AtyNegar at NSURL-2019 Task 8: Semantic Question Similarity in Arabic

Atieh Sharifi, Hossein Hassanpoor, Najmeh Zare Maduyieh
Department of NLP, Dade Pardazi Shenakht Mehvar Atynegar (DSA) Institute
Tehran, Iran
sharifiatieh@gmail.com
{hassanpoor, zare}@atynegar.ir

Abstract

Measurement of semantic similarity plays an important role in many areas of natural language processing. Several approaches have been proposed to determine the similarity of sentences in different languages but many of them are not extendable in all languages. According to the complicated Arabic language structure and lack of necessary resources and tools, the Arabic semantic similarity measurement is challenging.

In this paper, we proposed a supervised method for Arabic semantic question similarity measurement. Forty-one features (lexical, syntactic and semantic) are extracted from two question phrases, then the best distinctive features are selected by using SelectKBest algorithm. Finally, for sentences classification and determining the similarity score, SVM used.

The system participated in task8 of NSURL 2019 .The results of using this method on the data set of NSURL 2019 have a F-measure of 82.58 percent, which have improved the basic method.

1 Introduction

Nowadays we encounter a massive amount of text data. Due to the ease of changing a text, similar data are produced abundantly. Measuring text similarity is useful in many cases such as information retrieval, text classification, document clustering, topic detection, question answering, essay scoring, short answer scoring and machine translation. Because of the expansion of text resources and various applications of finding similar texts, the importance of similarity detection can be clearly understood (Gomaa and Fahmy, 2013). As a result, using appropriate methods that can easily recognize similar texts is of great importance.

The most fundamental part in sentences similarity measurement is determining words similarity. Words can be similar both lexically or semantically. Two words are lexically similar if they have a similar character sequence. However semantically similar words used in the same cases, same context or one is a type of another. In this paper several string-based algorithms proposed to determine lexical similarity. Also some corpus-based algorithms proposed to determine semantic similarity (Gomaa and Fahmy, 2013).

String-based algorithms operate on string sequences while corpus-based algorithms determine the similarity between words according to information gained from a large corpus. One approach to measure similarity is using deep learning to represent words and texts as vectors. Similar words have closer vectors and dissimilar words have distant vectors. Therefore, words similarities can be determined by measuring words vector distances. In this paper in addition to words vector representation, we also use sentences vector representation.

In order to increase accuracy, word alignment and syntactic overlapping used to determine similarity. 41 features obtained for two sentences which 38 of them chosen as effective features and used to train the model. The system participated in task8 of NSURL 2019 (Seelawi et al., 2019).

This paper is organized as follows: Section two presents related works in this field, section three introduces the proposed approach and section four representing the results. Finally, section five contains conclusion and suggestions.

2 Related Works

During the last decade, several methods were established to measure sentence similarity based on semantic, syntactic and statistic knowledge. In this section we introduce some related works in determining Arabic sentences and texts similarity.

Wali et al. (Wali et al., 2017) proposed a supervised approach in which three types of features, lexical, semantic and syntactico-semantic, are used to determine sentences similarity. Lexical feature computed based on common terms between the sentences and Jaccard coefficient. In computing semantic features, each sentence represented with a vector and then the cosine similarity of these two vectors are computed. The vectors created by forming a word set using only the distinct terms of the pair of sentences. If the term is in the sentence, the corresponding element in the vector, set to 1 and if the term isn't in the sentence, the corresponding vector element is equal to the highest similarity between the term and the words of the sentence. The similarity of two words calculated using the number of common synonyms of the two words based on LMF standardized dictionaries. The syntactico-semantic features also computed using these dictionaries and common semantic arguments between the pair of sentences. Finally Support Vector Machine used for regression. The F-measure of using this approach on gathered data is 85.6%.

Elghannam (Elghannam, 2016) computing Arabic texts similarity by their words similarity. Each word represented as a vector. This vector is a set of co-occurrence words extracted from a corpus. DISCO tool is used for this purpose. DISCO builds the second order word vectors by first counting words co-occurrences to build the co-occurrence matrix. Cosine similarity of two vectors shows the similarity between two words. The highest accuracy of this method on news data is 97%.

Nagoudi et al. (Nagoudi and Schwab, 2017) represents each word with a vector using word embedding. The vector of each text is the sum of its words vectors. The similarity of two texts computed using cosine similarity between texts vectors. To determine the importance of each word, the word IDF and the part of speech (each part of speech has a score) multiplies the word vector. Best result obtained by using syntactic template and the Pearson correlation is 79.69%.

Al-Smadi et al. (AL-Smadi et al., 2017) proposed a supervised approach to compute text similarity with lexical, semantic and alignment features. These include word overlap, POS tag n-grams overlap, NER overlap, Levenshtein similarity, words alignment and topic modeling (to recognize two texts with a same topic). Finally, a support vector machine used for regression. The results of this approach on news tweets has F-measure of 87.2%.

3 Proposed Method

As mentioned before, the proposed method is supervised approached including preprocessing, feature extraction and classification phases. The preprocessing phase includes removing diacritics, excess spaces, tatweel character and correcting punctuation spacing. In the feature extraction phase, 41 features (lexical, syntactic and semantic) are extracted from two question phrases, then the best distinctive features are selected by using SelectKBest algorithm. Then by classifying the sentences according to the best distinctive features, similarity of the questions is determined.

3.1 Feature Extraction

We consider a total of 41 features. These features explained below.

Words overlap: this type of features computed based on the number of common words in two sentences. These features obtained through the stems vectors of a sentence. In this step we perform tokenizing, then we remove stop words and punctuations. Finally, word stems compared with each other. These features computed for n-grams (n=1,2,3) and precision, recall and F-measure calculated for each of them (AL-Smadi et al., 2017; Karampatsis, 2015).

For n-grams (n=1, 2, 3), precision, recall and F-measure computed as below. If the denominators of the first and second relations are both zero, 1 is considered as the value of all three features. If one of the denominators or the numerator is zero, 0 considered as the value of all three features. These are true for POS tag overlap and NER overlap features too as is stated in the following sections.

$$lexicalP_n = \frac{num\ of\ common\ n{-}grams}{num\ of\ first\ sentence\ n{-}grams} \quad (1)$$

$$lexicalR_n = \frac{num\ of\ common\ n{-}grams}{num\ of\ second\ sentence\ n{-}grams} \quad (2)$$

$$lexicalF_n = \frac{lexicalP_n \times lexicalR_n}{lexicalP_n + lexicalR_n} \quad (3)$$

POS tag overlap: The syntactic similarity of the two sentences is obtained using the number of common syntactic patterns. POS tag vectors of sentences is used for these features. In this step, we removed punctuation marks. After word-tokenizing, their POS tags is specified. Similar to the words overlap feature, the syntactic pattern overlap is computed for 1,2,3-grams, and for each, the accuracy, recall and the F-measure are calculated as follows (AL-Smadi et al., 2017).

$$posP_n = \frac{num\ of\ common\ POS\ tags}{num\ of\ first\ sentence\ POS\ tags} \quad (4)$$

$$posR_n = \frac{num\ of\ common\ POS\ tags}{num\ of\ second\ sentence\ POS\ tags} \quad (5)$$

$$posF_n = \frac{posP_n \times posR_n}{posP_n + posR_n} \quad (6)$$

Named entity overlap: 9 features obtained in this step. These features gathered using named entities vectors. At first we tokenize the sentence. Then the entities are specified. Similarity is also based on the type of named entity (place, person, organization) and the word itself. The number of common named entities is calculated for 1,2,3-grams, for which the precision, recall and F-measure are calculated as follows (AL-Smadi et al., 2017).

$$nerP_n = \frac{num\ of\ common\ NEs}{num\ of\ first\ sentence\ NEs} \quad (7)$$

$$nerR_n = \frac{num\ of\ common\ NEs}{num\ of\ second\ sentence\ NEs} \quad (8)$$

$$nerF_n = \frac{nerP_n \times nerR_n}{nerP_n + nerR_n} \quad (9)$$

Levenshtein maximum similarity: In this step, we calculate the similarity between two sentences based on their words similarity. These features are obtained using stems vectors of the sentences. First, we tokenize and remove punctuation marks. Then the words stems are compared. The Levenshtein method is used to determine the words similarity. A matrix of Levenshtein values is created for two sentences in which rows represent the stems of the first sentence, and the columns represent the stems of the second sentence. Then, by using this matrix, a vector created which contains the lowest values of each matrix row. In fact, we consider the words in the second sentence that are the most similar to the words in the first sentence, and store their Levenshtein values in the vector V. Then we sort this vector and keep only five minimum values. Precision, recall and F-measure are obtained in this step (AL-Smadi et al., 2017). These features are calculated using the sum of vector V values as follows:

$$levP_n = \frac{sum\ of\ V\ values}{num\ of\ first\ sentence\ words} \quad (10)$$

$$levR_n = \frac{sum\ of\ V\ values}{num\ of\ second\ sentence\ words} \quad (11)$$

$$levF_n = \frac{levenshteinP_n \times levenshteinR_n}{levenshteinP_n + levenshteinR_n} \quad (12)$$

Alignment: This group of features computed by the assumption that the two semantically similar sentences can be aligned. To compute these features, we use the V vectors which we obtained in the previous section. If W_i in the first sentence is the most similar word to W_j in the second sentence, $|i-j|$ shows the value of W_i and W_j alignment. For each word in V, the alignment value computed and stored in vector Y (AL-Smadi et al., 2017). Then Precision, recall and F-measure calculated as follows:

$$alignmentP = \frac{sum\ of\ Y\ values}{num\ of\ first\ sentence\ words} \quad (13)$$

$$alignmentR = \frac{sum\ of\ Y\ values}{num\ of\ second\ sentence\ words} \quad (14)$$

$$alignmentF = \frac{alignmentP \times alignmentR}{alignmentP + alignmentR} \quad (15)$$

Character sequence: There are 4 features in this step, each of them divided to the minimum length of the two sentences. First we tokenize the sentences, then remove stop words and punctuation marks, after that we extract words stems and rebuild the sentences (Tian et al., 2017).

- LCPrefix: The largest common prefix of two sentences
- LCSuffix: The largest common suffix of two sentences
- LCSubString: The largest common substring of the two sentences
- LCSequence: The largest common sequence of two sentences. Here we consider the common characters in the two sentences.

BOW similarity: In this step, a vector is considered for each sentence. The cosine similarity of these two vectors is considered as a feature for the two sentences. Initially, we tokenize the sentences. Then the punctuation marks and stop words are removed from the tokens. The vectors of the two sentences have a same size which is equal to the size of the common unique words in these sentences. If the word in the vector exists in the sentence, the IDF of that word will replace it, otherwise its value will be zero (Tian et al., 2017). The IDF values are created using the Arabic Wikipedia corpus.

Word embedding similarity: A simple definition for word embedding is to consider a vector of numbers for each word. The words that are more similar to each other, have closer vector space. This vector specifies the syntax, semantic and other features of the word. This way, it is possible to display each word in tens or hundreds of dimensions. There are several algorithms for this purpose, while FastText is used here (Grave et al., 2018). Arabic Wikipedia used to build this model. Using the FastText pertained model, you can get a 300-dimensional vector for each word. First, tokenization performed, then the punctuation marks removed. Each sentence is represented as a vector. This vector is obtained from the sum of the vectors of the sentence words and finally their average. Then a feature is obtained using the cosine similarity of these two vectors (Eyecioglu and Keller, 2016).

Word mover's distance (WMD) score: In this feature, the distance between two sentences is obtained based on the words vectors distance using FastText model. The more similar the words of two sentences, the less distance between sentences vectors. Thus for same sentences this value tends to zero. First, the sentences are tokenized, then the punctuation marks and stop words removed. Finally, the distance between two sentences is calculated.

Doc2vec similarity: in the FastText model, each word represented by a vector. Unlike FastText, the doc2vec model gives us a numerical representation of a document, and we use it here to construct a vector for each sentence. Arabic Wikipedia has been used to construct this model. First, we tokenize the sentence and remove the punctuation marks. By using the pre-trained model, a doc2vec vector created for each sentence. Then a feature is obtained using the cosine similarity of these vectors.

3.2 Sentence Classification

We use support vector machine to classify the sentences. After feature extraction and creating the training data, preprocessing these data should be done. As the first step we normalize the data. In the process of normalization, the values of each feature (each column) is mapped to zero-mean and unit variance values. Finally, the best features are selected. For this purpose, SelectKBest algorithm is used. This algorithm gets the number of selected features n as the input. Then the model is built using the selected features.

4 Evaluation

In order to select appropriate features, we examined different inputs for SelectKBest algorithm. All the training data of NSURL task8, was used for test and the training model is built using SVC. Due to Table 1, SelestKBest(38) algorithm has the best results for detecting similar sentences. Table 2 shows the score and the effect of each feature using this algorithm. These features are sorted by their score. According to this table, precision, recall and F-measure of NER 3-grams overlap are the three deleted features that have the least score. Because of the short sentences, there is no NER overlap at the 3-gram level in the sentences, so this feature is not effective.

The SVC model is built, using the 38 features mentioned before. The model is evaluated using the NSURL 2019 task8 test data. The F-measure of the proposed method is 82.58%.

Algorithm	**F-score**	**Recall**	**precision**
SelectKBest(37)	0.798	0.767	0.832
SelectKBest(38)	0.808	0.783	0.834
SelectKBest(39)	0.807	0.782	0.834
SelectKBest(41)	0.806	0.782	0.831

Table 1: Results of testing SelectKBest algorithms on svc model.

5 Conclusion and Suggestions

In this paper, in order to detect similar Arabic questions, a supervised approach proposed. 38 effective feature s are extracted for each pair of questions which contain syntactic, semantic and lexical features. Semantic features are obtained using word embedding and doc2vec. Lexical features are obtained by words overlap feature. String based algorithms and syntactic features are also obtained using the syntactic structure of the sentence.

Due to Table 2, words overlap is one of the most effective features. After that, the largest common suffix, Word mover's distance, Levenshtein similarity, doc2vec similarity and bag of words similarity have the highest priority respectively.

Feature	Score	Feature	Score	Feature	Score	Feature	Score	Feature	Score
lexicalP1	4095	levF	1723	alignmentR	879	nerR1	208	nerP2	19.7
lexicalF1	3055	Doc2vec	1564	alignmentF	646	nerP1	206	LCPrefix	6
LCSuffix	2871	levP	1488	posP1	633	Word_Embedding	204	nerP3	2
lexicalP2	2624	BOW	1276	posP2	380	posF3	146	nerR3	2
lexicalP3	2624	LCSequence	1239	posF1	326	posR2	60	nerF3	2
lexicalF2	2179	lexicalR2	1231	alignmentP	314	posR3	50		
lexicalF3	2179	lexicalR3	1231	posP3	229	posR1	33		
WMD	1885	LCSubString	998	posF2	224	nerR2	20		
levR	1822	lexicalR1	910	nerF1	210	nerF2	19.9		

Table 2: Features ranking using selectkbest algorithm.

Lexical features are effective because there is high word overlap between similar sentences. Also there are some synonyms in some of similar sentences, so Word mover's distance feature can be helpful in detecting such sentences. But sometimes instead of two words, two phrases can be equivalent in meaning. Such cases are harder to detect. In Doc2Vec feature the whole sentence represents as a vector so it can covers some of the flaws. In some pairs of questions, the meaning has changed with displacement of the words, although in many cases this does not change the meaning, so the alignment feature can be somewhat effective in identifying similar sentences.

After examining the sentences which were incorrectly identified as similar, we found that removing stop words improved the accuracy of the system, although in some cases deleting these words has led to a mistaken identification. For example, some question words like who or when are effective in similarity detection but some of these words are ignored by removing stop words. Also in some cases there is excess information in one of the sentences which doesn't change the meaning but leads to incorrect similarity detection.

In order to improve the results, we can also consider the similarity of the question words in the two sentences. For example, the question word "which year" is equivalent to "when". Also the synonym words in two sentences can be determined using the semantic networks. Then in computing words overlap we can assume that these words are equal. To identify the similarity between words, instead of Levenshtein similarity, semantic networks can be used (Pawar and Mago, 2018). Syntactic n-grams overlap using the sentence dependency tree is another feature that can be effective in determining the similarity of two sentences (Segura-Olivares et al., 2013; Kohail et al., 2017)

References

Mohammad AL-Smadi, Zain Jaradat, Mahmoud AL-Ayy, and Yaser Jararweh. 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. https://doi.org/10.1016/j.ipm.2017.01.002.

Fatma Elghannam. 2016. Automatic Measurement of Semantic Similarity among Arabic Short Texts. *Communications on Applied Electronics (CAE), 6*(2):16-21. https://doi.org/10.5120/cae2016652430.

Asli Eyecioglu and Bill Keller. 2016. ASOBEK at SemEval-2016 Task 1: Sentence Representation with Character N-gram Embeddings for Semantic Textual Similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 736-740.

Wael Gomaa and Aly Fahmy. 2013. A survey of text similarity approaches. *International Journal of Computer Applications, 68*(13):13-18. https://doi.org/10.5120/11638-7118.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning Word Vectors for 157 Languages. In Proceedings of the International Conference on Language Resources and Evaluation. Pages 3483-3487.

Rafael Michael Karampatsis. 2015. CDTDS: Predicting Paraphrases in Twitter via Support Vector Regression. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 75–79.

Sarah Kohail, Amr Rekaby Salama, and Chris Biemann. 2017. STS-UHH at SemEval-2017 Task 1: Scoring Semantic Textual Similarity Using Supervised and Unsupervised Ensemble. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017).* Pages 175-179.

El Moatez Billah Nagoudi and Didier Schwab. 2017. Semantic Similarity of Arabic Sentences withWord

Embeddings. In *proceedings of the Third Arabic Natural Language Processing Workshop (WANLP)*, pages 18-24.

Atish Pawar and Vijay Mago. 2018. Calculating the similarity between words and sentences using a lexical database and corpus statistics. *arXiv:1802.05667*.

Haitham Seelawi, Ahmad Mustafa, Hesham Al-Bataineh, Wael Farhan, and Hussein T.Al-Natsheh. 2019. {NSURL}-2019 Task 8: Semantic Question Similarity in Arabic. In *Proceedings of the first International Workshop on NLP Solutions for Under Resourced Languages.* Trento, Italy.

Andrea Segura-Olivares, Alejandro Garc´ıa, and Hiram Calvo. 2013. Feature Analysis for Paraphrase Recognition and Textual Entailment. *Research in Computing Science, 70*:144-119.

Junfeng Tian, Zhiheng Zhou, Man Lan, and Yuanbin Wu. 2017. 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 Evaluations (SemEval-2017)*, pages 191–197.

Wafa Wali, Bilel Gargouri, and Abdelmajid Ben Hamadou. 2017. Enhancing the sentence similarity measure by semantic and syntactico-semantic knowledge. *Vietnam Journal of Computer Science, 4*(1):51-60. https://doi.org/10.1007/s40595-016-0080-2.