Journal of Universal Computer Science, vol. 26, no. 6 (2020), 671-697
submitted: 26/10/19, accepted: 9/3/20, appeared: 28/6/20

Question to Question Similarity Analysis Using Morphological, Syntactic, Semantic, and Lexical Features

Mahmoud M. Hammad, Mohammad AL-Smadi, Qanita Bani Baker
Muntaha Al-asa'd, Nour Al-khdour, Mutaz Bni Younes
Enas Khwaileh
(College of Information Technology
Jordan University of Science and Technology
Irbid, Jordan, 22110
m-hammad@just.edu.jo, maalsmadi9@just.edu.jo, qmbanibaker@just.edu.jo
mabalasad@gmail.com, naalkhdour17@cit.just.edu.jo
mmbniyounes18@cit.just.edu.jo, ekhwaileh18@cit.just.edu.jo)

Abstract In the digitally connected world that we are living in, people expect to get answers to their questions spontaneously. This expectation increased the burden on Question/Answer platforms such as Stack Overflow and many others. A promising solution to this problem is to detect if a question being asked is similar to a question in the database, then present the answer of the detected question to the user. To address this challenge, we propose a novel Natural Language Processing (NLP) approach that detects if two Arabic questions are similar or not using their extracted morphological, syntactic, semantic, lexical, overlapping, and semantic lexical features. Our approach involves several phases including Arabic text processing, novel feature extraction, and text classifications. Moreover, we conducted a comparison between seven different machine learning classifiers. The included classifiers are: Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression (LR), Extreme Gradient Boosting (XGB), Random Forests (RF), Adaptive Boosting (AdaBoost), and Multilayer Perceptron (MLP). To conduct our experiments, we used a real-world questions dataset consisting of around 19,136 questions (9,568 pairs of questions) in which our approach achieved 82.93% accuracy using our XGB model on the best features selected by the Random Forest feature selection technique. This high accuracy of our model shows the ability of our approach to correctly detect similar Arabic questions and hence increases user satisfactions.

Key Words: Arabic Language, NLP, Semantic Text Similarity (STS), Machine Learning, Text Classification, Lexical Features, XGB, LDA.

Categories: I.2.7, H.3.3

1 Introduction

Nowadays we are witnessing an increase in the number and popularity of Question/Answer platforms such as Stack Overflow, Quora.com, or Mawdoo3.com. These web platforms relying on asking questions by users and providing them with answers in a timely manner. One important factor to increase "time to respond" to users and hence increasing the reputation of these websites is to detect if a newly asked question has been asked before. If a similar question exists, then the answer of that question will be provided to the user. Detecting similar questions is a research and industrial problem that has not been solved completely especially for the Arabic language.

Accurately detecting similar questions would increase the reputation of such platforms, increase the satisfaction and the loyalty of their users, avoid duplicate answers, and eliminate the extra efforts needed to answer the new question. The main challenge to solve this problem is how to efficiently and accurately detect if two questions are similar.

Semantic Text Similarity (STS) techniques [Islam and Inkpen, 2008] [Cer et al., 2017] are concerned with recognizing the similarity of two texts. STS has been widely utilized in the Natural Language Processing (NLP) research such as text classification, text summarization, information retrieval, and word sense disambiguation. In this paper, we extend our previous work in [Al-asa'd et al., 2019] and propose a novel approach to detect the similarity between two Arabic questions based on their morphological, syntactic, semantic, and lexical features. Our approach involves several steps including Arabic text processing, novel feature extraction, and text classifications.

Text classification tasks are considered as supervised learning problems [Kotsiantis et al., 2007][Dietterich, 2000] where the model is trained using labeled data. Some machine learning algorithms have been used to solve the text classification problems such as Support Vector Machine (SVM) as in [Wang et al., 2006], [Joachims, 1999], and [Chen et al., 2009] and Decision Tree (DT) as in [Yeh et al., 2019]. To solve this problem, we extracted novel features from a large labeled dataset obtained from a real-world question/answer platform and trained seven different models using well known machine learning algorithms.

This paper presents several new non-trivial extensions to the preliminary version of our work described in [Al-asa'd et al., 2019]:

1. In addition to the XGB model that we designed in [Al-asa'd et al., 2019], in this manuscript, we designed and trained seven different machine learning models. We reported the performance of all of these models and we compared the performance of these models using various sets of features. The seven machine learning models we designed and trained in this paper are: Support Vector Machine (SVM), Decision Tree (DT) [Yeh et al., 2019], Logistic Regression (LR) [Tang and Li, 2019], Extreme Gradient Boosting (XGB) [Ali and Rafi, 2018], Random forests (RF) [Wang et al., 2018], Adaptive Boosting (AdaBoost) [Bhopale et al., 2018], and Multilayer Perception (MLP) [Lithgow-Serrano and Collado-Vides, 2019].

2. We enlarged the dataste used for training and testing. In [Al-asa'd et al., 2019], we trained our models on 4,000 pairs of ques-

tions whereas in this paper, we trained our models on 9,568 pairs of questions, i.e., 19,136 distinct questions.

3. To better represent Arabic questions and enhance the models, in this research work, we have extracted more features than the ones described in [Al-asa'd et al., 2019]. In addition to the morphological, syntactic, semantic, and lexical features extracted in [Al-asa'd et al., 2019], we also extracted overlapping and semantic lexical features such as complexity features, Latent Dirichlet Allocation (LDA) features, Term Frequency–Inverse Document Frequency (TF-IDF).

4. On top of the previous technical contributions, we conducted a literature review and discussed a large number of related research efforts in the area of Arabic text similarity detection methods. Moreover, the paper provides detailed description of the trained machine learning models and the utilized parameters to give the best results.

To summarize, the contribution of this paper is twofold. First, we extract novel features of Arabic questions including the morphological, syntactic, semantic, and lexical features. Second, we build seven different machine learning models and we compared between them. We used a real-world Arabic questions corpus that consists of 19,136 Arabic questions (9,568 pairs of Arabic questions) shared with us from the Mawdoo3 company (mawdoo3.com), the largest Arabic website with more than 50 million visitors monthly and it hosts more than 150,000 articles. To the best of our knowledge, this is the first research to apply text classification to detect similar Arabic questions using morphological, syntactic, semantic, lexical, overlapping, and semantic lexical features using various supervised machine learning models applied on a real-world dataset.

The remainder of this paper is structured as follows. Section 2 covers the related work. Section 3 describes the various machine learning algorithms that we have utilized and compared in this paper. Section 4 introduces our methodology of preparing the dataset and designing seven different machine learning models to solve the question to question similarity problem. Section 5.1 describes the different evaluation metrics we used to compare between the various machine learning classifiers. Section 5 covers our evaluation design and results. The last sections discusses important findings in our research (Section 6) and conclude the paper with avenue of future work.

2 Related Work

In this section, we review some previous work related to the Arabic text similarity classification task presented in a chronological order.

[Siolas and d'Alché Buc, 2000] proposed an approach to solve the text classification problem based on a priori semantic knowledge of words. They used two supervised classification algorithms, the Support Vector Machine (SVM) and the K-Nearest Neighbors (K-NN), and they found that SVM outperformed the K-NN.

[Al-Shalabi et al., 2006] evaluated the K-Nearest Neighbor (KNN) machine learning model to solve the Arabic text categorization problem. The keywords in the 621 studied documents selected based on Document Frequency (DF) threshold, then each document tokenized and the stop words removed. Next, the frequency of each token calculated and ranked. The category of a text is classified by matching the highest-ranking tokens with the list of keywords. Their satisfying results showed that KNN model an be used for Arabic text categorization. Similarly, [Duwairi, 2006] proposed a distance-based classifier for categorizing Arabic text. Each category is represented as a feature vector. Then a given document is classified based on the closeness of the document's feature vector to a feature vector of a category.

[Khreisat, 2006] applied an n-gram frequency statistics method for Arabic text classification. They employed the Dice's similarity measure and Manhattan distance as a dissimilarity measure. The chosen dataset was a corpus of Arabic text documents. These documents were fetched from online Arabic newspapers. The dataset falls under 4 categories; Sports, Economy, Technology, and Weather. An hold-out evaluation method was used to split the data. Only 40% of the corpus was used for the training phase while the other 60% split was used for testing. They found that classifying text documents from all the mentioned categories using the Dice measure outperforms the Manhattan distance measure with respect to recall and precision as evaluation metrics.

[Gharib et al., 2009] applied a Support Vector Machine (SVM) model to classify Arabic text documents. They also compared the results of the SVM model with Naive Bayes classifier, KNN classifier, and Rocchio classifier. Experiments were conducted on a set of 1132 documents and the results showed that Rocchio model achieved better results when the feature set is small while the SVM model gave better results when the size of the feature set is large.

[Zouaghi et al., 2011] evaluated the Lesk algorithm variants to remove text ambiguity using Arabic Wordnet (AWN). Using the Al-Wasit Dictionary and texts obtained from newspaper articles in various fields such as sport, politics, religion, science, etc. Their work was limited to a sample of 50 vague Arabic words. They performed two experiments: first, using the original Lesk algorithm with the dictionary as a resource. Second, they made some modifications to the Lesk algorithm by adding different similarity measures to determine the similarity of the relationship between two concepts in Arabic Wordnet (AWN). The results showed that the modified algorithm achieved 67% precision using a size of 2-word window.

[Ezzeldin and Shaheen, 2012] surveyed the various research efforts regarding Arabic Question Answering tasks including question understanding, information retrieval, and answer extraction. For each task, they enumerated the research contributions, the faced challenges in solving each task, and the various tools used for Arabic language processing. At the end of the survey, they concluded that there are many rooms for improvement regarding each Arabic Question Answering task.

[Froud et al., 2012] compared between root-based and stem-based techniques for measuring the similarity of Arabic texts using the Latent Semantic Analysis (LSA) contextual-representation method [Landauer and Dumais, 1997]. They used various distance functions and similarity measures such as Cosine Similarity,, Euclidean Distance, Pearson Correlation Coefficient, and the Jaccard Coefficient. This comparison showed that the stem-based is better than the root-based since the root-based depends on the three-letter rood of Arabic words and hence it affects the meaning of words in their context. Moreover, they found that the Jaccard similarity measure performed the worst comparing to all other similarity measures.

[Ababneh et al., 2014] evaluated different Text classification techniques. They implemented the K-NN algorithm on three variations of the Vector Space Models (VSMs); Cosine coefficient, Dice coefficient, and Jaccard coefficient. The Saudi Newspapers (SNP) dataset was used for the experiments. The dataset contains 5,121 Arabic documents belongs to 7 different predefined categories; Culture, Economics, General, Information Technology, Social, Sport, and Politics. In order to compare between the three measures, they computed the average over the 7 categories, then they found that the Cosine coefficient outperformed the other two measures (Dice and Jaccard).

The Qatar Computing Research Institute (QCRI) developed a system as part of the SemEval 2015 - Task 3 challenge [Nicosia et al., 2015]. The developed system by Nicosia et al. [Nicosia et al., 2015] selects the best answer for community question platforms in Arabic and English languages. They trained the model on various features including N-gram, heuristics, text similarity, specific words, and emotion analysis. They used three similarity measures to calculate the similarity between questions: lexical similarity, semantic similarity, and syntactic similarity. Their model achieved the best performance regarding the Arabic questions and the third place regarding the English questions.

[Alhutaish and Omar, 2015] addressed the problem of Arabic text classification into multi-categories problem. They conducted several experiments using the K-NN classifier with four similarity measures Inew, Jaccard, Dice, and Cosine similarity. The dataset used for evaluation consists of 3,172 Arabic documents which has four categories (Art, economic, politics and sport). The TREC-2002

Light Stemmer method used to stem data without prefixes and suffixes. Both stemmed and un-stemmed data were used in the experiments, the dataset represented using Bag-Of-Words (BoW) and character-level 3 (3-Gram). The results showed that K-NN with Inew achieved the best performance and takes less time.

[Al-Anzi and AbuZeina, 2017] showed that the cosine similarity is a preferable measure for Arabic text classification. They also provided a comparison between eight text classification techniques. This research supports our decision of including the cosine lexical feature in our model.

[Mohammad et al., 2017] proposed an approach for detecting similar news in Arabic Tweets of the Twitter social media. Despite their ability to detect similar Tweets, they only used two models, Support Vector Regression (SVR) and Maximum Entropy (MaxEnt) models whereas we used seven different models.

[Romeo et al., 2017] proposed a deep learning model for ranking Arabic questions. Once a user asks a new question, the proposed approach retrieve similar questions and ranks them to the user. So the user can get the answer faster from the similar questions. To extract features, such as lexical and syntactic information from Arabic text, they combined a community Question/Answer architecture into a UIMA-based framework. Their results showed that syntactic information is crucial for the task of ordering questions. Their model combines tree beads, floor decorations, and neural networks to select text. Mahmoud et al. [Mahmoud and Zrigui, 2017] proposed a paraphrase identification method to detect plagiarism in Arabic texts. On their approach, they detect the semantic similarity between Arabic sentences by using Natural Languages Processing (NLP) techniques such as Term Frequency - Inverse Document Frequency (TF-IDF). They trained their model using the Open Source Arabic Corpora (OSAC) dataset, mainly the historical documents which contains 3,233 documents. In addition, they used the word2vec algorithm to reduce arithmetic complexity and to improve the probability of word prediction in text and the distribution of word vector representations, after calculating the similarity measurement based on different comparison measurements such as Cosine similarity and Euclidean distance.

Similarly, [Zaher et al., 2017] and [Ghanem et al., 2018] developed approaches to detect plagiarism in Arabic text. [Zaher et al., 2017] developed a web tool to detect the plagiarism in Arabic documents called Arabic Plagiarism Detection System (APDS). APDS 82% recall on a small set of documents, 10 solutions of a course assignment submitted by students in the Sattam bin Abdul-Aziz University. On the other hand, [Ghanem et al., 2018] proposed a Hybrid Arabic Plagiarism Detection System (HYPLAG) to detect plagiarism. The ExAraPlagDet dataset was used to evaluate the HYPLAG system for detecting plagiarism in Arabic texts. They conducted a study in which a sample of university students participated in that study. The idea of the study is to understand the students' plagiarism pattern and the use of sentences stolen during writing research homeworks, as one of the issues of literacy theft. The results of the pilot system showed that RDI [Magooda et al., 2015], the best performing system in PAN@Fire-2015, performed the best with 89% F-measure.

[Hassan et al., 2019] proposed (1) a new synset-oriented word aligner based on the BableNet semantic network [Babelscape, 2019]; (2) they proposed three new unsupervised learning models to solve the semantic text similarity (STS): string kernel-based (SK), alignment-based (AL), and weighted alignment-based (WAL); and (3) they proposed an unsupervised ensemble STS approach called UESTS which utilizes four different similarity measures. They evaluated their approach on dataset provided by the SemEval workshop, an annual workshop for semantic text similarity problem.

[Hamza et al., 2019] built a taxonomy of Arabic question domains and they also proposed a technique for classifying Arabic questions to help question answering platforms to retrieve answers more efficiently.

[Bekkali and Lachkar, 2019] proposed a functional representation for short Arabic text conceptualization, such as tweets and text messages, based on concepts instead of terms using "BabelNet" as an external knowledge. By better representing short Arabic text using their proposed approach, the performance of the short Arabic text classification approaches would improve. Solving such a problem is challenging especially for Arabic language since a word could have multiple meanings and short messages tend to be ambiguous. To overcome that, they relied on Bag-of-Concepts (BoC) instead of Bag-of-Words (BoW) for representing short texts.

All of the aforementioned research efforts are related to our semantic text similarity research. However, they all do not solve the problem that we are trying to solve, that is, given two Arabic questions, can we determine if they are similar efficiently and accurately. Moreover, none of them compared between various machine learning models using different sets of features to determine the best set of features and the best model for Arabic question to question similarity problem.

3 Machine Learning Algorithms

Machine learning algorithms can be used either for supervised learning or unsupervised learning. Supervised learning is a data mining technique to infer a function from a labeled dataset. On the other hand, unsupervised learning techniques try to group unlabeled data into several groups, also called clusters, in which members of the same cluster are more similar than members of different clusters.

Since the dataset we have is a labeled dataset, i.e., each question is labeled with either "Yes" or "No", we used several supervised learning techniques to build various models that can determine if two previously unseen Arabic questions are similar or not. To help the reader understands the rest of the paper, this section describes the utilized supervised machine learning algorithms in more details.

3.1 Decision Tree (DT)

Decision Tree (DT) classifier is a supervised machine learning technique that is widely used due to its ability to break down complex decision making process and the ability to interpret its decision. The classification process is done through various rules (conditions) to make decisions in a tree structure. It arranges and splits the rules to build a decision tree based on the best features. Since the decision tree classifier is widely used as a based line to compare it with other classifiers, we used it in our approach.

The decision tree divides a dataset into smaller subdivisions. Each subdivisions contains a set of nodes. The final image of the dataset forms a tree structure where each node forms a condition. In particular, the decision tree does require assumptions for the distribution of the input data. It deals with nonlinear relationships between attributes and categories and it deals with digital and class inputs in a natural way. In addition, its structure makes it easy to explain the taken decision by the model. In our methodology, we trained a DT model with the following parameters: max_depth: 5, max_features: 'log2', min_samples_leaf: 3, and random_state: 25.

3.2 Logistic Regression (LR)

Logistic Regression (LR) classifier is a machine learning technique that is commonly used when having a binary classification task.

There are three types of LR models: (1) Binary model when there are only two categories to be learned such as in the email spam detection problem in which the model needs to mark the email as either span or not. (2) Multinomial Logistic Regression where the categories to be learn are more than two categories such as in the animal detection problem where the model needs to detect the type of an animal which could be cat, dog, Sheep, etc. (3) Ordinal Logistic Regression where there are three categories with ordering to be learned such as low, medium, and high categories.

The output of the logistic regression model is a binary value (either 0 or 1). Equation 1 shows the main equation used in logistic regression to make a prediction. Where y is the predicted output, b0 is the bias and b1 is the coefficient for the single input value (x). Each column in the input data has an associated b coefficient (a constant real value) that must be learned from the training data.

$$y = \frac{e^{(b0+b1*x)}}{1+e^{(b0+b1*x)}} \tag{1}$$

We trained our LR model with the following parameters:penalty='l1', C=0.1, multi_class='multinomial, and random_state=None.

3.3 Random Forest (RF)

Random Forest (RF) is an ensemble supervised learning technique for classification and regression. Ensemble learning algorithms combine two or more learning algorithms to achieve better results than the results that could be achieved individually by the constituent algorithms.

Random forest construct several decision tree classifiers on various subsamples of the dataset to overcome the overfitting problem of the decision tree. Overfitting means that the classifier generates a model that fits the training data but cannot be generalized to unseen samples. We trained our RF model with the following parameters: max_depth:9, criterion:'entropy', n_estimators=10, and max_leaf_nodes:None.

3.4 Multilayer Perceptron (MP)

Multilayer Perceptron (MLP) is a type of feedforward artificial neural network. It has at least three layers: an input layer, a hidden layer, and an output layer. Each node in all layers except the input layer has an activation function. An activation function of a node is a function that calculates the output of a node given an input or a set of inputs to that node such as a sigmoid function.

Multilayer perceptron model is a supervised backpropagation learning technique. Backpropagation is an efficient and iterative technique to calculate and update the weights of the neurons. It calculates the error between the output and the predicted output and then adjusts the weights based on the error.

We trained MLP model with the following parameters: hidden_layer_sizes=100, activation="relu", solver="adam", alpha=0.0001, batch_size="auto", learning_rate="constant", and random_state=None.

3.5 Adaptive Boosting (AdaBoost)

Adaptive Boosting (AdaBoost) is another ensemble supervised learning technique. AdaBoost is the first successful boosting algorithm. It aims to convert a set of weak classifiers into a strong one. It is calculated through **Equation 2**.

We trained AdaBoost model with the following parameters: n_estimators:50, random_state:None, and learning_rate:1.0.

$$H(x) = sign\left(\sum_{t=1}^{T} a_t h_t(x)\right) \tag{2}$$

Where $\boldsymbol{T}$ represents all weak classifiers, $\boldsymbol{t}$ represents a single weak classifier, $\boldsymbol{Alpha_t}$ is the weight of classifier $\boldsymbol{t}$ and $\boldsymbol{h_t(x)}$ is the output of the weak classifier $\boldsymbol{t}$.

3.6 Extreme Gradient Boosting (XGB)

Extreme Gradient Boosting (XGB) is an implementation of gradient boosted decision trees used in supervised learning tasks such as regression and classification tasks [Hamza et al., 2019]. XGB is one of the CART family. It produces a model of ensemble weak prediction models, mainly decision trees. Each time a weak leaner is added, the loss value is computed until the model achieves a satisfactory performance value. XGB is one of the CART family. It produces a model of ensemble weak prediction models, mainly decision trees. The classification is done through some iteration based on the ensemble tree-boosting method.

Each time a weak learner is added, the loss value is computed and the weights are adjusted on each iteration to decrease the loos and increase the performance of the model. At the end of this iterative process, all the modeled trees are summed up to form the final classification with the best performance. Moreover, XGB has a better control to overfitting than other boosting algorithm. XGB is known for having the best performance in many domains and various datasets due to its speed, performance, accepting various input types, e.g., dense matrix, sparse matrix, and data file.

We trained our XGB model with the following parameters: booster= 'gbtree', learning_rate:0.3, num_feature:'auto', subsample=1, max_depth=6, min_child_weight=1, and colsample_bytree =1 .

3.7 Linear Support Vector Machine (Linear SVM)

Linear Support Vector Machine (SVM) is a traditional machine learning algorithm for supervised learning tasks, classifications and regressions. It creates a hyperplane that separates the training data into classes. To ensure the effectiveness of classifying new data, a plane with a maximized margin is chosen, where the margin is the distance between the data related to classes. SVM implemented using different kernels such as Linear Kernel, Polynomial Kernel, and Radial Kernel.

Linear SVM uses the dot product as the similarity measure between support vector in training data and new data because the linear SVM focuses on inner product of two inputs rather than the inputs themselves, where the inner product calculate the sum of multiplication of each pair between two vectors. SVM offers high accuracy in the classification task.

Our SVM model was done using linear kernel. We trained the model with the following parameters: kernel: 'linear', C: 1.0, loss='squapink_hinge', random_state = None, and penalty='l2'.

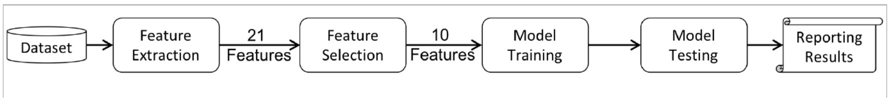

Figure 1: Model Architecture

4 Methodology

Question to Question (Q2Q) similarity detection task can be achieved mainly in two ways: string-based matching technique and machine-learning-based technique. For string-based technique, an approach needs to check if two questions contain same words or similar words, in terms of their meanings. Such an approach does not work in real-life as it is tedious, since one needs to build word-to-word similarity map, and it cannot be generalized to new questions with new words.

On the other hand, learning-based technique utilizes machine learning to automatically classify if two questions are similar or not. Such an approach is efficient and generalizable. In the learning-based approach, each pair of questions is an *instance* represented by a set of *features*. The main challenge of such an approach relies on carefully extracting features that can help the learning model to accurately classify if two arbitrary questions are similar or not.

To determine if two questions are similar, we used supervised learning models (recall Section 3). For supervised learning, each instance is given a label of either "Yes", if the two questions are similar, or "No" if the two questions are different.

Figure 1 overviews our machine learning approach which consists of five phases: (1) obtaining and cleansing the dataset (*Dataset* step in Figure 1), described in Section 4.1, (2) `Feature Extraction`, described in Section 4.2, (3) `Feature Selection`, explained in Section 4.3, (4) `Model Training` and (5) `Model Testing`, and `Reporting Results`, discussed in Section 5.2.

4.1 Preparing and Cleansing the Dataset

To develop our machine-learning-based technique, we used a real-world dataset shared confidentially with us by Mawdoo3 company, a leading Arabic content platform that allows users to ask and answer questions.

The dataset consists of 8,997 pairs of Arabic questions (17,994 questions) manually labeled by the Mawdoo3 company. Each pair of questions, an instance in the dataset, is labeled either with class "Yes", if the two questions are similar, or with class "No", if the two questions are not similar. Table 1 shows the number of instances on each class. Based on the table, there are 4,784 pairs of similar questions, i.e., 50% of the dataset, and 7,188 pairs of questions that are not similar, i.e., 70% of the dataset.

Table 1: Number of instances in the original dataset.

# of Yes	**# of No**	**Total**
4,784	7,188	11,972

Table 2 shows a randomly selected pair of questions that belong to the "Yes" class and another pair of questions that belong to the "No" class. In the first row of Table 2, `Question 1` asks about the birth city of the comprehensive thinker Al-Razi where as `Question 2` asks about the city of his museum. Clearly, those two questions are not similar since they are asking about two different things. On the other hand, `Question 1` and `Question 2` in the second row of Table 2 are asking about the first country to start the Communism political ideology in different ways.

Table 2: An example of two instances, two pairs of questions, where each instance belong to a class from our dataset.

Question1	**Question2**	**Label**
اين ولد الرازي؟	في اي مدينة يقع متحف الرازي؟	No
اين بدأت الشيوعية ؟	في اي دولة ظهرت الشيوعية ؟	Yes

Since, in the original dataset, 70% of instances belong to the "No" class, this would make the decision of the learning model bias to the No label. To avoid such bias in the learning process, we only included 9,568 pairs of Arabic questions distributed evenly, i.e., 4,748 instances belong to the "Yes" class and the same number of instances belong to the "No" class as shown in Table 3. This dataset used to train and test the models.

Table 3: Number of instances in the reduced dataset to avoid any bias in the learning process.

# of Yes	**# of No**	**Total**
4,748	4,748	9,568

Data cleansing is an important step in machine learning since it ensures that all instances have correct labels, remove any duplicate, correcting corrupt instances or missing labels, etc. Therefore, after we collected our dataset, we processed it and removed any unnecessary symbols such as ", (,), and _ and we added "?" for questions without question marks.

4.2 Feature Extraction

Feature extraction is a crucial step for machine learning approaches since all subsequent training, testing, and generalization steps depend on it. Therefore, we carefully designed and extracted novel features inherited from the Natural Language Processing (NLP) field. To that end, for each question, we extracted features that belong to five categories: morphological, semantic lexical, syntactic, semantic, and lexical features.

This section described the features that we extracted as well as the tools we used to extract them.

4.2.1 Extracting Lexical Features

Regarding to the lexical features, we computed the `Cosine` and the `Jaccard` similarities of two questions. Cosine similarity and Jaccard similarity measure the distance between two objects represented as vectors. If the distance is small, then the two objects are considered similar, otherwise, they are not similar.

These measures are widely used in text classification tasks. Jaccard similarity depends on the overlapping (same) word in two questions. Where as to compute the Cosine similarity, we leveraged the **word embedding** technique to represent each question as a numerical vector and then we calculated the Cosine similarity (the distance) between the two vectors.

Word embedding is a vector representation of a word. It is considered one of the most popular representations of text vocabulary. It can capture the context of a word in a text, such as semantic similarity, syntactic similarity, relations with other words, and many more. There are many techniques to generate word embeddings vectors such as Word2Vec (code.google.com/archive/p/word2vec), GloVe (nlp.stanford.edu/projects/glove), and FastText (fasttext.cc). In this paper, we use FastText to generate word embedding vectors since it is the only one that supports the Arabic language.

Lexical features are calculated using the below equations.

- *Cosine Similarity.* It is calculated for each pair of questions to measure how similar they are to one another. The resulting values are between 0 and 1 ($0 \leq \cos(A,B) \leq 1$). Given an input of two questions **A** and **B**, we use word embedding to represent both questions **A** and **B** as vectors. Cosine similarity is calculated using the dot product and magnitude as shown in Equation 3.

$$\cos(\mathbf{A},\mathbf{B}) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n}\mathbf{A}_i\mathbf{B}_i}{\sqrt{\sum_{i=1}^{n}(\mathbf{A}_i)^2}\sqrt{\sum_{i=1}^{n}(\mathbf{B}_i)^2}} \quad (3)$$

– *Jaccard Similarity.* It measures the similarity between two finite sets, by calculating the number of overlapped words in questions over the number of unique words in them. The resulting values are between 0 and 1 ($0 \leq$ J(A,B) ≤ 1). Given an input of questions **A** and **B**, Jaccard similarity is calculated using Equation 4.

$$J(\mathbf{A},\mathbf{B}) = \frac{\|\mathbf{A}\cap\mathbf{B}\|}{\|\mathbf{A}\cup\mathbf{B}\|} = \frac{\|\mathbf{A}\cap\mathbf{B}\|}{\|\mathbf{A}\| + \|\mathbf{B}\| - \|\mathbf{A}\cap\mathbf{B}\|} \quad (4)$$

4.2.2 Extracting Morphological, Semantic, and Syntactic Features

There are many data analysis tools to analyze and extract features from Arabic scripts but the most widely used by researchers in the NLP community is MADAMIRA [Pasha et al., 2014]. MADAMIRA is a comprehensive Arabic analyzer developed based on the aggregation of two systems MADA [Habash and Rambow, 2005] and AMIR [Diab et al., 2007] and we used it to extract the morphological, semantic, and syntactic features.

Arabic Morphology [Habash, 2010] is a critical analysis for NLP and it focuses on the meaning and surface form of the words. Arabic Morphology is divided into three categories:

– *Inflectional Morphology.* Inflectional morphology determines the forms (grammatical categories) of words after changing the affixation, adding morpheme or affix, or vowel to the words. There are two categories of inflectional morphology features: the verbs category features such as `aspect`, `mood`, `voice`, etc, and the subject category features such as `person`, `gender`, `number`, etc.

– *Cliticization Morphology.* A clitic is a word or part of a word that does not stand by itself and it depends on its neighboring words such as "m" in the word "I'm". Cliticization morphology features are `Proclitic 0`, `Proclitic 1`, `Proclitic 2`, `Proclitic 3`, and `Enclitic 0`.

– *Derivational Morphology.* Derivational morphology is the process of creating new words from existing words mainly by adding prefix or suffix to words. By doing so, the grammatical category of the words could be changed from one category to another. An example of the derivational morphology is the Part-of-Speech (PoS) feature, which is considered a syntactic feature.

In addition to the previously extracted features using MADAMIRA, we also extracted the `Named Entity Recognition (NER)` semantic feature. NER, or called entity identification, is the process to locate and classify the named entity mentioned in a text.

Table 4 presents the morphological, syntactic, and semantic featured that we extracted from our dataset using MADAMIRA. Based on the table, the total number of the extracted features is 19 divided as follow: 17 morphological features, 1 syntactic which is the PoS feature, and 1 semantic which is the NER feature.

Table 4: The extracted features using the MADAMIRA tool.

Feature Name and Abbreviation	**Feature Type**
Stem	Morphological
Gloss	Morphological
Lemma	Morphological
State (Stt)	Morphological
Case (Cas)	Morphological
Person (Per)	Morphological
Voice (Vox)	Morphological
Aspect (Asp)	Morphological
Mood (Mod)	Morphological
Gender (Gen)	Morphological
Number (Num)	Morphological
Proclitic 0 (Prc0)	Morphological
Proclitic 1 (Prc1)	Morphological
Proclitic 2 (Prc2)	Morphological
Proclitic 3 (Prc3)	Morphological
Enclitics 0 (Enc0)	Morphological
Base Phase Chunks (BPC)	Morphological
Part-of-Speech (PoS)	Syntactic
Named Entity Recognition (NER)	Semantic

4.2.3 Overlapping Features

In addition to the previously extracted features, we followed the work of [Mohammad et al., 2017] for extracting the lexical, syntactic, and semantic overlapping features. For each pair of questions (Q1 and Q2), we computed the n-grams and the stemmed n-grams for each question (uni-grams, bi-grams, and tri-grams). Next, we computed the lexical overlap features, the Precision, Recall, and F-measure, as shown in Equations 5, 5, and 7, respectively. Resulting in 9 lexical overlap features from the n-grams and another 9 lexical overlap features from the stemmed n-grams.

$$Precision_{lexical_n} = \frac{\text{Number of overlapping n-grams}}{\text{Number of n-grams in Question 1}} \tag{5}$$

$$Recall_{lexical_n} = \frac{\text{Number of overlapping n-grams}}{\text{Number of n-grams in Question 2}} \tag{6}$$

$$F - measure_{lexical_n} = 2 * \frac{Precision_{lexical_n} * Recall_{lexical_n}}{Precision_{lexical_n} + Recall_{lexical_n}} \tag{7}$$

As Das and Smith [Das and Smith, 2009] showed that lexical overlap features for two sentences could be high based on their lexical features, they have similar surface, even though they have different meaning. To avoid such a case, we computed the syntactic overlap features using the POS tags and we also computed the semantic overlap features based on the NER tag.

Equations 8, 9, and 10 calculates the Precision, Recall, and F-measure of the syntactic overlap features, respectively.

$$Precision_{syntactic_n} = \frac{\text{Number of overlapping POS tags}}{\text{Number of POS tags in Question 1}} \tag{8}$$

$$Recall_{syntactic_n} = \frac{\text{Number of overlapping POS tags}}{\text{Number of POS tags in Question 2}} \tag{9}$$

$$F - measure_{syntactic_n} = 2 * \frac{Precision_{syntactic_n} * Recall_{syntactic_n}}{Precision_{syntactic_n} + Recall_{syntactic_n}} \tag{10}$$

Equations 11, 12, and 13 calculates the Precision, Recall, and F-measure of the semantic overlap features, respectively.

$$Precision_{semantic_n} = \frac{\text{Number of overlapping NER tags}}{\text{Number of NER tags in Question 1}} \tag{11}$$

$$Recall_{semantic_n} = \frac{\text{Number of overlapping NER tags}}{\text{Number of NER tags in Question 2}} \tag{12}$$

$$F - measure_{semantic_n} = 2 * \frac{Precision_{semantic_n} * Recall_{semantic_n}}{Precision_{semantic_n} + Recall_{semantic_n}} \tag{13}$$

4.2.4 Semantic Lexical Features

Semantic lexical features have also been extracted. Mainly, the complexity features, Latent Dirichlet Allocation (LDA) features, and the Term Frequency–Inverse Document Frequency (TF-IDF) features. This section describes each one of them in more details.

– *Complexity Features.*

  Based on the NLP computations, the complexity features can be calculated at two levels: the sentence-level and the word-level complexities. The sentence-level includes features such as number of words in the sentence, the depth of the sentence tree, the name phrase in the sentence, or the verb phrase in the sentence, etc. These complexity features can be extracted using tools such as the Stanford Parser [Green and Manning, 2010]. On the other hand, word-level complexity features includes features such as number of characters in the word, lexical diversity of the vocabulary (AKA Type Token Ratio TTR), etc. In our work, we computed, as complexity features, the number of words in each question and the number of characters in each question.

– *LDA Features.*

  The Latent Dirichlet Allocation (LDA) [Blei et al., 2003] is a powerful tool mostly used in topic modeling. In our work, we utilize the LDA topic modeling technique to extract the topic each question belong to. The LDA model depend on Bag-of-Words (BoW) and bi-grams to extract the features.

– *TF-IDF Features.*

  Term Frequency–Inverse Document Frequency (TF-IDF) [Aizawa, 2003] measure the importance of a word in a sentence using a numerical statistical model. TF-IDF is heavily used in information retrieval and text mining applications. In our work, we computed the importance of the words in each question using the TF-IDF technique.

4.3 Features Selection

After we extracted the features, we used *Random Forest (RF)* [Reif et al., 2006] to automatically select the important features from our 36 features. Feature selection helps for building learning models faster, increase their performance, and reduces the possibility of overfitting [Guyon and Elisseeff, 2003].

Random Forest uses a measurement called impurity to find the best features that split the data. Thus when training a tree, it computes how much each feature decreases the weighted impurity of the tree, then the features are ranked based in their effect of decreasing the impurity of the tree. Figure 2 shows the importance values of the features ordered descending as obtained from running the Random Forest feature selection algorithm on our dataset. From Figure 2 , We chose the top 17 most important features for building our learning models. The top 17 features out of the 36 extracted features are: Cosine Similarity, Jaccard Similarity, Overlapping Features, Gender, State, Complexity Features, and LDA.

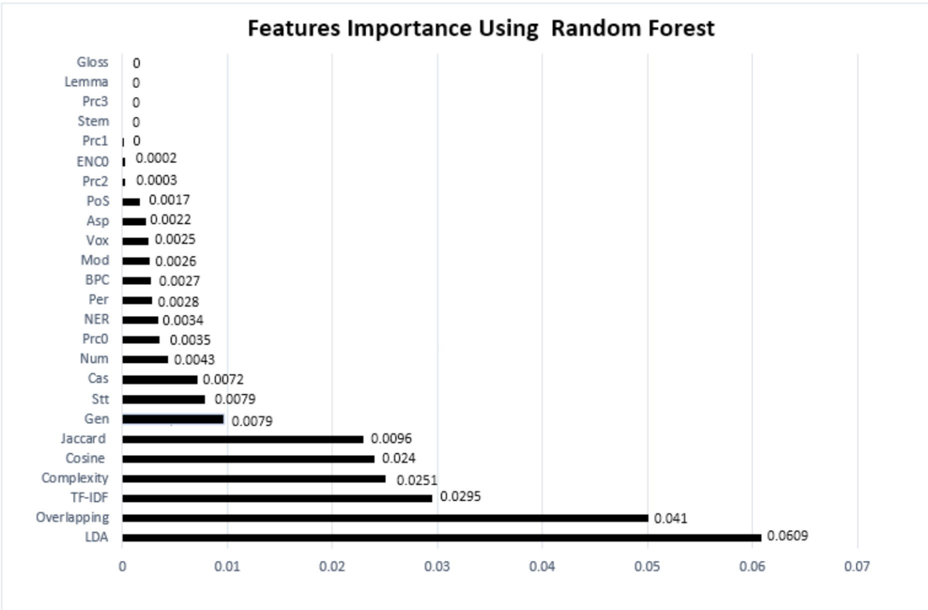

Figure 2: The importance values of all features as obtained from the Random Forest feature selection ordered descending based on their importance.

5 Experimental Design and Results

This section describes the evaluation metrics we used to measure the performance of each machine learning model. Next we report the performance of each machine learning model and compare between them.

5.1 Model Evaluation Metrics

To evaluate and compare between the various machine learning classifiers, we calculated and reported different measurements such as *Precision*, *Recall*, *F-measure*, and the *Accuracy*. This section provides a brief description of each one of them to help the reader understands them and their differences.

5.1.1 Precision

Precision measures the ability of a model to correctly detect relevant (true) instances among the retrieved instances calculated using Equation 14.

$$Precision = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}} \tag{14}$$

5.1.2 Recall

Recall measures the ability of a model to identify correct instances over all relevant instances in the dataset calculated using Equation refeq:r.

$$Recall = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}} \tag{15}$$

5.1.3 F-measure

F-measure is the harmonic mean of the precision and the recall. It is calculated using Equation 16.

$$F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{16}$$

5.1.4 Accuracy

Accuracy is the ratio of correctly predicted observation to the total observations. It is the simplest and most used measure to evaluation models. It is calculated using Equation 17 which is the same as Equation 18.

$$Accuracy = \frac{\textit{Number of Correct Predictions}}{\textit{Total Number of Predictions}} \tag{17}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

5.2 Evaluation Results and Findings

After we extracted the features from our dataset as explained in Section 4.1, we used a 9,568 pairs of Arabic questions to train and test our machine learning models. Table 5 shows the split of our dataset to trainign and testing datasets. As shown in the table, we split the data to 70% training dataset and 30% testing dataset. The training dataset consists of 6,700 pairs of questions in which the number of instances that belong to the "Yes" equals the number of instances that belong to the "No" class. The testing dataset contains 2,871 instances in which 1420 are "Yes" instances and 1451 are "No" instances.

Table 5: The Split of the Dataset to Training and Testing Datasets.

Data	**# of Yes**	**# of No**	**Total**
Train Data	3,364	3333	6,697
Test Data	1,420	1,451	2,871

Using the dataset mentioned in Table 5, we trained and tested our seven machine learning models and reported their performance. We conducted five different experiments in which each experiment included a different set of features. In the first experiment, we included all the features. On the second experiment, we only included the best 17 features determined by the Random Forest feature selection technique. On the third experiment, we only included the cosine similarity feature. On the fourth experiment, we only included the overlapping features. On the last experiment, we only included the semantic lexical features.

Table 6: Comparison of the Performance of the Seven ML Models Using *All* Features

Model	**Precision %**	**Recall %**	**F-Measure %**	**Accuracy%**
DT	68.01	77.50	72.44	70.43
MLP	75.39	79.79	77.53	76.80
RF	77.41	77.57	77.49	77.39
LR	75.67	78.40	77.01	76.52
XGB	82.93	82.29	82.61	82.62
AdaBoost	79.67	80.83	80.25	80.04
Linear SVM	76.78	79.93	78.33	77.81

Table 6 compares the performance of the seven machine learning models that we trained using all features. Based on the table the XGB model achieved the best with 81.61% F-measure where as the Decision Tree model achieved the worst with 72.44% F-measure. Comparing the results of this work over our preliminary work described in [Al-asa'd et al., 2019], we found that the performance, measured using the F-measure, of the XGB increased from 78.5% as in [Al-asa'd et al., 2019] to 81.61 as shown in Table 6 using all features.

Table 7: Comparison of the Performance of the Seven ML Models Using *the best 17* Features

Model	**Precision %**	**Recall %**	**F-Measure %**	**Accuracy %**
DT	74.30	73.26	73.78	73.88
MLP	76.29	80.00	78.10	77.50
RF	79.76	79.93	79.85	79.76
LR	75.43	78.47	76.92	76.38
XGB	83.45	82.29	82.87	82.93
AdaBoost	80.37	80.76	80.57	80.46
Linear SVM	76.74	79.03	77.87	77.46

Table 7 shows the results of our second experiment using the best 17 features as determined by the Random Forest feature selection technique. Based on the table, the XGB model achieved the best with F-measure 82.87% and the Decision Tree model achieved the worst with a 73.78% F-measure. Again, comparing the results of this work over our preliminary work described in [Al-asa'd et al., 2019], we found that the performance, measured using the F-measure, of the XGB increased from 78.2% as in [Al-asa'd et al., 2019] to 82.87 as shown in Table 7 using the best features.

Table 8 shows the results of our third experiment in which we trained each model on the Cosine Similarity feature only. Based on the table, the XGB model achieved the best with F-measure 65.66% and both the Support Vector Machine (SVM) and Logistic Regression (LR) models achieved the worst with 61.73% F-measure.

Table 9 shows the results of our fourth experiment in which we trained each model on the overlapping features only. Based on the table, the XGB model achieved the best with F-measure 72.79% and the Multilayer Perceptron (MLP) models achieved the worst with 70.08% F-measure.

Finally, Table 10 shows the results of our fifth experiment in which we trained each model on the semantic lexical features only. Based on the table, the XGB

Table 8: Comparison of the Performance of the Seven ML Models Using *the Cosine Similarity* Feature Only.

Model	Precision %	Recall %	F-Measure %	Accuracy%
DT	61.88	70.77	66.03	63.98
MLP	60.89	68.73	64.57	62.70
RF	62.37	69.79	65.87	64.23
LR	62.42	61.06	61.73	62.56
XGB	62.85	68.73	65.66	64.44
AdaBoost	62.36	68.94	65.48	64.05
Linear SVM	62.42	61.06	61.73	62.56

Table 9: Comparison of the performance of the seven ML models using the *Overlapping features* only.

Model	Precision %	Recall %	F-Measure %	Accuracy%
DT	71.57	71.97	71.77	72.00
MLP	68.04	72.25	70.08	69.49
RF	74.98	70.49	72.67	73.77
LR	69.55	72.54	71.01	70.71
XGB	73.54	72.04	72.79	73.35
AdaBoost	72.27	71.76	72.01	72.41
Linear SVM	70.03	71.27	70.65	70.71

Table 10: Comparison of the performance of the seven ML models using the *Semantic Lexical features* only.

Model	Precision %	Recall %	F-Measure %	Accuracy%
DT	72.89	52.08	60.75	66.25
MLP	71.87	72.92	72.39	72.10
RF	73.38	65.28	69.09	70.71
LR	70.73	68.96	69.83	70.11
XGB	75.78	77.15	76.46	76.18
AdaBoost	71.87	73.82	72.83	72.38
Linear SVM	71.83	68.89	70.33	70.85

model achieved the best with F-Measure 76.46% and the Decision Tree (DT) model achieved the worst with 60.75% F-Measure.

6 Discussion

To study the importance of each group of features on our best model, the XGB model, we conducted several experiments in which we trained our XGB model on all features except one group of features. To that end, Table 11 shows the results of this experiment. For example, after removing the Overlapping feature, the F-measure of the XGB became 78.20% after it was 82.61% on all features (see Table 11). This shows the impact of the overlapping features on the performance of the XGB model. Similarly, the performance, measured using the F-measure, of the XGB model went down from 82.61 on all features to 76.60% after removing the Semantic Lexical features.

These two declines in the performance of the XGB after removing one group, i.e., the Overlapping features and the Semantic Lexical features, confirm the importance of these two groups of features in the Arabic semantic text similarity tasks. Highlighting the additional contributions of this work over our preliminary work [Al-asa'd et al., 2019].

Table 11: The performance of each model using all features except on group of features.

Group Features	**Precision %**	**Recall %**	**F-Measure %**	**Accuracy%**
Semantic	81.75	81.81	81.78	81.71
Syntactic	82.52	82.29	82.41	82.38
Semantic Lexical	76.60	76.60	76.60	76.52
Overlapping	79.34	77.08	78.20	78.44
Morphological	83.04	82.29	82.66	82.69
Jaccard Similarity	82.64	81.67	82.15	82.20
Cosine Similarity	83.20	82.22	82.71	82.76

The results of our experiments conducted in this section and the previous section demonstrate that the XGB model along with our extracted features achieved the best in all experiments. Proofing the successfully of this model in solving Arabic Semantic Text Similarity (STS) tasks. Based on our experimental results, we encourage and recommend researchers and data scientists who are working in solving STS tasks and challenges to utilize our XGB model along with our features to perform their tasks.

7 Conclusion and Future Directions

A challenge facing the Question/Answer platforms is the ability to detect if a question being asked similar to a question in the database so they can serve their users faster and increase their satisfaction. Unfortunately, this problem remains a research and industrial challenge especially for the Arabic text. To address this challenge, we propose a novel Natural Language Processing approach that can detect if two Arabic questions are similar or not using their morphological, syntactic, semantic, lexical, semantic lexical, and overlapping similarity features. Moreover, we compared the performance of seven well known machine learning classifiers such as Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression (LR), Extreme Gradient Boosting (XGB), Random forests (RF), Adaptive Boosting (AdaBoost), and Multilayer Perceptron (MP). To extract the morphological, semantic, and syntactic features, we leveraged MADAMIRA where as to extract the lexical features, we calculated the Cosine similarity on the word embedding vectors of two questions and we also calculated the Jaccard similarity of two questions. Using a real-world dataset shared with us by Mawdoo3 company which consists of more than 11,972 labeled pairs of questions, our approach achieved 82.93% accuracy using our XGB classifier on the best selected features by the Random Forest feature selection technique.

Leveraging deep learning techniques to solve the Arabic question to question similarity problem and comparing the results with the traditional machine learning classifiers is an interesting avenue of future work.

References

[Ababneh et al., 2014] Ababneh, J., Almomani, O., Hadi, W., El-Omari, N. K. T., and Al-Ibrahim, A. (2014). Vector space models to classify arabic text. *International Journal of Computer Trends and Technology (IJCTT)*, 7(4):219–223.

[Aizawa, 2003] Aizawa, A. (2003). An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65.

[Al-Anzi and AbuZeina, 2017] Al-Anzi, F. S. and AbuZeina, D. (2017). Toward an enhanced arabic text classification using cosine similarity and latent semantic indexing. *Journal of King Saud University-Computer and Information Sciences*, 29(2):189–195.

[Al-asa'd et al., 2019] Al-asa'd, M., Al-khdour, N., Bni Younes, M., Khwaileh, E., Hammad, M., and AL-Smadi, M. (2019). Question to question similarity analysis using morphological, syntactic, semantic, and lexical features. In *Proceedings of the 16th ACS/IEEE International Conference on Computer Systems and Applications AICCSA 2019*. IEEE.

[Al-Shalabi et al., 2006] Al-Shalabi, R., Kanaan, G., and Gharaibeh, M. (2006). Arabic text categorization using knn algorithm. In *the Proc. of Int. multi conf. on computer science and information technology CSIT06*.

[Alhutaish and Omar, 2015] Alhutaish, R. and Omar, N. (2015). Arabic text classification using k-nearest neighbour algorithm. *The International Arab Journal of Information Technology*, 12(2):190–195.

[Ali and Rafi, 2018] Ali, B. and Rafi, M. (2018). Capturing discriminative attributes on text representation learning. In *Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence*, page 68. ACM.

[Babelscape, 2019] Babelscape (2019). BabelNet: a multilingual encyclopedic dictionary. `https://babelnet.org/`. Online; accessed August 2019.

[Bekkali and Lachkar, 2019] Bekkali, M. and Lachkar, A. (2019). An effective short text conceptualization based on new short text similarity. *Social Network Analysis and Mining*, 9(1):1.

[Bhopale et al., 2018] Bhopale, A. P., Kamath, S. S., and Tiwari, A. (2018). Concise semantic analysis based text categorization using modified hybrid union feature selection approach. In *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*, pages 1–7. IEEE.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

[Cer et al., 2017] Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

[Chen et al., 2009] Chen, Y., Garcia, E. K., Gupta, M. R., Rahimi, A., and Cazzanti, L. (2009). Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research*, 10(Mar):747–776.

[Das and Smith, 2009] Das, D. and Smith, N. A. (2009). Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics.

[Diab et al., 2007] Diab, M., Hacioglu, K., and Jurafsky, D. (2007). Automated methods for processing arabic text: from tokenization to base phrase chunking. *Arabic Computational Morphology: Knowledge-based and Empirical Methods. Kluwer/Springer*.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

[Duwairi, 2006] Duwairi, R. M. (2006). Machine learning for arabic text categorization. *Journal of the American Society for Information Science and Technology*, 57(8):1005–1010.

[Ezzeldin and Shaheen, 2012] Ezzeldin, A. M. and Shaheen, M. (2012). A survey of arabic question answering: challenges, tasks, approaches, tools, and future trends. In *Proceedings of The 13th International Arab Conference on Information Technology (ACIT 2012)*, pages 1–8.

[Froud et al., 2012] Froud, H., Lachkar, A., and Ouatik, S. A. (2012). A comparative study of root-based and stem-based approaches for measuring the similarity between arabic words for arabic text mining applications. *arXiv preprint arXiv:1212.3634*.

[Ghanem et al., 2018] Ghanem, B., Arafeh, L., Rosso, P., and Sánchez-Vega, F. (2018). Hyplag: Hybrid arabic text plagiarism detection system. In *International Conference on Applications of Natural Language to Information Systems*, pages 315–323. Springer.

[Gharib et al., 2009] Gharib, T. F., Habib, M. B., and Fayed, Z. T. (2009). Arabic text classification using support vector machines. *IJ Comput. Appl.*, 16(4):192–199.

[Green and Manning, 2010] Green, S. and Manning, C. D. (2010). Better arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 394–402. Association for Computational Linguistics.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182.

[Habash and Rambow, 2005] Habash, N. and Rambow, O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*, pages 573–580.

[Habash, 2010] Habash, N. Y. (2010). Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

[Hamza et al., 2019] Hamza, A., En-Nahnahi, N., Zidani, K. A., and Ouatik, S. E. A. (2019). An arabic question classification method based on new taxonomy and continuous distributed representation of words. *Journal of King Saud University-Computer and Information Sciences.*

[Hassan et al., 2019] Hassan, B., AbdelRahman, S. E., Bahgat, R., and Farag, I. (2019). Uests: An unsupervised ensemble semantic textual similarity method. *IEEE Access.*

[Islam and Inkpen, 2008] Islam, A. and Inkpen, D. (2008). Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10.

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Icml*, volume 99, pages 200–209.

[Khreisat, 2006] Khreisat, L. (2006). Arabic text classification using n-gram frequency statistics a comparative study. *DMIN*, 2006:78–82.

[Kotsiantis et al., 2007] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24.

[Landauer and Dumais, 1997] Landauer, T. K. and Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

[Lithgow-Serrano and Collado-Vides, 2019] Lithgow-Serrano, O. and Collado-Vides, J. (2019). In the pursuit of semantic similarity for literature on microbial transcriptional regulation. *Journal of Intelligent & Fuzzy Systems*, 36(5):4777–4786.

[Magooda et al., 2015] Magooda, A., Mahgoub, A., Rashwan, M., Fayek, M., and Raafa, H. (2015). Rdi system for extrinsic plagiarism detection (rdi_red). *Work. Notes PAN-AraPlagDet FIRE*, 2015:129–131.

[Mahmoud and Zrigui, 2017] Mahmoud, A. and Zrigui, M. (2017). Semantic similarity analysis for paraphrase identification in arabic texts. In *Proceedings of the 31st Pacific Asia Conference on Language, Information and Computation*, pages 274–281.

[Mohammad et al., 2017] Mohammad, A.-S., Jaradat, Z., Mahmoud, A.-A., and Jararweh, Y. (2017). Paraphrase identification and semantic text similarity analysis in arabic news tweets using lexical, syntactic, and semantic features. *Information Processing & Management*, 53(3):640–652.

[Nicosia et al., 2015] Nicosia, M., Filice, S., Barrón-Cedeno, A., Saleh, I., Mubarak, H., Gao, W., Nakov, P., Da San Martino, G., Moschitti, A., Darwish, K., et al. (2015). Qcri: Answer selection for community question answering-experiments for arabic and english. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 203–209.

[Pasha et al., 2014] Pasha, A., Al-Badrashiny, M., Diab, M. T., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., and Roth, R. (2014). Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *LREC*, volume 14, pages 1094–1101.

[Reif et al., 2006] Reif, D. M., Motsinger, A. A., McKinney, B. A., Crowe, J. E., and Moore, J. H. (2006). Feature selection using a random forests classifier for the integrated analysis of multiple data types. In *2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, pages 1–8. IEEE.

[Romeo et al., 2017] Romeo, S., Da San Martino, G., Belinkov, Y., Barrón-Cedeño, A., Eldesouki, M., Darwish, K., Mubarak, H., Glass, J., and Moschitti, A. (2017). Language processing and learning models for community question answering in arabic.

Information Processing & Management.

[Siolas and d'Alché Buc, 2000] Siolas, G. and d'Alché Buc, F. (2000). Support vector machines based on a semantic kernel for text categorization. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 5, pages 205–209. IEEE.

[Tang and Li, 2019] Tang, Z. and Li, J. (2019). Jointly considering siamese network and matchpyramid network for text semantic matching. In *IOP Conference Series: Materials Science and Engineering*, volume 490, page 042043. IOP Publishing.

[Wang et al., 2018] Wang, Y., Wu, C., Zheng, K., and Wang, X. (2018). Social bot detection using tweets similarity. In *International Conference on Security and Privacy in Communication Systems*, pages 63–78. Springer.

[Wang et al., 2006] Wang, Z.-Q., Sun, X., Zhang, D.-X., and Li, X. (2006). An optimal svm-based text classification algorithm. In *2006 International Conference on Machine Learning and Cybernetics*, pages 1378–1381. IEEE.

[Yeh et al., 2019] Yeh, J.-Y., Hsu, T.-Y., Tsai, C.-J., Cheng, P.-C., and Lin, J.-Y. (2019). On identifying cited texts for citances and classifying their discourse facets by classification techniques. *Journal of Information Science & Engineering*, 35(1).

[Zaher et al., 2017] Zaher, M., Shehab, A., Elhoseny, M., and Osman, L. (2017). A new model for detecting similarity in arabic documents. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 488–499. Springer.

[Zouaghi et al., 2011] Zouaghi, A., Merhbene, L., and Zrigui, M. (2011). Word sense disambiguation for arabic language using the variants of the lesk algorithm. *WORLD-COMP*, 11:561–567.