Benha Journal of Applied Sciences (BJAS) **print: ISSN 2356–9751**
Vol. (7) Issue (4) (2022), (133-142) **online: ISSN 2356–976x**
http://bjas.journals.ekb.eg

Arabic Semantic-Based Textual Similarity

Shimaa Ismail [1], AbdelWahab Alsammak[2] and Tarek Elshishtawy[1]
[1]Faculty of Computers and Artificial Intelligence, Benha Univ., Benha, Egypt
[2]Faculty of Engineering Shoubra, Benha Univ., Benha, Egypt
E-mail: Shimaa.mustafa@fci.bu.edu.eg

Abstract

Textual similarity is one of the most important aspects of information retrieval. This paper proposes several techniques of semantic textual similarity as well as the factors that influence them. Two-hybrid approaches for measuring the degree of similarity between two Arabic snipped texts are presented. The first proposed approach combined the word-based and vector-based similarity methods to construct semantic word spaces for each word of the input text. These words are represented in their lemma forms to capture all semantically related words. In this approach, the semantic word spaces are used to find the best matching between the input text words, and hence, the degree of similarity between the two snipped texts is computed. The second proposed approach combined semantic and syntactic based approaches. The basic Levenshtein concept represents the main structure for this approach. It has been modified to measure the edit cost at the token level not at the character level. In addition, the semantic word spaces are added to this approach to include the semantic features to the syntactic features. Some techniques are embedded to overcome the syntactic approach problems such as the word sequence. Pearson correlation coefficient is used to measure the degree of correctness of the two proposed approaches as compared to two benchmark datasets. The experiments achieved 0.7212 and 0.7589 for the two proposed approaches on two different datasets.

Keywords: Arabic Text Similarity, Semantic Similarity, Lexical Similarity, Word Embedding, Permutation Feature, Negation Effect.

1. Introduction

STS (Semantic Textual Similarity) metrics have been a major topic in a variety of studies and applications. Information retrieval, machine translation, text summarization, sentiment analysis, question generating and answering, automatic essay scoring, automatic short answer grading, and other activities all rely on them [1].

There are several barriers in determining the semantic similarity or relationship between two snipped Arabic texts, including morphological inflections and orthographic confusion related to optional diacritization. As a result, there are more homographs than in English, which adds to the confusion [2]. The Arabic word "ذهب" for example, maybe the verb "Went" or the noun "Gold" with various diacritics. It is possible to discern between the two meanings based on the part of speech tagging and the context of the word. As a result, it is important to do this work with a conventional morphological tool. In this research, the StanfordNLP library is used as a morphological tool to determine the Part of Speech (POS) tagging for words. In addition, the lemmatization approach was employed in our study to avoid the diacritization difficulty and its impact by using the lemma form of the word. The inflected word form is transformed into its dictionary lemma look-up form through lemmatization [3]. This lemma form is the shortest that captures all the word's semantic characteristics.

There are two different types of sentence similarity: lexical similarity and semantic similarity. Lexical similarity looks at a sentence as a series of characters and determines how similar these characters are. The semantic similarity, on the other hand, is determined by the meaning of the phrases. It is necessary to determine the degree of relatedness between sentences to quantify semantic similarity [4].

The Levenshtein approach is one of the approaches that measure the similarity of texts lexically. The Levenshtein distance is a statistic for calculating how many edit operations are necessary to convert one string to another [5]. The distance between two words is defined as the number of single-character changes (i.e., insertions, deletions, or replacements) necessary to change one word into the other in its original form. The edit distance is expanded in this study to detect distances between two sentences given in words rather than characters. In addition, semantic features are embedded to change the edit cost between the input words.

Semantic word spaces are generated from one of the word embedding models that represent the words in a vector space. This vector space defined the meaning of words and the relation between them. There are two algorithms used to extract these semantic word spaces in this research. In the two proposed approaches, the semantic word spaces are used in two different manners.

In this paper, two hybrid approaches are proposed to enhance the overall calculations of the similarities between two texts. An interlaced approach that combines word-based and vector-based approaches was proposed. A modified strategy that combines both syntax and semantic techniques also presented.

The rest of the paper is organized as follows. Section 2 represents the related work of the semantic similarity approaches. Section 3 explains the approach methodology. In section 4, the evaluations of the experimental results are described. Finally, the conclusion of our approach is presented in section 5.

2. Related Work

Arabic snipped texts can be classified according to the adopted methodology as follows:

One of the most used strategies for evaluating semantic similarity is deep learning with feature-engineered models. Tian et al. [6] used characteristics such as n-gram overlap, edit distance, and longest common prefix/ suffix/ substring to train deep learning algorithms. They were able to reach a PCC of 0.7440. On the aspects of alignment similarity and string similarity measurements, Henderson et al. [7] employed the same method with various algorithms, such as Recurrent Neural Networks (RNN) and Recurrent Convolutional Neural Networks (RCNN). For the same dataset, their PCC is 0.7304.

For the same dataset utilized, the semantic information space (SIS) is a technique that produced a high Pearson correlation coefficient. The non-overlapping Information Content (IC) computation is obtained using this method, which is based on the semantic hierarchical taxonomy in WordNet. This method was employed by Wu et al [8] in three studies. As they used the IC which is based on the word frequencies from WordNet and the British National Corpus. They also collaborate the IDF weighting system using the IC with cosine similarity. The highest Pearson correlation coefficient is presented in this competition by 0.7543.

BableNet is a huge, multilingual semantic network with broad coverage that gathered its data from Wordnet and Wikipedia [9]. The multilingual word-sense aligner proposed by Hassan et al., [10] relies heavily on the BableNet network. According to the Bable synsets, they built an aligner that aligns each word in one phrase to another word in the other. In many languages, these synsets reflect the word's meaning, named entity, and synonyms. For the used Arabic dataset, PCC is 0.7158.

Word embedding is a common method for various text applications and NLP activities. The distributed representation of words in a vector space is referred to as word embedding. Traditional NLP techniques miss syntactic (structure) and semantic (meaning) links across collections of words. As a result, using word vectors to represent words has its advantages. The word vectors are multidimensional continuous floating-point values in which semantically comparable words are transferred to geometrically close regions. Each point in the word vector represents a dimension of the word's meaning, i.e., words used in comparable contexts are mapped to a proximal vector space. Different techniques represent the words in vector spaces such as *Skipgram* skip-G, *Continuous Bag of Words* CBOW, and the co-occurrence frequency. The terms "flower" and "tree," for example, are semantically related since they both refer to plants and are used in the same context [11]. *FastText* is one of these word embedding models that presented word representation for the Arabic language. FastText is an unsupervised learning technique that generates vector representations for words in 294 languages [12]. It supports both CBOW and skip-G models.

Nagoudi et al. [13] used a word embedding model presented by Zahran et al. [14] based on CBOW, Skip-G, and GloVe approaches to determine the semantic similarity of Arabic sentences. Additionally, they included two weighing functions: IDF weighting and POS tagging. They achieved a PCC of 0.7463 ranked the first for applying an approach with the native language and the second among all participants.

Alian et al., [15] proposed a method that combined lexical, semantic and syntactic-semantic features with machine learning techniques like linear regression and support vector machine regression. They applied the Levenshtein method and one of the word embedding models to represent words in a vector space. They evaluated their approach on three different datasets. For the STS-MSRvid of the SemEval competition dataset, they achieved a PCC of 0.743.

The word-based similarity category treats the phrase as a collection of words; hence it is based on the similarity between terms. There are several approaches for determining phrase similarity in this area, including maximum similarity, similarity matrix, employing similar and dissimilar components, and word meaning disambiguation [4]. In addition, several of these strategies are integrated. First, the maximum similarity of each word in the first sentence and each word in the second sentence is determined using the Max similarity approach. The average similarity is then determined [16]. The similarity matrix method generates a matrix containing the results of calculating the similarity between each word in each sentence and the words in the other sentence [17]. Wang et al., [18] describe the use of similar and dissimilar parts to represent words using word embedding. They then used cosine similarity to create a similarity matrix. Furthermore, a semantic matching function was used to create a semantic matching vector for each word. They break down the resulting match vectors to discover which portions of each vector are similar and which are distinct. Finally, the similarity is calculated using these vectors. In [19], they utilized Wordnet synonyms to extend the words of the original phrases, then generated a vector representation for these words in addition to the vectors

of the set of terms in each sentence. Finally, the cosine similarity of the two vectors is used to calculate the similarity.

3. Methodology

In this research, we presented two-hybrid approaches for two different semantic techniques.

3.1 The First Proposed Approach

In the first approach, we present a hybrid methodology that combines two semantic similarity approaches: word-based and vector-based methods, to quantify the semantic similarity between two snipped Arabic texts. The vector space of each word is first retrieved in lemma form using the fastText vector model. The StanfordNLP library was used to generate their lemma form. To analyze natural language, the StanfordNLP package is employed. It turns a string of human language text into lists of sentences and words, generates basic forms of those words, parts of speech, and morphological aspects, and provides a syntactic structural dependency parse that is parallel across over 70 languages [20]. It is utilized for tokenization and lemmatization in both the fastText vector model and the input text in our technique. In the following part, we'll go through how to do that. Second, utilizing the vector word-space of sentences, a word-matching matrix is created. Finally, the degree of similarity between sentences is calculated.

Fig. (1) illustrates the suggested technique, which is divided into three stages:

1) Vector-Based Similarity,
2) Word-Based Similarity,
3) Similarity Measures.

3.1.1 Vector-Based Similarity

FastText Models are available and readable for the Arabic language. *FastText* models trained using CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5, and 10 negatives **[21]**. Arabic fastText corpus contains more than 356 thousand words. These words are totally different represented in their surface form. A screenshot from this dataset is shown in fig. (2).

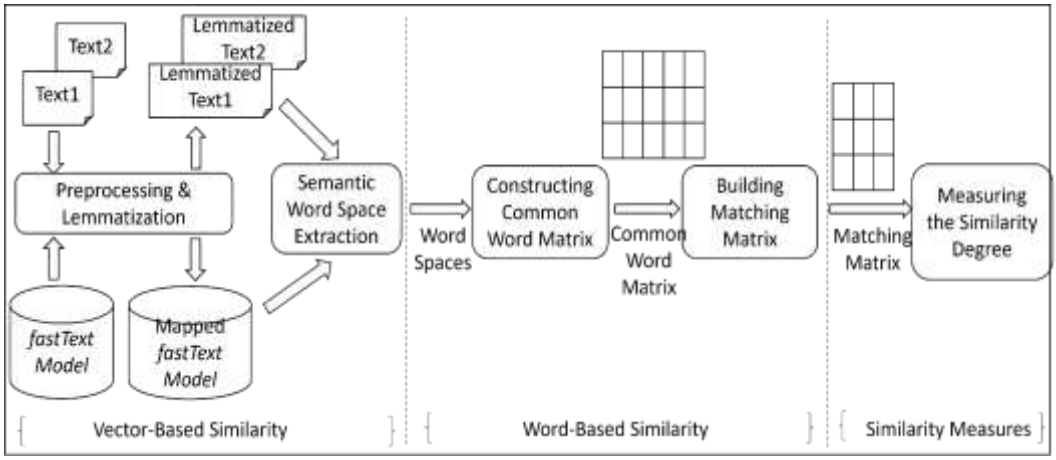

Fig. (1) The Proposed approach Overview.

	0	1	2	3	4	5	6	7
1	0.0127	0.143	0.0158-	0.0564-	0.0678-	0.0535	0.027	من
2	!	0.0394	-0.0668	-0.0247	-0.0421	0.0547	0.1658	-0.0388
3	.	0.0328	0.023	-0.0208	-0.0226	0.0231	0.0412	-0.0272
4	0.005-	0.0259	0.0273	0.0387-	0.0062-	0.0167	0.0728-	في
5	0.0026-	0.0151-	0.0332	0.0126-	0.0108-	0.0162	0.0227-	و
...	...	...	...	...	...	...	...	...
356312	0.0073-	0.0009	0.0069-	0.0086	0.0037-	0.0134-	0.0094-	أبحث
356313	0.0055	0.008-	0.0152-	0.0187	0.0141	0.0143-	0.0133	وحلاوه
356314	0.0017	0.003-	0.0064	0.0012-	0.0083-	0.0021	0.0032-	ودهنت
356315	0.0059	0.0074-	0.0432-	0.0214	0.002-	0.0003	0.003-	شَغّال
356316	0.0078	0.0085	0.0269	0.0055-	0.0074-	0.0048	0.0029	بتطق

Fig. (2) A screenshot from the fastText Arabic Model dataset

Many studies employed the vector space model's surface form to derive the semantic similarity between words that did not include additional semantically related terms. But in this research, a lemmatization technique is used to improve the search word space for the fastText model words. The lemmatization is applied for the input text and the fastText vector model and stated as a mapped fastText model. Some other preprocessing tools are applied for the text such as noise removal, word normalization, eliminating stopwords.

There are two techniques are used to extract the semantic word spaces for each word of the input words: using the closest words algorithm or a ready-made function built in the fastText module in python language.

3.1.1.1 Using the closest words algorithm

To extract the closest words for each word in the sentences, vectors are employed to extract semantic similarity using a word embedding or word representation. For related or near-synonymous words, these vectors shared the same semantic properties. The suggested method extracts comparable words from the mapped fastText model using the preprocessed and lemma form of input words. There are numerous indices in the mapped model for each word in the input text that contains the same word but distinct vectors. Fig. (3) showed the process of extracting the semantic word space for a specific word.

The mapped vector model is used to extract the closest words using Algorithm 1. Where np is the NumPy library stands for Numerical Python.

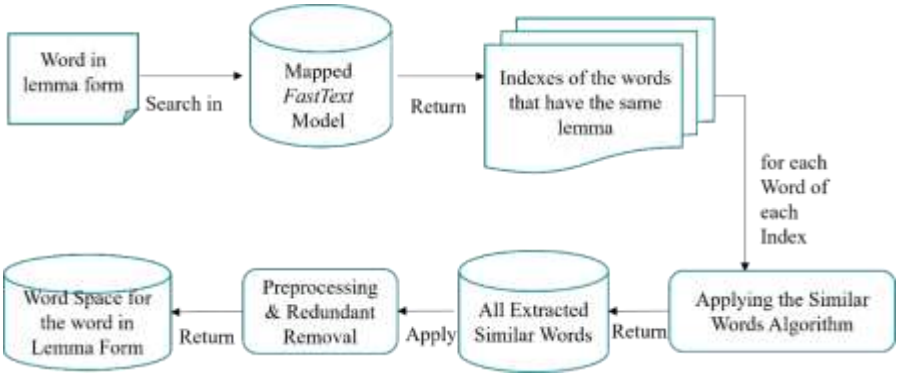

Fig. (3) The process of extracting the semantic word space for a word using the closest words algorithm

Algorithm 1: Finding the closest words for a specific word

Input: The Word Vector, N

Output: Similar Words

1. difference = vector of all words – vector of a word
2. delta = np.sum(difference * difference)
3. i = np.argmin(delta)
4. similar word = word of index i
5. drop the index of this similar word
6. iterate the previous steps by N times
7. Return similar words

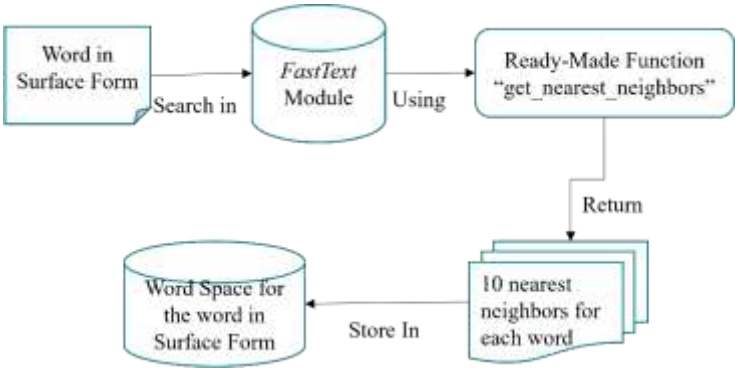

Fig. (4) The process of extracting the semantic word space for a word using the fastText library

Iterating over the whole vectors of the mapped fastText model to obtain the index of the most comparable word can be used to find the closest word. By repeating the loop N times with the same method, N numbers of related terms are discovered. The word space number for each word can be assigned, however, according to the mapped fastText model, each word has a distinct number of indexes. The main purpose is to increase the word search space by collecting more relevant comparable terms for a particular word. In this approach, we extract 10 closest words for each index.

3.1.1.2 Using the fastText module

The language model of the fastText organization was released as a Python module [22]. Similar words can be retrieved using a built-in function named "get nearest neighbors" in this package or module. Like the word space for each word, this function returns the 10 closest neighbors of the searched word in its surface form. The process of extracting the semantic word space using the fastText module is shown in Figure 4.

3.1.2 Word-based Similarity

In this stage, we tried to find the relatedness or the association between words. From the semantic words spaces, a common word matrix is built. This matrix is constructed by the common words of each pair of words by their semantic word spaces. From this matrix, a matching matrix is generated by selecting the most common words between each pair of words. The matching matrix consists of the words of the first sentence matched to some of the second sentence words with the number of common words (NCW).

3.1.3 Similarity Measures

For two sentences s1 and s2 with a length of n and m respectively, the similarity score is measured by Equation 1. Where p is the length of the matching matrix.

$$sim(s1, s2) = \frac{\sum_0^p (NCW/maximum\ semantic\ word\ space)}{max(n|m)} \quad (1)$$

The result value is a ratio from 0 to 1 scale. So, we multiply the output by 5 to make the output ratio is from 0 to 5 scale. The zero value represents that the two sentences are quite different, and the five value indicates that the two sentences are typical.

3.2 The Second Proposed Approach

In the second approach, a modified approach is presented to measure the similarity of two snipped Arabic texts lexically and semantically based on the edit distance approach. This proposed approach is hybrid in the sense that both syntax and semantic features are used to measure the similarity. Different knowledge resources are employed such the semantic word spaces. Also, the approach presents a solution to miss ordering of words between given two sentences. The modified edit distance approach is based on different weights (edit cost) according to the state of the two words.

The proposed workflow for measuring the edit cost between two words is shown in Fig (5).

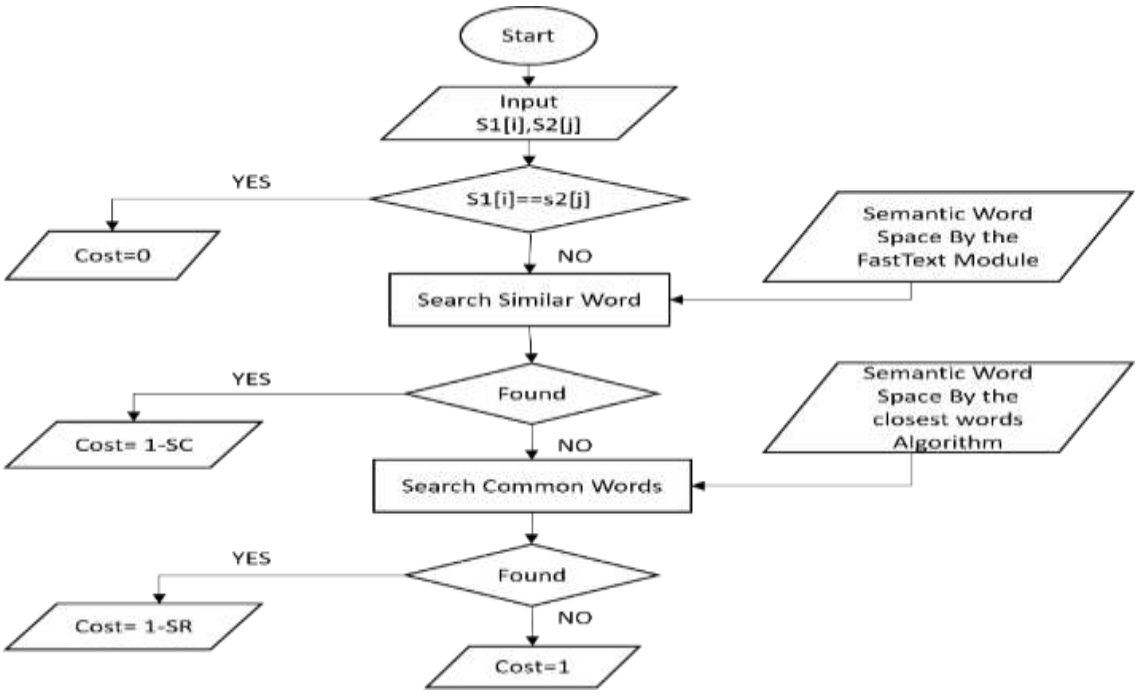

Fig. (5) The workflow diagram of finding the edit cost of two words

Algorithm 2: Finding the edit cost between two words

Input: Pair of words S1[i], S2[j]

Output: Edit cost

Let cost be the edit cost,
IF the word S1[i] = the word S2[j],
then, cost=0;
ELSE IF S1[i] is one of the semantic space words of S2[j] according to the fastText module,
then, cost= 1- similarity score (SC);
ELSE IF S2[j] is one of the semantic space words of S1[i] according to the fastText module,
then, cost = (1- SC);
ELSE IF there are common words bw the two semantic spaces obtained by the closest words algorithm,
then, cost = 1- similarity ratio (SR);
SR = NCW/ Max (|WS1|,|WS2|),
Where NCW is the number of common words between WS1[i] and WS2[j],
Where WS1 is the length of the word space of word i in S1 that differ according to each word,
Where WS2 is the length of the word space of word j in S2 that differ according to each word,
ELSE, cost=1;

The frame algorithm for calculating the edit cost of given two words in two sentences is presented in Algorithm 2.

In this proposed approach, new features are combined to provide accurate measures of the similarity between two snipped Arabic texts. The following summarizes the features and proposed modules of the approach.

3.2.1 Token Lemma Level

The Levenshtein approach is one of the lexical similarity strategies based on the edit distance metrics. The cost of changing one string into another is calculated using its methodology, which assigns a unity cost to all edit operations. This cost is utilized to create a character matrix between the two words, which then yields the edit distance. The suggested method extends the Levenshtein algorithm by computing 'edit distances' at the token level rather than the character level as in the traditional algorithm. The lemma versions of the tokens are represented. In addition, the input text is pre-processed using the mentioned pre-processing tools in the first proposed approach.

3.2.2 Embedding Semantic Knowledge

The cost between each pair of words is calculated using the semantic word spaces produced from the two algorithms outlined in the first proposed approach. In algorithm 2, the edit cost between each pair of words has a different value according to their state. First, the semantic word space obtained from the fastText module is used. The ten nearest neighbors' words obtained from the fastText module have similarity scores (SC) for each term. This score shows how closely the retrieved comparable word is semantically connected to the searched word. The edit cost between two words is determined by (1-SC) if the first word is one of the semantic space words of the second word and vice versa. Otherwise, the semantic word spaces generated from Algorithm 1 are used. The two semantic spaces' common terms are considered as NCW. The SR is then determined by dividing the NCW by the maximum length of the two semantic spaces, which is differing according to each word. The cost will be (1-SR). Otherwise, the cost equals one.

For two sentences S1, S2 with length n, m words respectively, a corresponding matrix CM is constructed depending on the edit cost between each pair of words. The value of CM[i+1][j+1] is as shown in Equation 2:

$$min\begin{cases}\mathrm{CM}[i][j+1]+\ \mathrm{cost},\\ \mathrm{CM}[i+1][j]+\ \mathrm{cost},\\ \mathrm{CM}[i][j]\ +\ \mathrm{cost}\end{cases} \quad (2)$$

Finally, the edit distance between the two sentences is represented by the final value of CM[-1][-1]. The similarity between the two sentences is measured by Equation 3.

$$sim(S1,S2)=1-\frac{Edit\ Distance}{\max(|S1|,|S2|)} \quad (3)$$

3.2.3 Applying Word Permutation

The Levenshtein method is based on word order and word syntactic dependencies. For example, the two phrases "Every morning the sun shines in the sky" and "The sun shines in the sky every morning" in Arabic "تسطع الشمس في السماء كل صباح " and " كل صباح تسطع الشمس في السماء" have the same meaning and their similarity should be 100 percent, however, when using the Levenshtein technique, the similarity would be zero owing to incorrect word order. As a result, a permutation approach is employed to determine the optimal word alignment between the two texts. A permutation is a mathematical approach for determining the number of alternative arrangements in a set when the order of the arrangements is needed. One of the two phrases is rearranged n! times, where n is the sentence's word length. In our approach, we utilized this technique for five words only to reduce the complexity of permutation

operations. For each candidate sequence, the edit distance is calculated. The candidate sequence with the shortest edit distance is chosen as the one that most accurately matches the alignment of the words in the two sentences.

4. Experiments

4.1 Dataset Description

We utilized the dataset from the Semantic Evaluation "SemEval" yearly competition to assess the performance of our approach. This event is used for a variety of languages and track issues. The semantic similarity between texts is one of these tracks (word phrases, sentences, paragraphs, or full documents). Furthermore, the text might be in monolingual or multilingual formats. 2017 was the final year of the competition that used the semantic similarity track.

Development Sets and Evaluation Sets are the two datasets included in the released dataset. One of the Development Sets for monolingual Arabic snipped texts is the STS-MSRvid dataset[1], which contains 368 pairs of sentences. The Evaluation Sets[2] are a collection of 250 sentence pairs with human judgment ratings that were published as the Evaluation Gold Standard[3]. In the output of these datasets, the Pearson Correlation Coefficient (PCC) between each pair of sentences is supplied in a one-column table. This coefficient runs from -1 to 1, with a value of (-1) indicating that the values of the two columns are completely different and a value of (1) indicating that they are identical. This coefficient is expressed in Equation 4.

$$PCC_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \quad (4)$$

Where $\bar{x}$ is the mean of x which is defined by Equation 5.

$$\bar{x} = \sum_{i=1}^{n} x_i/n \quad (5)$$

4.2 Experiments Evaluations

The two proposed approaches are evaluated with two different datasets in two tests as follows:

4.2.1 The First Approach (Evaluation Gold Standard)

The first approach is evaluated using the evaluation gold standard dataset. The experimental results are classified according to the applied algorithm of finding the closest words of the semantic word space of each word.

4.2.1.1 Using the closest words algorithm

In this experiment, two tests are accomplished. The first test uses the mapped vector model with the input text words in their lemma form. The second uses the input text and the fastText vector model in their surface form. First, the input text is preprocessed with the preprocessing tools except the stopwords are eliminated once and keep another. The results of the Pearson correlation coefficient are shown in Table (1).

Results of Table (1) proved that applying the lemmatization technique for the input text with the mapped vector model has a better effect than using the fastText model and the input text in their surface form. In addition, removing the stopwords from the input text improved the results slightly.

Table (1) Experimental Results using the closest words algorithm

Dataset With Pre-Processing	With Surface Form	With Lemma Form
With StopWords	0.6708	0.6886
Without StopWords	0.6887	**0.7000**

Table (2) Experimental Results using the fastText module.

The dataset in Surface Form	Built-In Function	Closest Words Algorithm
With StopWords	0.6513	0.6708
Without StopWords	0.6679	0.6887

Table (3) Experimental Results for studying the negation effect on the proposed Approach.

Dataset With Pre-Processing	Before Studying Negation	After Studying Negation
With StopWords	0.6924	0.7052
Without StopWords	0.7039	**0.7212**

[1] http://alt.qcri.org/semeval2017/task1/data/uploads/ar_sts_data_updated.zip

[2] http://alt.qcri.org/semeval2017/task1/data/uploads/sts2017.eval.v1.1.zip

[3] http://alt.qcri.org/semeval2017/task1/data/uploads/sts2017.gs.zip

4.2.1.2 Using the fastText module

In this experiment, the proposed approach is applied with input text in their surface form to be compatible with the results of the built-In function "get_nearest_neighbors". The results shown in Table (2) are obtained with stopwords and without stopwords.

Table (2) proved that the proposed algorithm for finding the closest words achieved good results than the built-in function that does the same task.

4.2.1.3 Studying Negation Effect

Negation is a significant factor that influences the sentence's orientation (Ismail et al., 2016). Negation terms in Arabic include (ليس، ليست، لن، لا، عدم، لم). The meaning of the sentence is reversed with these negative words. These negation terms are scanned in each sentence of the two input sentences in the proposed method. If a negation word is found in a sentence, the overall score will be reduced by (-1.5), as long as the other sentence does not include negation terms. The negation presence was represented by a score of (-0.5), which substituted the (+1) score from the common word score. The Pearson Correlation Coefficient of the entire dataset was modified by these scores, as seen in Table 3.

In the last experiment done, the Pearson Correlation Coefficient becomes close to the human judgment scores. The score of 0.7212 is the highest value obtained in applying the proposed approach.

4.2.1.4 Comparison with other approaches

Table (4) compares our proposed approach to other works in the AR-AR track of the SemEval competition 2017 that had the highest PCC for the participants.

In Table (4), some researchers used the google machine translator to increase the training dataset as a requirement for the deep learning approach. Therefore, their results are much better than the results of the traditional approaches that use the native language. The proposed approach is considered the second after [13].

4.2.2 The Second Approach (STS-MSRvid)

The second approach is evaluated using the STS-MSRvid dataset. The experimental results were carried out for each methodology and after applying the permutation feature as shown in Table (5).

Table (4) Results for the SemEval participants for Evaluation Sets

Language	Researchers	PCC
Google Machine Translator	**[8]**	0.7543
	[6]	0.7440
	[7]	0.7304
	[13]	0.7463
Native Language	**First Proposed Approach**	**0.7212**
	[10]	0.7158

Table (5) the proposed approach correlation results for the Development Sets

Input With Lemma Form		PCC	PCC + Permutation
Modified Edit Distance (MED)	With StopWords	0.6402	0.7010
	Without StopWords	0.6544	0.7314
MED + Semantic Word Space	With StopWords	0.6792	0.7349
	Without StopWords	0.6835	**0.7589**

Table (6) the proposed approach correlation with similar works for STS-MSRvid Dataset

Used Technique	Research	PCC
Lexical & Semantic Similarity	**Second Proposed Approach**	**0.759**
Similarity features + Machine Learning	**[15]**	0.743
Mean of IDF weighted vectors	**[13]**	0.691

The proposed approach is compared to other works that used the same dataset as shown in Table (6). It demonstrates that our suggested approach has a higher PCC than other research, implying that it has a benefit when compared to other methodologies.

5. Conclusion

In this paper, we proposed two-hybrid approaches that combine different semantic similarity techniques for measuring the similarity between two snipped Arabic texts. The input text is preprocessed with different preprocessing tools such as normalization and removing noise diacritics which improved its efficiency. Moreover, A lemmatization tool is applied for the input text and the used word embedding model to enrich the search word space. The semantic word spaces extracted by applying the closest words algorithm or the fastText module are also lemmatized. Applying the lemmatization technique proved that it has a great effect on results. In addition, the permutation tool is applied to overcome the word order problem that affects the similarity significantly. Finally, the experimental results for the two proposed approaches over two different datasets are satisfying and close to the most values for the researchers who used the same datasets.

References

[1] W.H.Gomaa and A.A.Fahmy, "A Survey of Text Similarity Approaches," International Journal of Computer Applications, vol.68, pp.13-18, 2013.

[2] N.Y.Habash, "Introduction to Arabic natural language processing". Synthesis lectures on human language technologies, vol.3(1), pp.1-187, 2010.

[3] S.Ismail, A.Alsammak and T.Elshishtawy, "A generic approach for extracting aspects and opinions of Arabic reviews," In Proceedings of the 10th international conference on informatics and systems, May., pp.173-179, 2016.

[4] M.Farouk, "Measuring sentences similarity: a survey". arXiv preprint arXiv:1910.03940, 2019.

[5] V.Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," Soviet Physics Doklady, vol.10(8), pp.707-710, 1966.

[6] J.Tian, Z.Zhou, M.Lan and Y.Wu, "Ecnu at semeval-2017 task 1: Leverage kernel-based traditional nlp features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity," In *Proceedings of the 11th international* workshop on semantic evaluation (SemEval-2017), August., pp.191-197, 2017.

[7] J.Henderson, E.Merkhofer, L.Strickhart and G.Zarrella, "MITRE at SemEval-2017 Task 1: Simple semantic similarity," In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), August., pp.185-190, 2017.

[8] H.Wu, H.Y.Huang, P.Jian, Y.Guo and C.Su, "BIT at SemEval-2017 Task 1: Using semantic information space to evaluate semantic textual similarity," In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, August., pp.77-84, 2017.

[9] R.Navigli and S.P.Ponzetto, "BabelNet: Building a very large multilingual semantic network," In Proceedings of the 48th annual meeting of the association for computational linguistics, July., pp.216-225, 2010.

[10] B.Hassan, S.AbdelRahman, R.Bahgat and I.Farag, "FCICU at SemEval-2017 Task 1: Sense-based language independent semantic textual similarity approach," In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), August., pp.125-129, 2017.

[11] Dzone "https://dzone.com/articles/introduction-to-word-vectors", [Date accessed 25/03/2022], 2018.

[12] 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.

[13] E.Nagoudi, J.Ferrero and D.Schwab, "LIM-LIG at SemEval-2017 Task1: Enhancing the semantic similarity for arabic sentences with vectors weighting," In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pp.134-138, 2017.

[14] M.A.Zahran, A.Magooda, A.Y.Mahgoub, H.Raafat, M.Rashwan & A.Atyia**.** "Word representations in vector space and their applications for arabic." In International Conference on Intelligent Text Processing and Computational Linguistics, pp.430-443, Springer, Cham, 2015.

[15] M.Alian, A.Awajan, A.Al-Hasan and R.Akuzhia**,** "Building Arabic Paraphrasing Benchmark based on Transformation Rules," Transactions on Asian and Low-Resource Language Information Processing, vol.20(4), pp.1-17, 2021.

[16] R.Mihalcea, C.Corley, & C.Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai* ,Vol.6(2006), pp.775-780, 2006.

[17] S.Fernando, M.Stevenson, "A semantic similarity approach to paraphrase detection". In Proceedings of the 11th annual research colloquium of the UK special interest group for computational linguistics, pp.45-52, 2008.

[18] Z.Wang, H.Mi, A.Ittycheriah, "Sentence similarity learning by lexical decomposition and composition". arXiv preprint arXiv:1602.07019, 2016.

[19] K.Abdalgader, A.Skabar, "Short-text similarity measurement using word sense disambiguation and synonym expansion". In Australasian joint conference on artificial intelligence pp.435-444, Springer, Berlin, Heidelberg, 2010.

[20] StanfordNLP Package, "StanfordNLP 0.2.0 - Python NLP Library for Many Human Languages", "https://stanfordnlp.github.io/stanfordnlp/index.html", [Date accessed 25/03/2022]

[21] E.Grave, P.Bojanowski, P.Gupta, A.Joulin, and T.Mikolov, "Learning word vectors for 157 languages," arXiv preprint arXiv:1802.06893, 2018.

[22] FastText Model, "Word vectors for 157 languages", "https://fasttext.cc/docs/en/crawl-vectors.html", [Date accessed 25/03/2022].