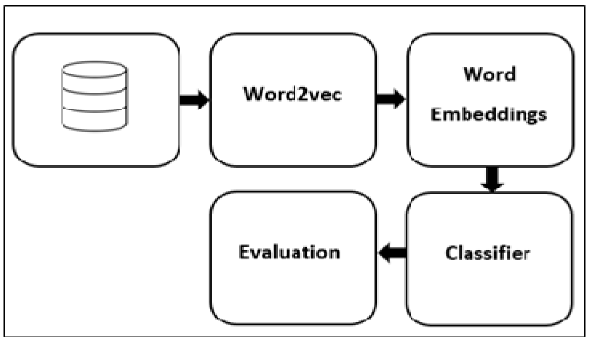
A hybrid model that integrates embedding via word2vec in recommender systems was proposed in [12]. The proposed model employed detailed descriptions rather than meta-data only, which will enrich the obtained data and lead to better recommendations.

Word2vec technique was used with deep neural networks (DNN) to solve cold start problem in E-commerce [13]. The issue appears in many common situations, such as like suggesting new products or similar products and or targeting specific users for offers or discounts. Word2vec was used to convert the textual information into latent vectors, which in response increases the accuracy of the suggestions provided to the user. Word2vec was also used with deep learning to propose more general approach to solve the cold start problem in recommender systems [14].
Security is another area that used word2vec technique to propose a model that finds similarities between various reviews for spam detection [15]. The results of Word2vec were used with support vector machine (SVM) to classify the text as a spam or not.

Some work has been conducted to compare the performance of fastText and work2vec. Erdinc and Guran compared the performance using Turkish documents [16], and another research by Kang and Yang was conducted to compare the differences between them. They found that word2vec technique performed better than fastText in terms of sentiment analysis tasks [17]. To the best of our knowledge, no prior work used Arabic dataset to compare the accuracy of classifier algorithms with Word2vec and fastText.

3. Methodology

To replicate the study that was performed previously using fastText word embedding model [1], we followed the same methodology to train and evaluate the classification models. First, we applied word2vec model to generate word embeddings. Word embeddings are generated at words level creating one vector for each word. Then, the average of words embeddings is computed for each article to return one embedding representing the whole article. This value was used to train the classifier model; We used a set of well-known classifiers following



the previous work by Alghamdi and Assiri [1]. Lastly, classifiers were evaluated based on their accuracy and F1 scores. The overall methodology is illustrated in Figure 1.

Fig. 1 Overall Methodology

4. Experimental Study

We conducted a controlled experiment that aims to study the research hypothesis that fastText technique generates better Arabic word embeddings than word2vec model. The following is the null and alternative hypotheses:

H0: The performance of fastText technique and Word2vec technique are the same.

H1: The performance of fastText technique is better than that of word2vec technique.

4.1 Dataset

We used an Arabic-labeled dataset that was collected from Arabic Wikipedia Articles using Wikipedia API. It consists of 1,909 articles belonging to five categories, which are health, sport, beauty, technology, and work. Data is already cleaned; stop words and unrelated words in the articles were removed. Therefore, no additional pre-processing was applied. Data can be found here [1].

4.1 Word Embeddings

A pre-trained Word2vec model was used to generate word embeddings[2]. The model was trained on Wikipedia articles with 300-vector length. We used this model as an equivalent to the pre-trained fastText model that was used in the previous study [1].

We generated the embeddings at the word level for each article; then, we computed the average embedding to represent each article with a single embedding. However, we could not generate one embedding for each article because Word2vec does not support sentence embeddings.

4.3 Classifiers

We used Python and the Scikit-learn library [18] to implement the models. We applied same set of classification algorithms as in the previous study: Gaussian Naive Bayes (GNB), Logistic Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Nu-Support Vector Classifier (Nu-SVC), Decision Tree, Multi-layer Perception (MLP), and Gaussian Process Classifier (GPC). Then, 10-fold stratified cross-validation with default parameters [18] was used for validation (Figure 2).

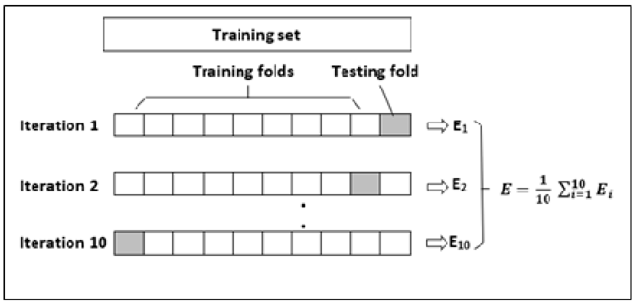


Fig. 2 10-folds cross-validation [19]

4.4 Evaluation

To evaluate the performance of the classifiers using word2vec embedding model and compare it to the classifiers when fastText model was used, we replicated the word embedding process by using the average embedding as the input to train the classification algorithms, following the approach in [1]. As mentioned earlier that we were unable to generate sentence embeddings using word2vec; thus, we could not replicate this part of the study. We used two metrics: accuracy and F1 score as shown in equations 1 [20] and 2 [21].

$$Accuracy = (T_P + T_N)/(T_P + T_N + F_P + F_N) \quad (1)$$
$$F1 = 2 * (Precision * Recall)/ (Precision + Recall) \quad (2)$$
$$Precision = T_P/(T_P + F_P) \quad (3)$$
$$Recall = T_P/ (T_P + F_N) \quad (4)$$

T_P denotes the number of true positives, which is the true class i events that are successfully predicted, and T_N denotes the number of true negative, which are the events that are not class i and were not predicted as class i. F_p and F_n are the numbers of falsely predicted events and falsely not predicted events, respectively. F1 score is the harmonic mean of precision (equation 3) and recall (equation 4). The metrics were computed independently for each class, and then the average was computed.

4.5 Results and Discussion

The results of the 10-fold cross-validation evaluation are shown in detail in Table 3. The average accuracy and F1 score compared to the results found by the previous work [1] are shown in Tables 1 and 2. As shown, the average accuracy and F1 score using word2vec model are slightly less than those found when using fastText technique. However, the difference is not significant. These results were expected, as fastText is a better alternative to word2vec as Mikolov et al. showed in their work [6]. Based on their work, fastText overcome the limitation of word2vec embedding technique, as word2vec cannot generate embeddings for out-vocabulary (OOV) words, which are the words that did not appear in the training step of the embedding model.

However, we found that word2vec has almost the same results as fastText, although we did not perform the stemming phase before generating word2vec embeddings. One reason for such finding could be due to the fact that word2vec technique was trained on all Arabic Wikipedia articles, which was also the source of the collected dataset. In other words, all existing words in the dataset were previously used in the training of word2vec technique, allowing word2vec to generate its embeddings. Therefore, for new words, word2vec might be unable to generate embedding as with fastText. To conclude, based on our experiment, we accept the null hypothesis: *the performance of fastText and Word2vec techniques are the same.*

[1] https://github.com/Nuhagh/MastersThesis

[2] https://github.com/rozester/Arabic-Word-Embeddings-Word2Vec

Table 1 Average Accuracy Comparison Results

Model	Avg Accuracy %(Word2Vec)	Avg Accuracy % (fastText)
GNB	74.58	76.84
LR	86.69	87.74
SVM	85.85	87.63
KNN	80.98	84.61
NuSVC	90.71	82.02
Decision Tree	70.87	72.54
MLP	86.48	87.26
GPC	87.05	87.84

Table 2 Average F1 Score Caomparison Results

Model	Avg F1 Score %(Word2Vec)	Avg F1 Score % (fastText)
GNB	74.42	76.95
LR	85.95	87.94
SVM	86.15	87.86
KNN	81.23	85.27
NuSVC	80.96	82.42
Decision Tree	71.12	72.82
MLP	86.40	87.30
GPC	87.30	88.12

5. Conclusion

In conclusion, this experimental study was conducted to compare the quality of the word embeddings generated by word2vec and fastText embedding techniques using an Arabic dataset while controlling all factors that might affect the obtained results. We evaluated the embeddings through extrinsic evaluation using a multi-class classification algorithm. We evaluated word2vec embeddings and compared the results with the results of the previous study that evaluated the fastText embedding model [1]. We found that the results are almost the same, although fastText models have proven that they overcome the limitation of Word2Vec techniques. We suggested the reason for the similar results in the discussion section.

References

[1] N. Alghamdi and F. Assiri, "Solving the cold-start problem in recommender systems using contextual information in arabic from calendars," Arabian Journal for Science and Engineering, pp. 1–9, 2020.
[2] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in Recommender systems handbook. Springer, 2011, pp. 1–35.
[3] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "Aravec: A set of arabic word embedding models for use in arabic nlp," Procedia Computer Science, vol. 117, pp. 256–265, 2017.
[4] A. A. Altowayan and L. Tao, "Word embeddings for arabic sentiment analysis," in 2016 IEEE International Conference on Big Data (Big Data). IEEE, 2016, pp. 3820–3825.
[5] A. Dahou, S. Xiong, J. Zhou, M. H. Haddoud, and P. Duan, "Word embeddings and convolutional neural network for arabic sentiment classification," in Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers, 2016, pp. 2418–2427.
[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.
[7] G. G. Chowdhury, "Natural language processing," Annual review of information science and technology, vol. 37, no. 1, pp. 51–89, 2003.
[8] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," arXiv preprint arXiv:1310.4546, 2013.
[9] P. Rodriguez Bertorello, "Recommendation engine: Semantic cold start," Available at SSRN 3655839, 2020.
[10] F. Anwar, N. Iltaf, H. Afzal, and H. Abbas, "A deep learning framework to predict rating for cold start item using item metadata," in 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). IEEE, 2019, pp. 313–319.
[11] Y. C. Yoon and J. W. Lee, "Movie recommendation using metadata based word2vec algorithm," in 2018 International Conference on Platform Technology and Service (PlatCon). IEEE, 2018, pp. 1–6.
[12] F. Anwaar, N. Iltaf, H. Afzal, and R. Nawaz, "Hrs-ce: A hybrid framework to integrate content embeddings in recommender systems for cold start items," Journal of computational science, vol. 29, pp. 9–18, 2018.
[13] H. Wang, D. Amagata, T. Makeawa, T. Hara, N. Hao, K. Yonekawa, and M. Kurokawa, "A dnn-based cross-domain recommender system for alleviating cold-start problem in e-commerce," IEEE Open Journal of the Industrial Electronics Society, vol. 1, pp. 194–206, 2020.
[14 J. Yuan, W. Shalaby, M. Korayem, D. Lin, K. AlJadda, and J. Luo, "Solving cold-start problem in large-scale recommendation engines: A deep learning approach," in 2016 IEEE International Conference on Big Data (Big Data). IEEE, 2016, pp. 1901–1910.
[15] X. Wang, K. Liu, and J. Zhao, "Handling cold-start problem in review spam detection by jointly embedding texts and behaviors," in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 366–376.
[16]H. Y. Erdin and A. Güran, "Semi-supervised turkish text categorization with word2vec, doc2vec and fasttext algorithms," in 2019 27th Signal Processing and Communications Applications Conference (SIU). IEEE, 2019, pp. 1–4.
[17] H. Kang and J. Yang, "Performance comparison of word2vec and fasttext embedding models," (J. DCS), vol. 21, no. 7, pp. 1335–1343, 2020.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R.Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
[19] S. Raschka, Python Machine Learning. Packt Publishing, 2015.
[20] D. L. Olson and D. Delen, Advanced data mining techniques. Springer Science & Business Media, 2008.
[21] Y. Sasaki et al., “The truth of the f-measure. 2007,” URL: https://www. cs. odu. edu/˜mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07. pdf [accessed 2021-05-26], 2007.

Fatmah Assiri Associate professor in the College of Computer Science and Engineering at University of Jeddah, KSA. Received her BC from King Abdulaziz University, Saudi Arabia, and Msc. and Ph.D in Computer Science from Colorado State University, United States. Her research interests are software testing and validation, big data, and machine learning.

Nuha Alghamdi Lecturer of Information Technology in the College of Computer Science and Engineering, University of Jeddah, KSA . Held Msc. in Information Technology from King Abdulaziz University, Jeddah, KSA. Her current research interest is Data Science and NLP specifically for Arabic Language.

Table 3: Cross-Validation Scores Using Eight Models with Average Word Embeddings *

	GNB		LR		SVM		KNN		NuSVC		Decision Tree		MLP		GPC	
Fold no.	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
1	76.96	77.17	84.81	85.44	84.29	84.74	82.72	83.05	80.62	81.08	68.58	69.24	85.86	86.00	86.38	86.93
2	74.34	73.84	81.67	81.87	85.86	86.27	79.58	79.98	80.10	79.70	74.34	74.56	87.95	88.00	86.38	86.54
3	76.96	76.69	90.05	90.27	89.52	89.70	82.72	83.19	84.29	84.70	71.72	72.44	90.05	90.00	90.05	90.32
4	77.48	77.22	84.81	85.23	83.76	84.26	78.53	78.72	81.67	81.84	71.72	71.51	85.34	85.00	84.29	84.89
5	74.86	75.48	86.91	87.28	86.38	86.83	85.34	85.84	80.62	81.47	74.86	75.04	87.43	87.00	87.95	88.27
6	70.15	70.20	81.67	82.03	81.15	81.53	77.48	77.69	75.91	76.65	63.87	63.96	80.62	81.00	82.72	82.84
7	75.91	75.59	84.29	84.21	85.34	85.43	80.62	80.68	82.19	82.11	65.96	65.94	82.19	82.00	86.91	87.02
8	74.34	73.38	91.62	91.45	89.00	88.94	83.76	83.65	80.10	79.88	73.82	74.40	92.14	92.00	91.09	91.08
9	71.72	71.21	84.81	85.26	83.76	84.21	79.58	80.04	82.72	82.61	70.68	71.03	86.91	87.00	87.43	87.78
10	73.15	73.46	86.31	86.53	89.47	89.64	79.47	79.50	78.94	79.58	73.15	73.11	86.31	86.00	87.36	87.35
Average	74.58	74.42	85.69	85.95	85.85	86.15	80.98	81.23	80.71	80.96	70.87	71.12	86.48	86.40	87.05	87.30
Std. Dev.	2.39	2.43	3.20	3.11	2.79	2.67	2.52	2.56	2.28	2.16	3.67	3.73	3.37	3.30	2.44	2.38

* All are percentages.