Factors affecting sentence similarity and paraphrasing identification

Marwah Alian & Arafat Awajan

International Journal of Speech Technology

ISSN 1381-2416

Int J Speech Technol
DOI 10.1007/s10772-020-09753-4

Springer

International Journal of Speech Technology
https://doi.org/10.1007/s10772-020-09753-4

Factors affecting sentence similarity and paraphrasing identification

Marwah Alian[1,2] · **Arafat Awajan**[2]

Received: 24 February 2020 / Accepted: 3 September 2020

Abstract
Sentence similarity determines whether two sentences are close in their structure and meaning. The detection of sentence similarity can be affected by several factors such as sentence representation, similarity measure, and words weighting function. In this study, the impact of three factors that influence similarity detection and paraphrasing identification is evaluated using clustering algorithms. In the evaluation of the impact of these factors, we tried different word embedding models, clustering algorithms, and weighting methods for the context words. The clustering algorithms are applied to an Arabic paraphrasing benchmark that consists of 1010 pairs of Arabic sentences constructed on the basis of Arabic transformation rules and labeled for similarity and paraphrasing. Experimental results show that pre-trained embedding, weighting context words with part of speech, and labeling sentence pairs by the majority of experts provides better recall and precision.

Keywords Paraphrasing identification · K-means clustering · Agglomerative clustering · Evaluation · Sentence similarity

1 Introduction

The estimation of similarity between two parts of text either as words, sentences, or documents is an essential part of many Natural Language Processing (NLP) applications such as text summarization, question answering, information retrieval, and document clustering (Klavans et al. 1999).

Semantic similarity is the score that represents semantic relations between two texts, such that the higher the score value, the more similar the meaning of the two texts (Alian and Awajan 2018). Deciding whether two texts have a qualitative semantic relation between them is a challenging task. A semantic relation between two texts could be a paraphrase relation or an entailment relation. In the former, the two texts share the same meaning, whereas in the latter, a text is inferred from the other one (Lintean and Rus 2012).

Paraphrasing is the process of representing a sentence with different words and structure to produce a new sentence (Fernando and Stevenson 2008; Awajan and Alian 2020). Paraphrasing may be used to exhibit a good understanding of what has been read by rewriting it with new words or structure given that the original text is referenced, otherwise the result will be considered a type of cheating or plagiarism.

Paraphrase identification involves the detection of different linguistic phrases or expressions with similar meaning. Conversely, determining the degree of similarity is part of the semantic similarity task (Jaradat et al. 2017). Semantic similarity is also an essential part of paraphrase detection, as it measures the similarity score between texts to determine whether they are similar in meaning or not (Alian and Awajan 2018; Srivastava and Govilkar 2017).

Several approaches have been developed to measure the semantic similarity between sentences. Those approaches can be divided into four main categories: co-occurrence–based, statistical corpus-based, feature-based and word embedding-based techniques (Alian and Awajan 2020). In co-occurrence–based approaches, the text is represented as a bag of words. In a statistical corpus-based approach, latent semantic analysis is applied to represent texts as vectors in a reduced dimensional space. Feature-based approaches focus on two similarities: the similarity of words and of word order. Finally, word embedding approaches consider the context of the words when representing words in the distributional space (Alian and Awajan 2018).

The aim of this research is to examine the impact of factors that affect similarity detection and paraphrasing

✉ Marwah Alian
marwah2001@yahoo.com

Arafat Awajan
awajan@psut.edu.jo

[1] Hashemite University, Zarqa, Jordan

[2] Princess Sumaya University for Technology, Amman, Jordan

Published online: 16 September 2020

identification. Our approach is developed on the basis of using clustering algorithms to analyze sentence distribution for depicting sentence similarity.

The main factors to be studied to show their impact include the use of word embedding, word weighting methods, and the technique employed for dataset labeling. The implementation of word embedding is considered by two methods: self-training embedding and pre-trained embedding. The two methods for word weighting are also evaluated, as the weight reflects the importance of the context word, Tf-idf, and part of speech (POS) weighting. Another aspect that may influence the results of similarity detection and paraphrasing identification as evaluated by an experiment is the way in which the dataset is labeled for similarity either by a majority of experts or by a similarity threshold.

To analyze these factors, we use an Arabic paraphrasing benchmark which consists of 1010 Arabic sentence pairs. The first part of each pair is a sentence taken from an Arabic book or generated by Arabic experts from words taken from Arabic lexicons or the AWSS dataset (Oshea et al. 2013). These sentences are transformed into other sentences using Arabic transformation rules. This benchmark is labeled by Arabic students at different levels of the Art College of Hashemite University (Alian et al. 2019).

Two main types of clustering algorithms include the hierarchical and partitioning algorithms. One of the most commonly used partitioning algorithms is K-means, which is considered the simplest method for partitioning a dataset into clusters of similar objects. K-means is widely used for text clustering because of its ability to converge to a local optimum, although it is utilized with a massive data matrix. The goal of K-means is to make the distance between objects that belong to the same cluster as short as possible (Naeem and Wumaier 2018).

For hierarchical algorithms, one of the most popular approaches used for text clustering is agglomerative clustering, which calculates the similarity between all data points in all clusters and then combines the most similar points in a cluster (Froud and Lachkar 2013).

In this paper, we discuss how K-means clustering works for Arabic sentence similarity, investigate the impact of three factors affecting sentence similarity detection, and then compare the results with the agglomerative clustering algorithm. Both clustering algorithms are applied to an Arabic paraphrasing benchmark through which the impact of the three factors is evaluated using recall, precision, and F-measure.

This paper is organized as follows. Related studies are discussed in Sect. 2. The structure of the Arabic paraphrasing benchmark is explained in Sect. 3. A description of the pre-trained word embedding used in this research is provided in Sect. 4. The process of evaluating the three factors is provided in Sect. 5. The experiments are explored and discussed in Sect. 6, followed by the conclusion in Sect. 7.

2 Related work

Several researchers have proposed different methods in the field of data mining to determine features or patterns in data and categorized data points into clusters or groups according to their similarity and on the basis of those patterns (Naeem and Wumaier 2018). However, with the increase in the amount of available online texts, text clustering becomes an important task in obtaining good results for many NLP applications, such as text mining, information retrieval, and many other applications (Froud and Lachkar 2013).

For example, Lydia et al. (2018) have proposed clustering identical documents in interrelated folders and minimizing the complexity of searching for a document. They use K-means clustering and document preprocessing to group similar documents into a single cluster where the similarity between documents is computed based on weighted terms similarity. The proposed work consists of a number of steps: preprocessing, terms weighting, features extraction and clustering. The weighting step is based on the Tf-idf while the pre-processing step is to remove stop words that are considered to be noise that affects the results of clustering. Then the extraction step of the features is performed by applying stemming to the document terms and then determining the Tf-idf weights for each term to filter the features to the maximum weighted terms. Finally, K-means is applied to partition the dataset into groups of similar documents.

Preprocessing methods, such as stemming, are studied by Bsoul and Mohd (2011) with five similarity measures to evaluate their impact on the performance of Arabic document clustering. The results demonstrate that, although these preprocessing methods enhance precision and recall results, a large amount of noise will be found in the document representation by grouping words that are not semantically similar to the same stem or failing to group semantically similar words to the same stem.

The work of Froud and Lachkar (2013) provides a study of the agglomerative hierarchical algorithm using different linkage methods and a number of distance/similarity measures, such as Cosine Similarity, Euclidean Distance, Pearson correlation and Jaccard coefficient. The efficiency of using these measures is tested on clustering Arabic documents. They also investigate the effect on document clustering when applying stemming to the terms in the documents. The results of this study show that the ward linkage outperforms other linkage methods while applying stemming provides faster clustering and produces smaller document representation.

While the study of Alkoffash (2012) compares the performance of K-means and Kmediods algorithms using a labeled set of documents with their identified clusters. The dataset consists of 242 predefined clustered documents. Keywords

are extracted as a feature set to enhance the performance of document clustering. The results of the comparison show that the two algorithms can perform well for Arabic documents and the average precision and recall of Kmediods is better than Kmeans results.

Hussein et al. (2016) try to overcome the problem of the high dimensionality of vectors representation by reducing the feature set to have only key-phrases that represent each document. They experiment different types of similarity measures such as string-based, knowledge-based and corpus-based similarity measures. The results show an improvement in accuracy and the complexity of the hierarchical clustering algorithm is reduced.

The use of neural embedding methods with clustering algorithms has shown a significant improvement in the performance of document clustering. For example; Rahaman and Hosein (2017) have proposed to use Chinese Restaurant algorithm with Gaussian word embedding for clustering Arabic documents. Words are represented using Gaussian word embedding without the need of external information. The results of the proposed method provide more coherent clusters than those provided by traditional K-means.

Soliman et al. (2019) used K-means with word embedding models to improve document clustering accuracy. They evaluated their work by applying Word2Vec representation to the terms in Arabic news documents and utilizing Euclidean distance as a normalization of the length of document vectors. The results show that the use of embedding models with K-means clustering outperforms the traditional K-means with Tf-idf weighting in terms of precision, recall, F-measure, purity, and other measures.

The literature motivated the current authors to apply K-means to an Arabic paraphrasing benchmark to study the impact of different factors affecting sentence similarity detection and paraphrasing identification through using different word embedding models, weighting methods, and dataset labeling methods.

3 Arabic paraphrasing benchmark

Arabic Paraphrasing benchmark consists of pairs of Arabic sentences. The first sentence of each pair is either collected from books used to teach Arabic, such as Jordanian Arabic curriculums and lexicons (Jarim and Ali 2004; Omar 1998, 1420; Alkholi 2001) or generated by Arabic experts based on Arabic Word Semantic Similarity (AWSS) dataset (Oshea et al. 2013). The second part of each pair is a transformed sentence from the first one. The two experts who collect and transform the sentences have a doctorate of philosophy (PhD) in Arabic literature and a good experience in teaching graduate students.

Paraphrasing in Arabic texts is based on the hypothesis: "two sentences are paraphrased if they consist of identical words except one word in the first sentence and its synonym in the second one". In addition, paraphrased sentences can be generated on the basis of the Arabic transformation rules (Al-Kholi 1999) that have been construed by Chomsky (1957).

The transformation rules are made up of six rules, namely: permutation, deletion, addition, reduction, expansion, and replacement. Permutation is the process of changing the order of words in the first sentence to get a transformed sentence while deletion removes one word from the sentence. The reduction rule replaces two words with one word with the same meaning. In the addition rule, a word or phrase is added to the structure of the sentence while in the expansion rule; a word is replaced by two words or phrases of the same meaning (Awajan and Alian 2020; Alian et al. 2019).

Suppose that A, B, and C are words or phrases in a sentence. The transformation rules that would be applied to the sentence in order to provide another sentence with or without the same meaning are represented using these symbols in Table 1. These rules are described in more details by Al-Kholi (1999).

4 Pre-trained word embedding

Word representation as vectors in the distributional space, also known as word embedding, has been commonly used in NLP applications. Two pre-trained word embeddings are used to represent context words in the Arabic paraphrasing benchmark: AraVec and FastText. Sentence embedding or sentence vector representation is computed as the mean of its content words embeddings.

4.1 Arabic word embedding (AraVec)

AraVec (Mohammad et al. 2017) is a pre-trained Arabic word representation (i.e., word embedding model). It was produced using the Word2Vec skip-gram technique trained on three domains of Arabic content with a vector dimension of 300 and a vocabulary size of 145,428. Domains with Arabic content articles utilized to build AraVec models include web pages, Arabic tweets, and Wikipedia articles. These domains provide more than 3.3 billion tokens employed for the construction of AraVec models.

In addition, AraVec is available online with vector dimensions of 100 and 300, as well as skip-gram and CBOW for

Table 1 Transformation rules (Alian et al. 2019)

Transformation rule	Representation by symbols	English translation	Arabic example
Permutation	A+B = B+A	The winner gets the prize. The prize was gotten be the winner	تسلم الفائز الجائزة تسلم الجائزة الفائز
Deletion	A + B = […] + B A + B = A + […]	Ask the villagers about the thief. Ask the village about the thief	اسألوا أهل القرية عن اللص اسألوا القرية عن اللص
Addition	A=A+B	A clear Sky The sky is clear	السماء صافية إن السماء صافية
Expansion	A = B+C	I want it to rain I would like it to rain	وددت نزول المطر وددت لو ينزل المطر
Reduction	A + B = C	The weather is hot cold The weather is fair	الجو حار بارد الجو معتدل
Replacement	A = B	The professor participated in the literary evening The professor participated in the evening of poetry	شارك الأستاذ في الأمسية الأدبية شارك الأستاذ في الأمسية الشعرية

The words to which the rules have been applied are underlined

word vectors trained on Twitter and Wikipedia Arabic content.[1]

4.2 FastText embedding

Grave et al. (2018) contributed in a pre-trained word vector representation for 157 languages including Arabic. The word vectors have been trained on Wikipedia and the Common Crawl corpus using an extension of the FastText model with subword information. Word representations were generated using character ngrams, with each ngram represented as a vector and then taking the sum of the character ngram vectors to obtain the vector representation of the word. Although the word with its full content characters is included as part of the character ngrams, the FastText model still learns one vector for each word (Grave et al. 2018; Bojanowski et al. 2017). The pre-trained FastText word embedding models for Arabic and other languages are available online for 300-dimensional word vectors.[2]

5 Factor evaluation process

The impact of three factors that may affect the similarity detection between sentences and paraphrasing identification, is studied and evaluated by applying clustering algorithms. These factors are the vector representation of words, the weighting function of words in a sentence, and the way in which the dataset is labeled.

The evaluating method for the three factors is shown in Fig. 1. Figure 1a describes the first factor assessment process where the first step is to obtain words embeddings from self-trained embeddings or pre-trained embeddings.

Self-trained embeddings are obtained from training the Word2Vec model on context words in the Arabic paraphrasing benchmark. Pre-trained embeddings are loaded, and then context words embeddings are obtained. Two pre-trained models are used: AraVec and FastText.

The sentence embedding is computed as the mean of its content words vectors. Words without embeddings in a pre-trained model are skipped in the sentence embedding computation.

The second factor evaluation as shown in Fig. 1b depends on the best results achieved by the word embedding evaluation. Each word embedding is multiplied by the word weight which reflects the importance of a word within a sentence.

[1] https://github.com/bakrianoo/aravec.

[2] https://fasttext.cc/docs/en/crawl-vectors.html.

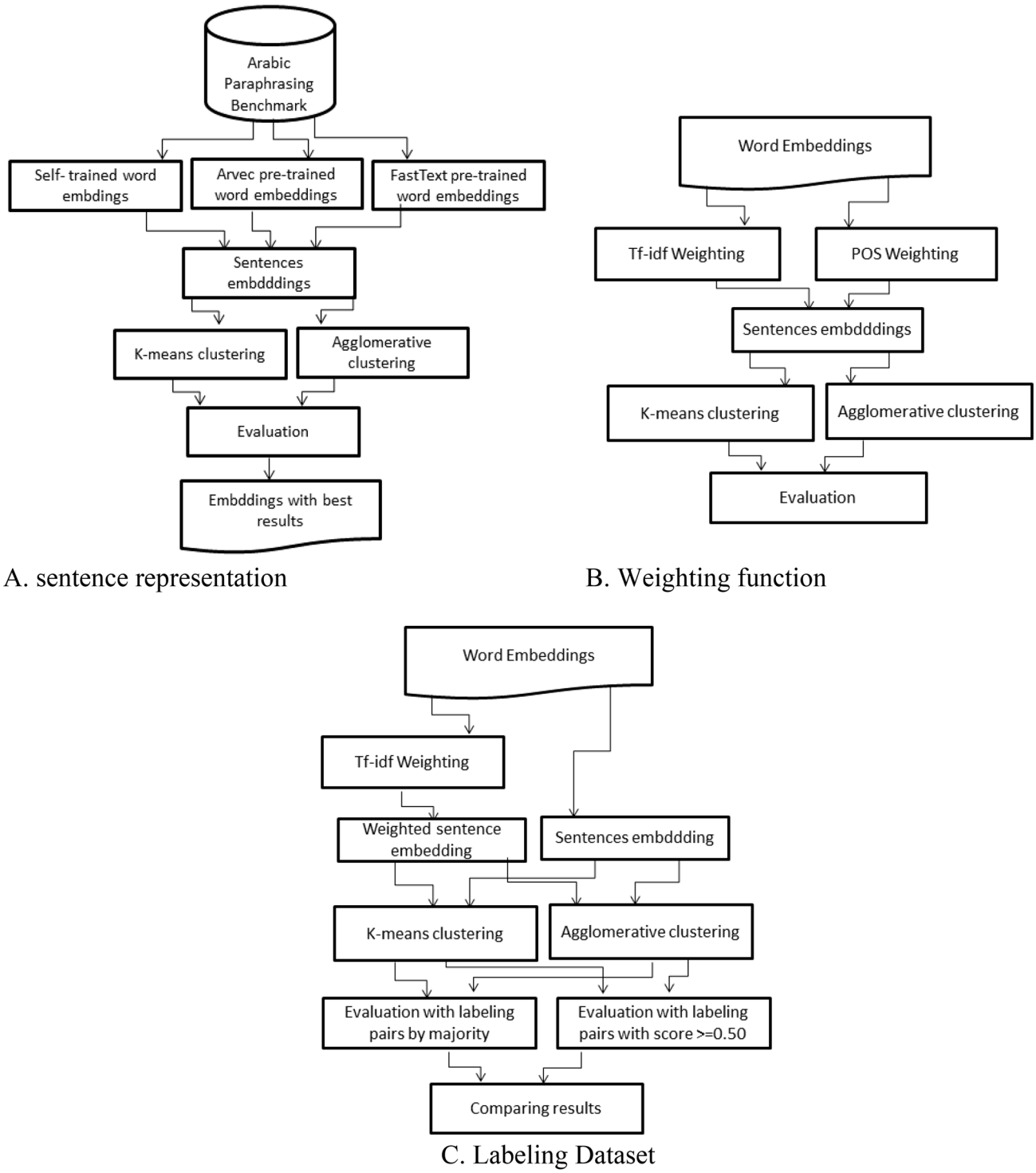

Fig. 1 Evaluation of factors affecting sentence similarity with K-means and Agglomerative clustering

Two weighting methods are used: Tf-idf weight and POS weight.

Tf-idf is increased proportionally with the increase in the frequency of the word in the document but is inversely proportional to the word's frequency in the corpus (Mohsen et al. 2018). The POS weight of each word in a sentence is assigned to the POS tag. To obtain these tags, the Stanford POS tagger for the Arabic language is used. Then, the weight for each type of tag is assigned according to the weights introduced by Nagoudi et al. (2017): verb = 0.4, noun = 0.5,

adjective = 0.3, and preposition = 0.1. We assigned a weight of 0.2 to the adverb and personal pronoun tags.

The label of similarity for a sentence pair is assigned using two methods: labeling by a majority of experts and by a score threshold. Labeling by majority provides a sentence pair with a similar label if three of the five experts decide that the sentence pair is similar, otherwise a sentence pair is labeled as non-similar.

Labeling according to a score threshold characterizes a pair of sentences as "similar" if the average score of similarity given by the experts is greater than or equal to 0.5. Otherwise, the sentence pair is labeled as non-similar. The evaluation of the way the dataset is labeled is shown in Fig. 1c.

After the step of representing sentences as vectors in the distributional space, K-means and agglomerative clustering algorithms are applied to these vectors with the following parameters: cosine similarity measure, complete linkage, and the number of clusters set to 1010 for the 1010 sentence pairs in the dataset. If the two parts of the sentence pair are grouped together in one cluster, then they are detected by the clustering algorithm to be similar; otherwise, they are considered dissimilar.

The average silhouette_score for all K-means and agglomerative experiments has been recorded in the range [0.48–0.53]. These values are closer to 1, an outcome which indicates that a sentence or data point is very similar to the other sentence or data point in the cluster and is far away from the neighboring clusters (Joshi 2017). Then, the evaluation is conducted using three metrics: precision, recall, and F-measure.

Table 2 Similarity confusion matrix for Self-trained

(Obtained)	Labeled (actual)	
	Similar	Not similar
Same cluster	638	95
Different cluster	226	51

Table 3 Similarity confusion matrix for the AraVec pre-trained embeddings

(Obtained)	Labeled (actual)	
	Similar	Not similar
Same cluster	699	105
Different cluster	156	41

Table 4 Similarity confusion matrix for the FastText pre-trained embeddings

(Obtained)	Labeled (Actual)	
	Similar	Not similar
Same cluster	611	100
Different cluster	253	46

Table 5 The confusion matrix for paraphrasing identification with Self-trained embeddings

Obtained	Labeled	
	Paraphrased	Not paraphrased
Paraphrased	576	157
Not paraphrased	188	89

6 Experiment and results

Several experiments are conducted to investigate the impact of three factors affecting the detection of sentence similarity and the identification of paraphrasing. All experiments are performed on the basis of K-means and agglomerative clustering and then evaluated using recall, precision, and F-measure.

In the first factor evaluation experiments, the effect of word embedding is tested for which three word embedding models are used: self-trained embedding, AraVec, and FastText. In the second factor evaluation experiments, the impact of the word weighting method is explored where the Tf-idf weight and POS weight are tested. In the third factor experiments, the way in which the dataset is labeled is investigated, and two labeling methods are considered: labeling by the majority of experts and by a score threshold.

The resulting clusters containing similar sentences are compared with the labeled dataset to form the confusion matrix for each experiment and then the recall, precision, and F-measure are determined. The following subsections describe and analyze the experiments conducted.

6.1 Word embedding method

Three experiments are performed to evaluate the method used to represent words in the distributional space. The first experiment is based on self-training embeddings, in which we train the Word2Vec model with a vector dimension of 300 and a window size of 5 on the Arabic paraphrasing benchmark that consists of 2020 sentences with 3145 vocabulary terms.

The second experiment is performed on pre-trained embedding using the AraVec model and employs (77,600,000) Arabic tweets collected from different locations (Mohammad et al. 2017). The AraVec model has (1476,715) vocabulary terms. The vectors of words in the

Table 6 The confusion matrix for paraphrasing identification with Aravec embeddings

Obtained	Labeled	
	Paraphrased	Not paraphrased
Paraphrased	611	173
Not paraphrased	148	78

Table 7 The confusion matrix for paraphrasing identification with FastText embeddings

Obtained	Labeled	
	Paraphrased	Not paraphrased
Paraphrased	543	168
Not paraphrased	216	83

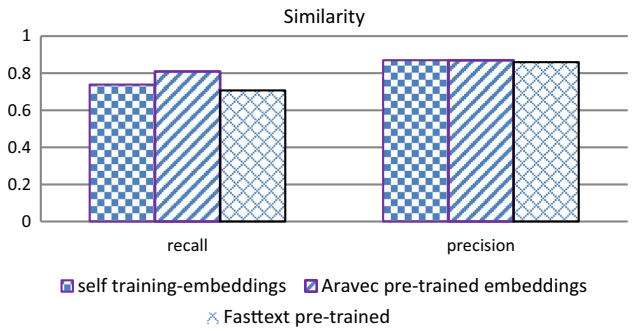

Fig. 2 Recall and precision for self-trained and pre-trained embeddings for similarity

vocabulary are available online with dimensions of 100 and 300.

In the third experiment, we use the FastText pre-trained embeddings with a dimension of 300 and a vocabulary size of (2,000,000).

In all experiments, the sentence embedding is constructed as the mean of its content words' vectors derived from the models mentioned above. Then, K-means is used to partition the dataset into clusters of similar sentences. The clustering parameters are as follows:

- The number of clusters (K) is chosen as half the number of sentences which is equal to 1010 clusters, because we compare each pair of sentences if they are similar, paraphrased or not.
- The linkage approach is "complete" or "maximum."
- The similarity measure is the cosine similarity.

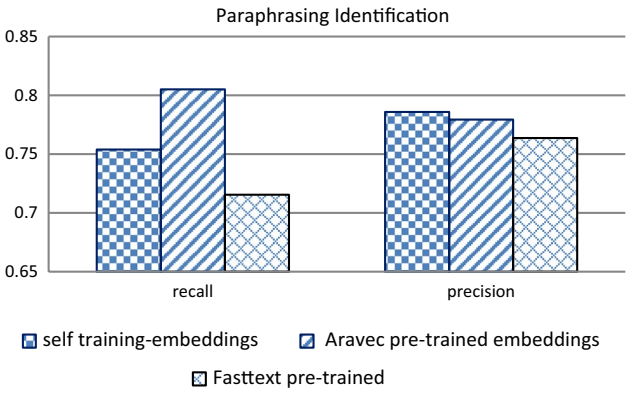

Fig. 3 Recall and precision for Self-trained and pre-trained embeddings for paraphrasing identification

Tables 2, 3, and 4 show the confusion matrices inferred from Alhawarat and Hegazi (2018) to represent the results of the use of self-trained, AraVec, and FastText embeddings, respectively, with k-means clustering for sentence similarity detection.

The confusion matrices for paraphrasing identification with self-trained embeddings, AraVec, and FastText embeddings are represented in Tables 5, 6, and 7, respectively.

The results of recall and precision for the detection of similar sentences are shown in Fig. 2, where the use of AraVec pre-trained embeddings provides better recall and precision compared to other methods.

The results of paraphrasing identification are shown in Fig. 3 where the pre-trained AraVec model outperforms other embedding models in terms of recall whereas the self-trained model provides a better precision value.

In detecting similar sentences, the results of K-means with pre-trained embedding are compared to the agglomerative clustering (for the same clustering parameters) in terms of recall, precision, and F-measure as shown in Table 8. The performance of K-means is closer to that of agglomerative clustering with the AraVec model in terms of the F-measure value, whereas the FastText model achieves less value.

6.2 Word weighting method

Each word in the structure of a sentence has a significant importance, and this is implemented as a weight given to each word using a weighting method. In this study, two methods of weighting are tested: Tf-idf weighting and POS weighting. In Tf-idf weighting, the weight is computed based on the frequency of the word in the sentence and the corpus, while in POS weighting, the weight is given to the POS tag of the word in the sentence (i.e. verb, noun, adverb, adjective, etc.).

Table 8 Comparing K-means performance with Agglomerative clustering

Clustering algorithm	Embedding model	Recall	Precision	F-measure
K-means	Aravec	0.789351852	0.869897959	0.827669903
	FastText	0.707175926	0.859353024	0.775873016
Agglomerative clustering	Aravec	0.793981481	0.865069357	0.828002414
	FastText	0.736111	0.859459	0.793017

Table 9 The results of adding weight to the words with clustering algorithms

Clustering algorithm	Weighting	Recall	Precision	F-measure
K-means	Tf-idf	0.747685	0.875338753	0.8064919
	POS	0.780093	0.884514	0.829028
Agglomerative clustering	Tf-idf	0.78125	0.869845	0.823171
	POS	0.828704	0.87104623	0.849348

Table 10 Labeling by majority results

Clustering algorithm	Recall	Precision	F-measure
K-means Tf-idf	0.87468	0.790751	0.830601
K-means	0.789352	0.869898	0.82767
Agglomerative Tf-idf	0.781503	0.871134	0.823888

Table 11 The results of labeling by similarity score $\geq 50\%$

Clustering algorithm	Recall	Precision	F-measure
K-means Tf-idf	0.748082	0.80137	0.77381
K-means	0.79863	0.743622	0.770145
Agglomerative Tf-idf	0.793151	0.746134	0.768924

The effect of adding weights to context words on the detection of similarity between sentences is investigated using K-means and agglomerative clustering with Aravec embeddings, and the results of these clustering algorithms are shown in Table 9.

When we include the weight for each word to generate sentence embedding, the results for sentence similarity detection show better recall compared to clustering without including weights. K-means achieves better recall and precision with the POS weighting method compared to Tf-idf weighting. This also provides better performance in terms of precision compared to agglomerative clustering.

Moreover, the results show an improvement in the clustering results for sentence similarity detection, taking into account the weight of words over clustering without weighting.

For paraphrasing identification, K-means with Tf-idf weighting show a very small difference in the recall value over K-means without weighting. It achieves a recall value of 0.785 while the recall value is 0.779 without weighting. Also, it achieves the same precision value of 0.81.

6.3 Labeling dataset

Two methods are used to give the final label to the sentence pair in the dataset: labeling by a majority of experts and labeling by a similarity threshold of 0.50. In order to evaluate the effect of these methods, experiments are carried out using K-means and agglomerative clustering with AraVec embeddings and Tf-idf weighting. The results of labeling by majority are shown in Table 10 while the results of labeling based on the similarity threshold are shown in Table 11.

It is shown that the choice of the way in which the dataset is labeled has an impact on precision and recall results. Table 10 shows that labeling by majority provides a good improvement in the performance of both K-means and agglomerative clustering in terms of F-measure.

In addition, labeling by majority provides better recall using K-means while it provides better precision using agglomerative clustering.

7 Conclusion

Factors affecting sentence similarity and paraphrasing identification are studied by applying different clustering algorithms to an Arabic paraphrasing benchmark that consists of pairs of Arabic sentences with two labels, one for similarity and the other for paraphrasing.

The factors investigated for their impact on sentence similarity detection include a word embedding model used to represent words as vectors, the weighting function that is used to represent the importance of words in a sentence, and the way in which sentence pairs are labeled for similarity. These factors are evaluated using K-means and agglomerative clustering.

Clustering algorithms using AraVec pre-trained embeddings produce better recall value when detecting similar and paraphrased sentences. Conversely, using POS weighting

with word embedding provides better recall and precision for K-means and agglomerative clustering algorithms.

Labeling of sentence pairs by a majority of experts provides a recall of 0.87 and 0.782 for K-means and agglomerative clustering, respectively. By contrast, the recall value resulting from the labeling by a score threshold is 0.74 for K-means and 0.79 for agglomerative clustering. Thus, a better recall is achieved by K-means with labeling by majority. Moreover, labeling by majority provides an enhancement in terms of the F-measure for both clustering algorithms.

References

Alhawarat, M., & Hegazi, M. (2018). Revisiting KMeans and topic modeling, a comparison study to cluster arabic documents. *IEEE Access, 6,* 42740–42749.

Alian, M., & Awajan, A. (2020). Evaluating Factors affecting sentences similarity and paraphrasing identification using K-means clustering. In *The 35th International Business Information Management Association (35th IBIMA)* (pp. 952–959).

Alian, M., & Awajan, A. (2018). Semantic similarity approaches—Review. In *2018 international arab conference on information technology (ACIT2018)*, Werdanye, Lebanon (pp. 1–6).

Alian, M., Awajan, A., Al-Hasan, A., & Akuzhia, R. (2019). Towards building Arabic paraphrasing benchmark. In *The second international conference on data science, E-learning and information systems (DATA' 2019)*, Dubai.

Alkholi, M. A. (2001). *Semantics- Elm AldlAlAh (Elm AlmEnY*) (Vol. 1). Amman: dar Al-falah.

Al-Kholi, M. A. (1999). *Transformation rules for Arabic language*. Jordan: dar Al-Falah.

Alkoffash, M. (2012). Automatic arabic text clustering using K-means and K-mediods. *International Journal of Computer Applications, 51*(2), 5–8.

Awajan, M., & Alian, A. (2020). Paraphrasing identification techniques in English and Arabic texts. In *The 11th international conference on information and communication systems*, Irbid, Jordan (pp. 155–160).

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5,* 135–146.

Bsoul, Q. W., & Mohd, M. (2011). Effect of ISRI stemming on similarity measure for arabic document clustering. In *Asia information retrieval symposium* (pp. 584–593).

Chomsky, N. (1957). *Syntactic structure*. Paris: Mouton Publishers.

Fernando, S., & Stevenson, M. (2008). A semantic similarity approach to paraphrase detection. In *The 11th annual research colloquium of the UK special interest group for computational linguistics*.

Froud, H., & Lachkar, A. (2013). Agglomerative hierarchical clustering techniques for arabic documents. In *Advances in computational science, engineering and information technology. Advances in intelligent systems and computing* (Vol. 225). Heidelberg: Springer.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., & Mikolov, T. (2018). Learning word vectors for 157 languages. In *Proceedings of the international conference on language resources and evaluation* (*LREC 2018*).

Hussein, M., Alsammak, A., & Elshishtawy, T. (2016) In *The 10th international conference on informatics and systems* (pp. 61–67).

Jaradat, M., Al-Ayyoub, Z., Jararweh, M., & Al-Smadi, Y. (2017). Paraphrase identification and semantic text similarity analysis in Arabic news tweets using lexical, syntactic, and semantic features. *Information Processing and Management, 53*(3), 640–652.

Jarim, A., & Ali, M. (2004). *Clear Syntax for Arabic Grammar (AlnHw AlwADH fy qwAEd AllgAh AlErbyAh)* (2nd ed.). Riyadh: Egyptian and Saudi dar for Publishing.

Joshi, P. (2017). *Artificial intelligence with python*. Birmingham, UK: Packt Publishing Ltd.

Klavans, J., Eskin, E., & Hatzivassiloglou, V. (1999). Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. In *SIGDAT conference: empirical methods in NLP and very large corpora* (pp. 204–212).

Lintean, C. M., & Rus, V. (2012). Measuring semantic similarity in short texts through greedy pairing and word semantics. In *The twenty-fifth international florida artificial intelligence research society conference* (pp. 244–249).

Lydia, E. L., Govindaswamy, P., Lakshmanaprabu, S., & Ramya, D. (2018). Document clustering based on text mining K-means algorithm using euclidean distance similarity. *Journal of Advanced Research in Dynamical and Control Systems, 10*(2), 208–214.

Mohammad, A. B. S., Eissa, K., & El-Beltagy, S. R. (2017). AraVec: A set of arabic word embedding models for use in arabic NLP. *Procedia Computer Science, 117,* 256–265.

Mohsen, G., Al-Ayyoub, M., Hmeidi, I., & Al-Aiad, A. (2018). On the automatic construction of an arabic thesaurus. In *9th international conference on information and communication systems* (*ICICS*).

Naeem, S., & Wumaier, A. (2018). Study and implementing K-mean clustering algorithm on english text and techniques to find the optimal value of K. *International Journal of Computer Applications, 182*(31), 975–8887.

Nagoudi, E. M. B., Ferrero, J., & Schwab, D. (2017). LIM-LIG at SemEval-2017 task1: Enhancing the semantic similarity for arabic sentences with vectors weighting. In *11th international workshop semantic evaluation* (*SemEval 2017*) (pp. 134–138).

Omar, A. M. (1420–1999). *Language excercises and grammar* (Vol. 2). Kuwait: Kuwait University.

Omar, A. M. (1998). S*emantics -Elm AldlAlAh* (5th ed.). Cairo, Egypt: Book World.

Oshea, F. A., Bandar, J. D., Crockett, Z., & Almarsoomi, K. (2013). AWSS: An algorithm for measuring arabic word semantic similarity. In *2013 IEEE international conference on systems, man, and cybernetics* (pp. 504–509).

Rahaman, I., & Hosein, P. (2017). Exploiting Gaussian word embeddings for document clustering. In *Future technologies conference (FTC)* (pp. 1015–1018).

Soliman, H. R. H., Grida, M., & Hassan, M. (2019). Arabic text custering based on K-means algorithm with semantic word embedding. *Journal of Theoretical and Applied Information Technology, 97*(21), 2497–2509.

Srivastava, S., & Govilkar, S. (2017). A survey on paraphrase detection techniques for Indian regional languages. *International Journal of Computer Applications, 163*(9), 0975–8887.