International Journal of Speech Technology
https://doi.org/10.1007/s10772-022-09966-9

ArSphere: Arabic word vectors embedded in a polar sphere

Sandra Rizkallah[1] · **Amir F. Atiya**[1] · **Samir Shaheen**[1] · **Hossam ElDin Mahgoub**[2]

Received: 10 February 2021 / Accepted: 15 January 2022

Abstract
Word embeddings mean the mapping of words into vectors in an *N*-dimensional space. ArSphere: is an approach that designs word embeddings for the Arabic language. This approach overcomes one of the shortcomings of word embeddings (for English language too), namely their inability to handle opposites (and differentiate those from unrelated word pairs). To achieve that goal the vectors are embedded onto the unit sphere, rather than onto the entire space. The sphere embedding is suitable in the sense that polarity can be addressed by embedding vectors at opposite poles of the sphere. The proposed approach has several advantages. It utilizes the extensive resources developed by linguistic experts, including classic dictionaries. This is in contrast to the prevailing approach of designing the word embedding using the concept of word co-occurrence. Another advantage is that it is successful in distinguishing between synonyms, antonyms and unrelated word pairs. An algorithm to design the word embedding has been derived, and it is a simple relaxation algorithm. Being a fast algorithm allows easy update of the word vector collection, when adding new words or synonyms. The vectors are tested against a number of other published models and the results show that ArSphere outperforms the other models.

Keywords Word embeddings · Word vectors · Arabic NLP · Antonymy detection · Synonymy detection · Polarity

1 Introduction

Word embedding is the idea of transforming a word into a numeric vector. The rationale for this approach is that machine learning algorithms operate on numerical values, so this step will make natural language processing (NLP) tasks more amenable to be implemented using these machine learning and neural network algorithms. Word embeddings work by designing a mapping between each word in the corpus to a high dimensional vector. The vectors are placed in such a way that words with similar meanings have their vectors close to each other, distancewise. Conversely, words that are different in meaning have vectors that are far away in distance.

Recent research has shown that word embeddings are beneficial in many applications, including sentiment analysis (Altowayan & Tao, 2016; Boudad et al., 2020; Dahou et al., 2016; Tubishat et al., 2018; Ye et al., 2018), named entity recognition (El Bazi & Laachfoubi, 2019; Habibi et al., 2017; Helwe & Elbassuoni, 2019; Pennington et al., 2014; Shaalan, 2014; Wu et al., 2015d), question answering (Gomaa and Fahmy 2014; Hammo et al., 2004; Medved & Horák, 2018; Nakov et al., 2019; Rizkallah et al., 2022; Shen et al., 2017) and recommender systems (Hasanzadeh et al., 2020; Rizkallah et al., 2021; Vasile et al., 2016).

Extensive work has been performed on word embeddings for the English language. However, there is a limited number of attempts for the Arabic language. In this paper we develop a model for Arabic word embeddings, hoping that this would fill a need for efficient Arabic NLP applications using machine learning and other algorithms, especially with the availability of several Arabic resources to work with (Aly & Atiya, 2013). The proposed method addresses a deficiency that exists in almost all other word embedding algorithms (for English or other languages). The problem is that they do not specifically handle opposites relations among the words. For example the distance among antonymous words would not necessarily be different from the distance among unrelated words.

An antonym is basically a pair of words that have opposite meanings, such as "جيد" (good), "سيئ" (bad) or "صعود" (rise), "هبوط" (drop). On the other hand unrelated words

✉ Sandra Rizkallah
sandrawahid@hotmail.com

[1] Computer Engineering Department, Faculty of Engineering, Cairo University, Giza, Egypt

[2] AlKhawarizmy Software, Giza, Egypt

Published online: 03 March 2022

are pairs that have nothing in common, for example "كتاب" (book), "سفينة" (ship) and "أسبوع" (week), "سيارة" (car). The two concepts of antonyms and unrelated word pairs are totally different and have to be distinguished in a successful word embedding method.

This is the issue that we set out to solve in our newly proposed word embedding. In our approach the word vectors are embedded onto a unit sphere, rather than an extended space like other algorithms. The dot product between two vectors reflects the meaning similarity of the corresponding words. Synonyms' vectors have a dot product close to 1, unrelated pairs' vectors have a dot product close to 0, and antonyms' vectors have a dot product close to -1. This way we clearly differentiate between these three types of word relatedness, especially between the antonyms and unrelated word pairs. The developed model therefore tackles the polarity (or opposites) issue by embedding the antonyms on opposite poles of the sphere. This clear differentiation between similarity scores of synonyms/unrelated words/antonyms is rarely addressed in the literature.

Moreover, we note that most of the word embeddings generated for Arabic in the literature rely only on the Arabic text that is available online, for example from social media, Wikipedia, World Wide Web and customer reviews. In this paper we rely on a more professional source of Arabic words. The collected Arabic data are extracted using a tool created by AlKhawarizmy Software (KS, 2018). KS possesses a comprehensive Modern Standard Arabic (MSA) "lexicon", which contains synonyms, antonyms, and morphological and semantic features of MSA words. The source of the KS tool is the well-known dictionary (Omar, 2008). KS then supplemented it with many contemporary words, (including many loan words) from various content, in order to contain MSA (Modern Standard Arabic) words that are commonly used in the press, and in contemporary writings, etc. About 98.5% of the Quranic words are covered, except those that have specific inflected forms not commonly used in contemporary Arabic text.

The developed approach is a simple relaxation algorithm. It is fairly fast, and it does not require an extensive training process as in the competing approaches that typically use deep learning. In addition, the generated vectors can be for composite words (two or more words attached together that have a particular meaning), such as "إبن الخال" (cousin) and "الإمارات العربية المتحدة" (United Arab Emirates). We treat a composite word as if it is a single entity with a unique meaning, and it has its specific vector. Another beneficial aspect is that the proposed method uses a very large vocabulary, and over 1.12 million similarity scores to train the model, most of them from the aforementioned reputable lexicon. The proposed word vector model uses an undiacritized dictionary. The reason for that is that the available Arabic texts in most applications are undiacritized. Even though having a diacritized word vector would create less word ambiguities, it would render it much less applicable. However, the proposed approach can very well be designed for diacritized dictionaries, if such an application is needed. The reason is that the underlying source dictionary of Omar (2008) is originally diacritized.

Word vector embeddings are beneficial in many applications. One particular application is sentiment analysis. One approach is to combine both lexical and word vectors methods. For example we can classify a review, whether it is positive or negative. We design a small sized lexicon of positive and negative words that act as seeds, then determine the polarity of a word in the sentence by measuring the distance of the word's vector to each vector of the positive seeds as well as those of the negative seeds in the lexicon (Rizkallah et al., 2020a). The use of the lexical approach in sentiment analysis for the Arabic language has been developed and thoroughly discussed in El-Beltagy and Ali (2013), Al-Ayyoub et al. (2015) and El-Beltagy et al. (2016). Moreover, the proposed model exploits the synonymy/antonymy constraints in learning the word vectors. This makes our model suitable for enhancing the performance of ontology matching tasks, suggested in Kolyvakis et al. (2018), which needs synonymy/antonymy information to learn semantic word representations. Other suitable models for Arabic language ontology are proposed in Mezghanni and Gargouri (2017) and Ghoniem et al. (2019).

In summary the contributions of this work are the following:

- Propose a word embedding approach that tackles the word polarity issue.
- The proposed approach can successfully distinguish between synonyms, antonyms, and unrelated words, in contrast with the majority of existing works.
- It is based on embedding the word vectors on a sphere, whose geometric aspect is conducive to modeling opposing words as vectors on opposite poles of the sphere.
- It is a semi-supervised approach, and has the advantage of making use of the extensive resources that are developed by linguistic experts, such as authoritative dictionaries.
- The developed algorithm is a fairly fast relaxation algorithm, and tends to be faster than the prevailing approaches that are based on deep neural networks.
- Having a fast algorithm allows for easy retraining of the vectors, whenever we need to update the dictionary, or add new synonyms.

The rest of the paper is organized as follows: Sect. 2 presents the work done in literature in the field of word embeddings with a focus on Arabic language. Section 3 explains the proposed approach. How the used Arabic vocabulary

is collected and analyzed is presented in Sect. 4. Section 5 covers the results obtained and comparisons with the other existing approaches. Finally, Sect. 6 highlights the main conclusions and contributions.

2 Literature review

The early pioneers on word embeddings include Google's word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), fastText (Mikolov et al., 2017) and others, which are designed for the English language. The word2vec method is based on the continuous skip-gram model. This is accomplished by a training method that obtains vector representations that predict the surrounding words. As well as introducing the continuous bag-of-words model where the current word is predicted based on the surrounding words. The GloVe method of word vector modeling uses an approach that combines the approaches of global matrix factorization and local context window. They make use of ratios of co-occurrence probabilities. The result is a global log-bilinear regression model, where the model directly captures global corpus statistics. The fastText embedding is built upon the continuous bag-of-words used in word2vec. The authors use a combination of known improvements to learn high-quality word vector representations. To obtain higher accuracy, the authors add position-dependent weights and subword features to the continuous bag-of-words model architecture.

After these pioneering methods many other approaches have been developed for the English language and also other languages. It is out of scope to review all these, as our focus is on the Arabic language. However to be more comprehensive, we present some of the approaches that tackle the antonymy detection issue in the English language. In Yih et al. (2012) the authors devised an approach based on latent semantic analysis (LSA) that detects antonyms but fails to differentiate between antonyms and unrelated word pairs. In Zhang et al. (2014) a Bayesian probabilistic tensor factorization model is introduced. The model uses pre-trained word embeddings. Another approach (Ono et al., 2015) introduces Skip-Gram with Negative Sampling (SGNS). The approach assigns antonyms low similarity scores. However, this way one cannot differentiate between antonyms and unrelated words. The authors in Dou et al. (2018) modified the objective function of Skip-Gram model to account for the polarity detection issue. In summary, among the works on English language word embedding very few tackle the polarity issue, and even those which do, are not very successful in distinguishing between antonyms and unrelated words. In parallel with the current work, we have also designed a word vector embedding that considers antonyms as well, but for the English language (Rizkallah et al., 2020b). In contrast with this work, our work here on the Arabic language uses classic resources developed by linguistic experts, and compiled in the AlKhawarizmy Software Tool, making the word relations highly authoritative.

A considerable amount of research has been produced in Arabic NLP (Zitouni, 2014). Also, much work has been developed for word embeddings for the Arabic language (Salama et al., 2018). AraVec (Soliman et al., 2017) is one of the prominent word embedding approaches for Arabic language. It relies on Arabic data collected from different sources including Twitter, World Wide Web (www) and Wikipedia (wiki). Their proposed word embedding model is based on word2vec's implementation (Mikolov et al., 2013) using continuous bag-of-words (CBOW) and skip-gram (SG) techniques. Experimentation is done to tune different hyperparameters, such as minimum count and window size. The models are distributional, in the sense that words occurring in similar context will have close vectors. The authors in Bojanowski et al. (2017) extend the skip-gram model (Mikolov et al., 2013) by incorporating subword information. In their method, morphology is modeled by considering subword units and representing words by a sum of its character n-grams. Their model is evaluated on nine languages including Arabic.

The authors in Altowayan and Tao (2016) use word-2vec's continuous bag-of-words model (Mikolov et al., 2013) to learn Arabic word embeddings. The Arabic corpus is collected from various sources including a local Arabic newspaper, Arabic editions of international news networks and consumer reviews. The embeddings are used for training binary classifiers to perform sentiment analysis. Again following Mikolov et al. (2013), the authors in Dahou et al. (2016) use continuous bag-of-words and skip-gram models to generate Arabic word embeddings. A web-crawled corpus for MSA and dialectal Arabic text is created. For sentiment classification, a convolutional neural network is trained on top of the pretrained vectors.

The authors in Zahran et al. (2015) build three models for Arabic word embeddings: continuous bag-of-words, skip-gram and GloVe. Moreover, cosine error regression neural network is used to map Arabic vectors to their corresponding English vectors using English-Arabic dictionary for training. Polyglot (Al-Rfou et al., 2013) used neural networks to train word embeddings for more than one language including Arabic. The corpus used for generating the embeddings is Wikipedia. The model is trained to distinguish between the original phrase and a corrupted version of the phrase, penalizing the scoring of the corrupted version. The authors in Malhas et al. (2016) propose a supervised learning approach in which a learning-to-rank model is trained over questions and answers extracted from Arabic community question answering forums. The features generated from the data are

word embeddings. These word embeddings are computed using word2vec (Mikolov et al., 2013).

In Al-Azani and El-Alfy (2017b) the authors used the word vectors designed in Altowayan and Tao (2016) as features for polarity determination in sentiment analysis. Several machine learning classifiers are trained. Ensemble learning methods are also applied to improve the performance of individual classifiers. In Al-Azani and El-Alfy (2017a) the authors used word2vec model (Mikolov et al., 2013) for generating Arabic word embeddings. They design several deep learning architectures using convolutional neural networks (CNNs) and Long Short Term Memory (LSTM) recurrent neural networks for the purpose of applying it to sentiment analysis.

Another Arabic word embedding model known as: ArWordVec is built using Arabic tweets (Fouad et al., 2020). The approach follows word2vec's continuous bag-of-words and skip-gram (SG) techniques (Mikolov et al., 2013). Moreover, the authors also used GloVe approach (Pennington et al., 2014). In Lachraf et al. (2019) the authors trained Arabic-English cross-lingual word embedding models again using continuous bag-of-words and skip-gram (SG) techniques. In Altowayan and Elnagar (2017), sentiment-specific Arabic embeddings are built using fasText tool. The word embedding implemented was shown to improve the accuracy of the sentiment analysis.

Other language models known as BERT (Devlin et al., 2018) have evolved. These are transformer based models developed to pre-train deep bidirectional representations using unlabeled text. Fine tuning can be applied to use such models for NLP tasks. AraBERT (Baly et al., 2020) and Multi-dialect-Arabic-BERT (Talafha et al., 2020) are pre-trained BERT-based models for the Arabic language.

To our knowledge, all the proposed Arabic word embeddings approaches do not tackle the polarity issue of words. This means that the problem of distinguishing between synonyms and antonyms still persists, because often synonymous pairs vectors as well as antonymous pairs vectors will lie in close proximity in the vector space. Moreover, there will still exist the inconsistency in handling antonyms i.e. in some cases antonyms will have similarity scores close to 1 (synonyms) and in other cases the scores will be close to 0 (unrelated). These are precisely the problems we attempt to solve in our proposed approach.

3 Proposed approach

This section details the proposed approach indicating the relevant algorithm and equations. We defined these equations in Rizkallah et al. (2020b) (a sphere-type embedding model for the English language), but we included them here too for convenience and readability.

Each word is represented as a unit vector in N-space. The fact that the length of the vector is 1 means that the word is embedded on a unit sphere. The dot product between two vectors represents the similarity between their corresponding words. Moreover, since the word vectors are unit vectors, the dot product and the distance are related in one to one way as shown as follows in (1):

$$\|x_i - x_j\|^2 = \|x_i\|^2 + \|x_j\|^2 - 2x_i^T x_j = 2\left[1 - x_i^T x_j\right] \quad (1)$$

where x_i is the vector corresponding to word i. For example, antonyms such as "صعود" (rise) and "هبوط" (drop) lie at two opposite poles of the sphere, so their vectors' dot products is close to -1. On the other hand, synonyms such as "سعادة" (happiness) and "بهجة" (joy) lie at a close distance, so their vectors' dot products is close to 1. This is shown in Fig. 1. Text or verb negation in our case amounts to multiplying the word vector by -1, thus turn it to become an antonym.

Let s_{ij} be the target similarity score of a pair of words. We develop an algorithm that estimates the vectors x_i that would yield the similarity numbers as close as possible to the given similarity numbers. Any vector x_i is therefore placed optimally, so that it satisfies the similarity scores with the other word vectors in the training set. The target similarities s_{ij} are determined by the designer. We typically select these target similarity scores such that they are close to 1 for synonym pairs, and close to -1 for antonym pairs, and zero for unrelated words.

Initially we feed the algorithm with a set of word pairs associated with similarity scores (training set). Then, other relations between words can be inferred after training. For example: if word1 and word2 are synonyms while word2 and word3 are antonyms, the algorithm will infer that word1 and word3 are antonyms. Of course, not all word pairs are synonyms, antonyms, or unrelated. The relation is graded, and the algorithm does estimate a graded score.

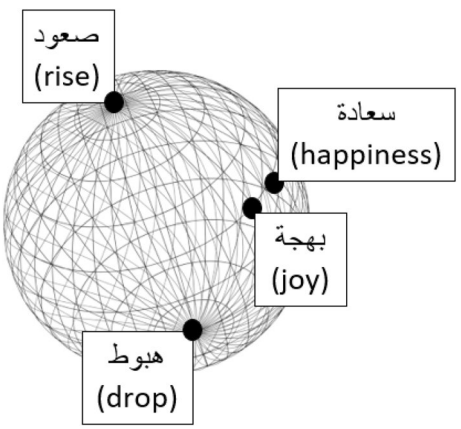

Fig. 1 An illustration of the embedding sphere of the proposed model, showing some synonyms and antonyms

For example "يجري" (run) and "يمشي" (walk) are not exactly synonyms, but are related. The algorithm may assign a similarity score of around 0.8.

We define the following error function:

$$E = \sum_{i \neq j} \sum_{j} w_{ij} \left[x_i^T x_j - s_{ij} \right]^2 \quad \text{subject to } \|x_i\| = 1, i = 1, \ldots, K \tag{2}$$

where w_{ij} is a weighting coefficient that represents the confidence in the similarity estimate s_{ij}, and $\|x_i\|$ represents the length of vector x_i.

Solving this large optimization problem by obtaining all K vectors x_i at once, is hard so we propose a relaxation algorithm, which tackles one x_k at a time. In this algorithm we focus on some x_k and fix all other x_l's for the time being. Then we optimize E w.r.t. x_k. This is performed using a simple equation. Then we cycle through each of the other x_l's, one by one, and obtain their optimal values. The details are described as follows.

We focus here on x_k as the vector to be optimized. Using a Lagrange multiplier to take into account the unity norm constraint, we obtain

$$V = \sum_{i \neq j} \sum_{j} w_{ij} \left[x_i^T x_j - s_{ij} \right]^2 - \lambda(\|x_i\|^2 - 1) \tag{3}$$

After some simplifications we formulate the simplified augmented objective function (4):

$$V \equiv x_k^T A x_k - 2 b^T x_k + c \tag{4}$$

where A; b; c matrices are defined as:

$$A = \sum_{j \neq k} w_{kj} x_j x_j^T - \lambda I \tag{5}$$

$$b = \sum_{j \neq k} w_{kj} x_j s_{kj} \tag{6}$$

$$c = \sum_{j \neq k} w_{kj} s_{kj}^2 + \lambda \tag{7}$$

Taking the derivative w.r.t. x_k and equating to zero, we get

$$\frac{dV}{dx_k} = 2Ax_k - 2b = 0 \tag{8}$$

$$x_k = A^{-1} b \tag{9}$$

To evaluate λ we enforce the condition $x_k^T x_k = 1$.

$$x_k^T x_k = b^T A^{-2} b = 1 \tag{10}$$

where we used the fact that A is a symmetric matrix. We get

$$b^T \left[A' - \lambda I \right]^{-2} b = 1 \tag{11}$$

where

$$A' = \sum_{j \neq k} w_{kj} x_j x_j^T \tag{12}$$

To evaluate λ at every iteration, we implemented a one-dimensional search since λ is just a scalar. The search will find the value of λ such that Eq. (11) is satisfied. The one-dimensional search is done using the bisection method. After obtaining x_k as above, we apply the same procedure to get the next vector, and so on. After obtaining all the vectors, another cycle is performed. The cycles continue until convergence. Convergence means that the algorithm terminates, in the sense that each cycle ultimately results in very small changes in the vectors' positions. When that happens the similarity scores will be satisfied with good accuracy. This is because each cycle leads to a decrease in error (given by Eq. (2)) until the decrease in error is insignificant. In all our runs it did converge, and in a rather small number of cycles, typically less than 100.

Note that the vector x_k is initialized randomly as any unit-length vector. We also note that the weighting factor w_{ij} in Eq. (2) has the purpose of providing confidence weighting in the suggested similarity score. For example, if the similarities are obtained from a human expert or a renowned dictionary, then the confidence in the score is higher than the case when they are obtained from a not too trustworthy source.

Figure 2 summarizes the algorithm for learning our spherical word vectors. Once learning is complete, we may like to estimate the similarity between some two words k and l for some application. We simply consider their corresponding vectors x_k and x_l and perform a dot product. This dot product is the similarity score.

4 Arabic data collection and analysis

The training set for our model is a set of word pairs with predefined estimated similarity scores. Synonym pairs, unrelated word pairs and antonym pairs are assigned similarity scores close to 1, 0 and − 1 respectively. To collect such a set we have initially gathered a list of 2000 most frequently used MSA words. This set of words acts merely as a seed that will be grown to an extensive vocabulary. The next step was to obtain synonyms and antonyms for these words. This was done using the AlKhawarizmy Software (KS) dll tool. A word is given as input, and we obtain its synonyms and antonyms as output. It was not merely a one-time application of the dll tool on the 2000 words. It was more a chain-like application, where each newly obtained word (through a synonym or antonym) is fed again to the tool to extract

Algorithm: Learning Word Vectors
Inputs: Similarity_Pairs, Weight_Pairs, Vector_dimension
Outputs: Word Vectors

```
For each word k in the vocabulary:
     Generate a Random Vector Xk of dimension Vector_dimension
Compute Error using Eq.(2)
Repeat until Convergence:
     For each word k:
          Compute matrix A` using Eq.(12)
          Compute vector b using Eq.(6)
          Apply one-dimensional search to find scalar λ satisfying Eq.(11)
          Compute matrix A using Eq.(5)
          Compute vector Xk using Eq.(9)
     Compute Error using Eq.(2)
```

Fig. 2 Learning spherical vectors algorithm

more words/synonyms/antonyms. This sequence of applications can be repeated many times to increase the size of the obtained vocabulary. We have performed two applications until the vocabulary size was considerably increased.

In addition to this, we have collected some extra pairs of words that are important synonyms/antonyms and were missing from the previous corpus, many of them being contemporary words. The added pairs are for the purpose of increasing the vocabulary of the developed model. These pairs are for example added by extracting synonyms and antonyms using the KS tool for Arabic words that appear in datasets such as the "ASTD" dataset (Nabil et al., 2015). Moreover, we have added unrelated word pairs randomly from the collected vocabulary (after manually checking that these pairs are indeed unrelated). This is in order to train the model how to handle any unrelated words. We have also added composite words, Arabic lists of countries, capitals, country-capital, colors, animals, family, etc. The designed vectors are useful for many applications such sentiment analysis (SA). If the text is dialect then it could be translated to its equivalent MSA words (Mahgoub et al., 1990), where it could use a standardized MSA word embedding or machine learning for the sentiment prediction (Rizkallah et al., 2018) (The KS tool also provides dialect to MSA translation).

Figure 3 shows the developed algorithm for constructing the vocabulary. The algorithm works by passing over each word in the initially collected set of 2000 MSA words and obtaining its synonyms and antonyms from the KS dll. The initial words together with the collected synonyms and antonyms form a new set of words where for the newly collected words (through the dll), synonyms and antonyms are obtained from the KS dll.

To achieve best results, we modeled the collected vocabulary as a graph where words are vertices or nodes, relations between words that exist in the training set are edges and similarity scores are weights on the edges. This modeling is beneficial because we can analyze this graph and obtain the number of connected components in the graph. A connected component is a group of vertices, which are connected to each other directly (through a direct edge), or indirectly through following a number of connecting edges from the origin vertex to the destination vertex. If we have some disconnected components, then this means that none of the words in a certain component has a relation (synonym/antonym/unrelated) with any other word in another disconnected component of the graph. In such a case the algorithm would not know where to place the disconnected components with respect to each other. There is no similarity score or distance between a component and another one disconnected from it, so that we can frame one component a certain distance with respect to the other. Thus, the vocabulary graph obtained from the approach may have some erroneous word similarity scores, with the errors increasing if the number of disconnected components increase. This problem can be solved by decreasing the number of disconnected components, ideally obtaining one giant connected component. We have achieved that by first identifying the connected components. Then we connected some of the words from these components, "building bridges" among the disconnected components. This is performed by adding these pairs of words (from the disconnected components) to the training set. The similarity scores of these pairs are estimated subjectively based on the words' meanings.

Table 1 and Fig. 4 show some key numbers and statistics of the final collected vocabulary. As can be seen, the size of the vocabulary and the number of the similarity relations are quite large, compared to those of other Arabic word vector works (we have over 1 million similarity relations). Also there are many edges (similarity relations) for every node (word). This makes the mapping more robust, because the vector for each word will be placed optimally based on the

Algorithm: Arabic Vocabulary Extraction
Inputs: Voc → Initial Vocabulary of 2000 MSA words
Outputs: New Vocabulary (NewVoc), Synonyms List and Antonyms List

```
NewVoc=Voc
For each word in Voc
    DllOutput ← GetSynonyms(word,KS dll)
    Synonyms ← Parse(DllOutput)
    For each synonym in Synonyms
          if(synonym not in Voc)
                NewVoc ← NewVoc+synonym
          Synonyms List ← Add synonym pair(word, synonym)
    DllOutput ← GetAntonyms(word,KS dll)
    Antonyms ← Parse(DllOutput)
    For each antonym in Antonyms
          if(antonym not in Voc)
                NewVoc ← NewVoc+antonym
          Antonyms List ← Add antonym pair(word, antonym)
end
For each word in NewVoc and not in Voc
    DllOutput ← GetSynonyms(word,KS dll)
    Synonyms ← Parse(DllOutput)
    For each synonym in Synonyms
          if(synonym not in NewVoc)
                NewVoc ← NewVoc+synonym
          Synonyms List ← Add synonym pair(word, synonym)
    DllOutput ← GetAntonyms(word,KS dll)
    Antonyms ← Parse(DllOutput)
    For each antonym in Antonyms
          if(antonym not in NewVoc)
                NewVoc← NewVoc+antonym
          Antonyms List ← Add antonym pair(word, antonym)
end
```

Fig. 3 Arabic vocabulary extraction algorithm

Table 1 Vocabulary specifications

Number of distinct words (graph nodes)	39,165
Number of synonyms pairs	617,884
Number of antonyms pairs	507,218
Number of unrelated pairs	63,272
Number of graph edges	1,188,374
Ratio of number of edges to number of nodes	30
Average degree of the graph	60
Number of connected components	1

consensus of similarity relations of the words connected to it in the graph.

The word vectors generated are publicly available at https://github.com/SandraRizkallah/ArSphere-Arabic-WordVectors.

5 Results and discussion

5.1 Similarity scores comparison

Distinction between various relations among words, such as synonyms, antonyms and unrelated, are essential for many applications including sentiment analysis, question answering, automatic translation, building lexicons and others. Moreover, the similarity scores between the words have to

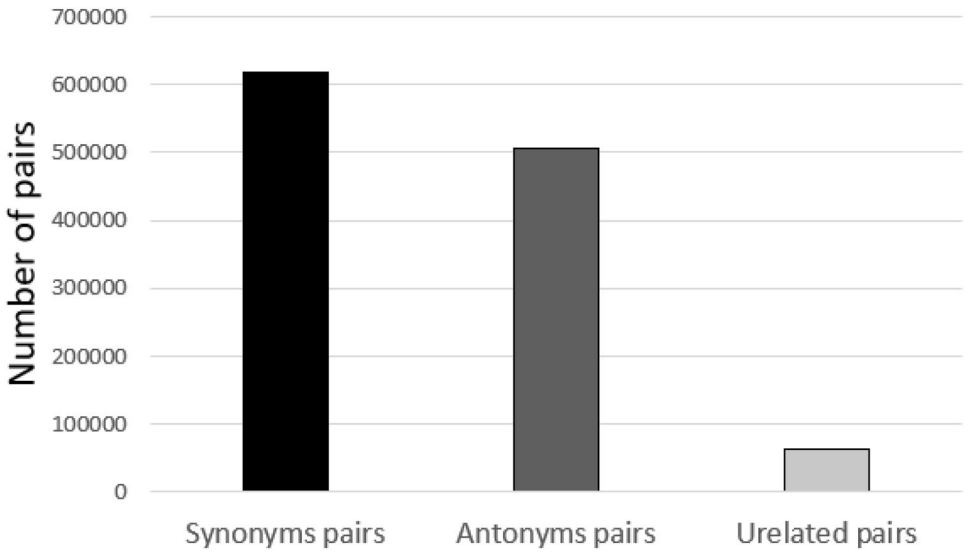

Fig. 4 Three similarity categories

reflect the real relation between these words based on their meanings. These scores reflect the real quality of the generated word vectors as now the word is substituted by a vector so this vector has to convey the semantics of the word. To test the quality of the generated vectors using the approach we implemented, we have included a table of word pairs together with the similarity scores between these pairs. The similarity score is computed as the dot product between two vectors. In our table, we sought to include word pairs belonging to the following different categories:

1. Synonyms that are in our training set
2. Synonyms that are NOT in our training set
3. Antonyms that are in our training set
4. Antonyms that are NOT in our training set
5. Unrelated words that are in our training set
6. Unrelated words that are NOT in our training set
7. Word pairs belonging to a certain list i.e. country-capital, colors, etc.

Of course, categories 2, 4, 6, and part of 7 correspond to a true unbiased test, since they are not part of the training set. In other words, the computed similarities are based on the vectors' positions and their dot products, and the method was not given their true similarities during the training phase.

The obtained similarity scores are compared with the similarity scores generated using other published Arabic word vectors. To unify the reference of comparison, all the vectors are normalized before obtaining the dot product. We have compared with a number of published vectors including:

- Polyglot (Al-Rfou et al., 2013), vector of dimension 256
- Altowayan-arabic-news (Altowayan & Tao, 2016), vector of dimension 300 referred to as Altow.
- Bojanowski (Bojanowski et al., 2017), vector of dimension 300 referred to as Bojan.
- AraVec (Soliman et al., 2017) N-grams CBOW twitter model, vector of dimension 100 referred to as AVm1
- AraVec (Soliman et al., 2017) N-grams SG twitter model, vector of dimension 100 referred to as AVm2
- ArWordVec (Fouad et al., 2020) SG 3 model, vector of dimension 300 referred to as ArW3
- ArWordVec (Fouad et al., 2020) SG 5 model, vector of dimension 300 referred to as ArW5

Note that the competing AraVec and ArWordVec approaches have several versions available. To have a more concise comparison, we have selected the best two of these versions, and have shown the results for these (as shown above). Our vectors (ArSphere) are of dimension 50. The computational time for our model is 3.6 h using an Intel® Core™ i3 CPU M350 @2.27GHz with 8 GB RAM running Windows 10. In Altowayan and Tao (2016) it is reported that their training time is around 1.7 hours using a 3.1GHz CPU with 16 GB of RAM. While the training time reported for AraVec (Soliman et al., 2017 is 10 h for the Wikipedia model, 1.5 days for the Twitter model and 4 days for the Common Crawl model using a Quad core Intel i7-3770 CPU @3.4 GHz with 32 GB of RAM running Ubuntu 16.04. The other models did not specify their training time.

Table 2 includes the comparison with other published Arabic vectors. If a cell contains N1/N2/NN, this means that there is no vector in the dictionary for word1/word2/word1 and word2 respectively.

Table 2 is a lengthy snapshot into the performance of ArSphere and the competing models. It is hard to develop an exhaustive test for word embeddings that gives a numerical score of performance, because of the somewhat subjective nature of word similarities. The best way to test the approach

Table 2 Performance comparison

word1	word2	ArSphere	Polyglot	Altow.	Bojan.	AVm1	AVm2	ArW3	ArW5
Synonyms in training set: Category (1)									
رد (reply)	جواب (answer)	0.791	0.447	0.245	0.428	0.692	0.64	0.476	0.463
بهجة (joy)	سعادة (happiness)	0.731	0.347	0.08	0.431	0.74	0.722	0.38	0.377
مرأة (woman)	سيدة (lady)	0.796	0.497	0.221	0.458	0.624	0.559	0.324	0.331
تصالح (reconciliation)	السلام (peace)	0.814	0.119	− 0.014	0.223	0.255	0.263	N1	N1
قدوة (model)	مثالية (perfect)	0.687	0.118	0.091	0.338	0.439	0.513	0.216	0.201
نموذج (sample)	مثال (example)	0.84	0.469	0.31	0.468	0.767	0.749	0.529	0.538
هائل (tremendous)	رائع (wonderful)	0.819	0.445	0.238	0.307	0.514	0.553	NN	NN
Synonyms NOT in training set: Category (2)									
والدة (mother)	والد (father)	0.576	0.686	0.476	0.654	0.838	0.876	0.577	0.596
تاريخ (date)	وقت (time)	0.539	0.211	0.077	0.187	0.353	0.342	0.225	0.223
أصدقاء (friends)	أصحاب (friends)	0.821	0.268	0.168	0.264	0.656	0.594	0.341	0.34
موهوب (talented)	مبدع (creative)	0.283	0.771	0.434	0.487	0.748	0.752	0.524	0.497
عباقرة (geniuses)	ذكاء (intelligence)	0.74	0.033	0.122	0.315	0.425	0.46	N1	N1
قاض (judge)	محكمة (court)	0.513	0.483	0.458	0.579	0.544	0.662	0.393	0.402
Antonyms in training set: Category (3)									
صواب (right)	خطأ (error)	− 0.877	0.427	0.415	0.429	0.668	0.706	0.456	0.409
سعد (happy)	شقاء (misery)	− 0.742	0.095	0.103	0.123	− 0.006	0.17	0.031	0.047
قوة (strength)	ضعف (weakness)	− 0.73	0.352	0.251	0.402	0.652	0.664	0.473	0.456
سأل (asked)	أجاب (answered)	− 0.826	0.735	0.272	0.572	0.474	0.573	0.244	0.329
رديء (poor)	جيد (good)	− 0.769	0.529	0.288	0.451	0.58	0.648	N1	N1
استورد (imported)	صدر (exported)	− 0.578	0.179	N1	0.184	0.077	0.203	N1	N1
كذاب (liar)	صادق (honest)	− 0.789	0.307	0.109	0.243	0.699	0.572	0.407	0.424
حزن (sadness)	فرح (joy)	− 0.587	0.583	0.289	0.39	0.798	0.724	0.544	0.545
Antonyms NOT in training set: Category (4)									
متطور (developed)	تخلف (backwardness)	− 0.611	− 0.091	− 0.014	0.112	0.279	0.382	0.121	0.133
هبوط (drop)	صعود (rise)	− 0.694	0.437	0.483	0.484	0.806	0.783	0.498	0.466

Table 2 (continued)

word1	word2	ArSphere	Polyglot	Altow.	Bojan.	AVm1	AVm2	ArW3	ArW5
طويل	قصير	− 0.855	0.803	0.539	0.726	0.827	0.785	0.538	0.528
(long)	(short)								
ناعم	خشن	− 0.695	0.885	N2	0.604	0.823	0.736	0.34	0.333
(smooth)	(rough)								
عال	منخفض	− 0.741	0.685	0.228	0.52	0.4	0.493	0.255	0.254
(high)	(low)								
نسى	استذكر	− 0.773	N2	0.035	0.355	0.158	0.285	N2	N2
(forgotten)	(recall)								
سجناء	حرية	− 0.216	0.234	0.122	0.375	0.301	0.353	0.192	0.221
(prisoners)	(freedom)								
باسم	عابس	− 0.884	N2	− 0.106	0.164	0.009	0.149	N2	N2
(smiling)	(surly)								
ميسر	صعب	− 0.818	0.31	0.029	0.282	0.13	0.309	0.233	0.225
(easy)	(difficult)								
Unrelated words in training set: Category (5)									
افترق	مبادل	− 0.026	N1	N2	0.135	0.112	0.195	N1	N1
(parted)	(exchanger)								
تاريخ	أبهج	− 0.008	N2	N2	0.199	− 0.168	0.16	N2	N2
(date)	(pleased)								
ثلاث	جودة	− 0.004	0	0.01	0.103	0.092	0.152	0.08	0.098
(three)	(quality)								
دولار	يوم	− 0.002	0.155	0.088	0.188	0.248	0.229	0.148	0.141
(dollar)	(day)								
دكتور	جزء	− 0.002	− 0.062	0.093	0.128	0.213	0.136	0.088	0.114
(doctor)	(part)								
اطمأن	تطبيب	0.095	N2	N2	0.301	0.282	0.335	NN	NN
(rest)	(medication)								
خطر	عمالقة	0.034	− 0.065	0.07	0.134	0.126	0.227	0.074	0.05
(danger)	(giants)								
Unrelated words NOT in training set: Category (6)									
القتال	القديم	0.279	− 0.052	− 0.065	0.128	0.069	0.173	− 0.044	− 0.061
(fighting)	(old)								
كشف	وارث	− 0.127	0.086	0.045	0.113	0.142	0.226	N2	N2
(revealed)	(heir)								
بهجة	إمبراطور	− 0.032	− 0.125	− 0.108	0.092	− 0.009	0.139	0.053	0.038
(joy)	(emperor)								
آل	عصرية	− 0.115	− 0.052	0.077	0.09	0.145	0.101	0.062	0.096
(supporters)	(trendy)								
عائلة	كتاب	− 0.019	0.011	− 0.008	0.113	0.277	0.255	N1	N1
(family)	(book)								
أتباع	الحقيبة	− 0.065	− 0.118	− 0.07	0.066	− 0.036	0.218	0.081	0.103
(followers)	(bag)								
مفاتيح	باخرة	− 0.109	0.194	− 0.018	0.173	0.215	0.239	N2	N2
(keys)	(ship)								
سال	أجاب	− 0.027	0.412	0.039	0.184	0.474	0.573	0.244	0.329
(flowed)	(answered)								
Lists: Category (7)									
مصر	القاهرة	0.787	0.614	0.503	0.671	0.777	0.809	0.586	0.612
(Egypt)	(Cairo)								
الإمارات	بلد	0.849	N1	N1	N1	N1	N1	N1	N1

Table 2 (continued)

word1	word2	ArSphere	Polyglot	Altow.	Bojan.	AVm1	AVm2	ArW3	ArW5
العربية المتحدة (United Arab Emirates)	(country)								
زرافة (giraffe)	حيوان (animal)	0.847	N1	N1	0.475	0.526	0.609	N1	N1
إبن_الأخ (nephew)	عائلة (family)	0.822	N1	N1	N1	N1	N1	NN	NN
حفيد (grandson)	حفيدة (granddaughter)	0.604	0.632	0.344	0.696	0.756	0.801	0.421	0.377
غزال (deer)	زرافة (giraffe)	0.602	N2	N2	0.383	0.543	0.57	N2	N2
فرنسا (France)	كندا (Canada)	0.833	0.59	0.356	0.467	0.809	0.804	0.513	0.551
أسود (black)	رمادي (grey)	0.329	0.807	0.296	0.65	0.678	0.694	0.493	0.522
بنفسجي (purple)	أزرق (blue)	0.659	0.836	N1	0.661	0.798	0.813	N1	N1

is to run on many word pairs and have the reader subjectively judge whether the estimated word similarities are sensible. We must mention that the experiments performed are not a true comparison between word vector methods in the true sense of the word. The presented results are mainly an illustration. The competing approaches are unsupervised while our approach is semi-supervised, so the conditions of design of the word vectors may be different. This may put the competing methods at some unfair disadvantage when comparing.

From the results obtained in Table 2, we observe the following findings.

- Our proposed model provides accurate estimation of the similarities of most synonyms (close to 1), most antonyms (close to − 1), and unrelated words (close to zero).
- Even though the target similarities for our model are either 1, − 1, or 0, the algorithm typically converges to values that are close to these, but not exactly. The reason is that for every vector, there is a competition from other vectors to pull the considered vector towards or away, in a way that would place it in a position that suitably satisfies the collective synonym/antonym relations.
- Considering "synonyms in training set" category, our approach is better than the other models; it successfully assigned similarity scores higher than 0.5 for all the introduced 7 pairs followed by AraVec N-grams SG twitter model which succeeded in 6 out of 7 pairs. Considering "synonyms not in training set" category, our approach successfully assigned similarity scores higher than 0.5 for 5 out of the 6 introduced pairs followed by the two AraVec models that succeeded in 4 out of 6 pairs. For the pair "موهوب" and "مبدع" that our approach missed (assigning it a similarity score lower than 0.5), the other models assigned better similarity scores close to or higher than 0.5.
- In the proposed model and also the competing models unrelated word pairs have similarity scores close to zero, which is the correct way to have.
- In our model, antonyms stand out by having negative similarity scores unlike the competing models, whereby antonyms are treated in some cases as synonyms having scores close to 1, and in other cases as unrelated words having similarity scores close to 0. Therefore, often one cannot differentiate between synonyms, antonyms and unrelated words in the other models.
- Our approach has a number of composite words, and some of them even consist of three words.
- In some of the other models normalization rules are applied for the Arabic vocabulary collected, for example letters "أ" , "إ" and "آ" are replaced with "ا" whereas the letter "ة" is re-placed with "ه" , and the letter "ى" is replaced with "ي ". These rules may cause confusion for the Arabic language because each letter's shape can change the meaning of the given word; for example: "سأل" means asked while "سال" means flowed. In our model, we do not have this problem. We have the word "سأل" and the word "سال" as different entities. For example, the estimated similarity scores of these two words with the word "أجاب" , which means answered, are − 0.826 and − 0.027 respectively. These generated

scores are reasonable since "سأل" and "أجاب" are antonyms while "سال" and "أجاب" are unrelated.
- An example of a pair of words where our model succeeds is the following. The pair "أصدقاء" and "أصحاب" are exact synonyms, as they both denote "friends". Our model indeed correctly gives it a high similarity score of 0.821. On the other hand, it was given low similarity scores of 0.268, 0.168, 0.264, 0.341 and 0.34 in the five models Polyglot, Altow., Bojan, ArW3 and ArW5 (Al-Rfou et al., 2013; Altowayan & Tao, 2016; Bojanowski et al., 2017; Fouad et al., 2020) respectively. Only AraVec models (Soliman et al., 2017) succeed, giving it high scores of 0.656 and 0.594.
- One idea that could be investigated in the future is how to combine different word vector methods. For example, AraVec also produces good results. One could possibly use AraVec's vectors as initial vectors for our algorithm, and run our algorithm starting these initial vectors. This is expected to improve upon the currently used randomly generated initial vectors.

A point that is worth mentioning is that the Arabic language uses diacritics. Two words can have the same letters but different diacritics and also different meanings. For example: "بَرّ" means land while "بِرّ" means righteousness. Our model's similarity score of the word "بر" with the word "فضيلة" which means virtue is 0.575 while it is 0.513 with the word "أرض" which means ground. The score is a bit higher for the first pair than the second one. These situations can cause some ambiguity, and misleading similarity scores for most word vector models, because of the lack of diacritization. A preprocessing step is needed to estimate the intended meaning from the context of the sentence.

Now, there could be a question. Is the polarity issue a problem only in the Arabic language embeddings, or it is a prevalent problem in the other major languages. To explore this, we have considered English language embedding models such as word2vec, Glove and fastText, and have performed a small experiment, whereby we tested a few antonym pairs and unrelated word pairs. Of course, one cannot exactly compare with the pairs in Table 2, because these are two different languages. But this experiment will give a general idea about the performance of English language embeddings for the polarity issue. Table 3 shows the results. One can observe that even these widely-used word vector embeddings are not differentiating well between pairs of antonyms and pairs of unrelated words. From Table 3, it can be noted that the percentage of words belonging to the antonyms category having a similarity score higher than or equal to 0.5 are 50%, 75% and 75% for word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2017) respectively. This imposes ambiguity in differentiating between synonyms and antonyms. While the remaining percent of the antonyms pairs have similarity scores less than 0.5, thus exhibiting also an ambiguity between these pairs and unrelated pairs, and making the point that these leading

Table 3 English word embeddings performance on some antonyms and unrelated words (EngSphere is our sphere-based approach applied to the English language)

word1	word2	word2vec (Mikolov et al., 2013)	GloVe (Pennington et al., 2014)	fastText (Mikolov et al., 2017)	EngSphere (Rizkallah et al., 2020b)
Antonyms					
right	error	0.132	0.406	0.38	− 0.586
strength	weakness	0.592	0.551	0.71	− 0.983
poor	good	0.46	0.659	0.629	− 0.677
sadness	joy	0.598	0.614	0.745	− 0.977
developed	undeveloped	0.247	0.238	0.441	− 0.838
long	short	0.577	0.821	0.754	− 0.944
smooth	rough	0.367	0.529	0.634	− 0.897
easy	difficult	0.589	0.671	0.731	− 0.973
Unrelated					
date	pleased	0.014	0.32	0.267	− 0.024
three	quality	0.026	0.48	0.368	0.247
dollar	day	0.221	0.446	0.452	0.425
doctor	part	0.046	0.418	0.342	0.119
fighting	old	0.126	0.392	0.396	− 0.015
joy	emperor	0.105	0.2127	0.33	0.014
family	book	0.143	0.499	0.402	0.449
followers	bag	0.039	0.146	0.302	0.126

methods did not manage to distinguish between antonyms and unrelated words. In Table 3, we also added the results obtained by our approach that we developed for the English language (Rizkallah et al., 2020b), it can be shown that the approach clearly identifies antonyms by assigning all the pairs negative similarity scores less than − 0.5 succeeding in differentiating between antonyms and synonyms as well as antonyms and unrelated pairs.

Furthermore, we evaluated the performance on antonyms-synonyms distinction dataset (Nguyen et al., 2016). This dataset contains word pairs labeled as either antonyms or synonyms. After refining the dataset to only include words that are present in the evaluated models, the total number of pairs are 1649 pairs, 825 antonyms and 824 synonyms. In our experiment, each pair is classified as antonym or synonym based on the dot product of the unit vectors of the corresponding words in the pair. If the dot product is greater than or equal 0.5, the pair is classified as synonym else the pair is classified as antonym. The models word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2017) score an accuracy of 55.73%, 60.28% and 64.89% respectively. An English model developed using our approach scores an accuracy of 78.90%.

5.2 Semantic textual similarity

Semantic Textual Similarity (STS) determines the degree of similarity between two pieces of text (Gomaa & Fahmy, 2013). SemEval-2017 Task 1 (SemEval, 2017) has included an Arabic dataset where the human judged similarity scores are given for 250 pairs of Arabic sentences and required to be compared with their corresponding machine-generated similarity scores. In order to assess the quality of our ArSphere vectors, we adopted a method that uses them for such task. The method works by getting the vector for each word in the sentence and computing a representative vector for the whole sentence by summing these vectors after weighing each vector by its corresponding TF-IDF (term frequency-inverse document frequency) and term frequency. TF-IDF weights are computed using a collection of Arabic corpora obtained from CNN, BCC and other sources known as "OSAC" (Saad & Ashour, 2010). The term frequency represents the number of times the term occurs in the sentence divided by the length of the sentence. After normalizing the representative vector for each sentence, dp: the dot product between the two vectors of the pair of sentences is computed. In addition, a length feature is defined as:

len_feat=1-abs(length of sentence1-length of sentence2)/(length of sentence1+length of sentence2)

The final machine-similarity score is evaluated as the weighted summation between dp and len_feat:

machine-similarity score=weight1*dp+weight2*len_feat

where weight1 and weight2 are chosen to be 0.7 and 0.3 respectively. The machine-generated scores are then compared with the given human-judged scores through Pearson correlation as required in the competition. We have also compared this task results with the other published Arabic embedding models through applying the same procedure explained but using their vectors. Moreover, we considered the AraBERT model (Baly et al., 2020) where a vector is computed for the whole sentence (known as CLS vector in BERT models). The dot product between the normalized vectors of each query sentence is computed to represent the similarity score between these sentences. Finally, we introduce the competition's summary results. Table 4 combines this task results. It can be noted that our model achieved the best result among the other Arabic embedding models.

5.3 Sentiment analysis

We applied Sentiment Analysis on "ASTD: Arabic sentiment tweets dataset" (Nabil et al., 2015). This dataset contains 10,006 Arabic tweets, each tweet is labeled as either positive, negative, objective or mixed. A hybrid method is developed combining word embeddings and dictionary/lexicon-based approaches (Alashri et al., 2019; Kumar & Babu, 2020; Moussa et al., 2020; Singh & Sachan, 2019; Taj et al., 2019). The method uses a small-sized dictionary of positive and negative Arabic words. This dictionary (pos-neg dict) is based on the work done in Salameh et al. (2015) and Mohammad et al. (2016) where Arabic positive and negative seeds are translated from 77 synonyms of positive and

Table 4 STS SemEval-2017 Task 1

Model	Pearson correlation
ArSphere (our approach)	0.655
AraVec SG-twitter (Soliman et al., 2017)	0.585[a]
Bojan. (Bojanowski et al., 2017)	0.570
AraVec CBOW-twitter (Soliman et al., 2017)	0.568[a]
Altow. (Altowayan & Tao, 2016)	0.568
Polyglot (Al-Rfou et al., 2013)	0.530
ArWordVec SG-3 (Fouad et al., 2020)	0.504
ArWordVec SG-5 (Fouad et al., 2020)	0.492
AraBERT (Baly et al., 2020)	0.415
Competition maximum	0.754
Competition minimum	0.003
Competition average	0.52
Competition median	0.596

[a]This score is reported in their paper

negative English words. To determine whether a word in the tweet is positive, negative or neutral, this is done through the vector of the word and computing its similarity scores with the vectors of the words in the pos-neg dict. Then, a tweet is classified as Positive, Negative or Objective based on the percentages of positive and negative words constituting the tweet. We have excluded the tweets in the Mixed class (i.e positive and negative in the same tweet). This is in order to compare with the ArWordVec (Fouad et al., 2020) method, since they applied their word vector embedding on this same data set and excluded the Mixed category too. Table 5 summarizes the results of our model and two of the best models of ArWordVec. One can observe that our proposed method outperforms its competitor.

5.4 Machine translation application

Word embedding is one of the approaches that have recently been successfully applied to the problem of machine translation (Artetxe et al., 2017; Xing et al., 2015; Zou et al., 2013). Most methods rely on the existence of large parallel corpora available for the source and target languages. These parallel corpora are difficult to find especially for low-resources languages. Moreover, in the Arabic language there exists a number of dialects e.g. Egyptian dialect, Levantine dialect, Gulf dialect and others. Therefore, translation between dialects is another challenge in the Arabic language (Diab et al., 2014). Our proposed approach allows to facilitate the task of machine translation. This is because there is no need for comprehensive parallel text corpora. Instead, only the training data as described (pairs of synonyms, antonyms and unrelated words) need to be obtained in both languages or dialects. In our approach we use a complete lexicon for dialect 1 to design the word embedding sphere 1. We use another complete lexicon for dialect 2 to design its word embedding sphere 2. The lexicon of each dialect consists of complete sets of synonyms, antonyms, and unrelated words for this particular dialect. For Arabic dialects, there exists a number of works that constructed these dialect lexica (Boujelbane et al., 2013; Diab et al., 2014; Rizkallah et al., 2018), with all their synonyms and antonyms. The construction of each embedding sphere follows the approach we presented above in this paper. So far in this previous step the two embedding spheres of both dialects are unrelated. In the next step we align the two spheres with respect to each other such that the vectors corresponding to dialect 1 words coincide (or be close to) the vectors of equivalent words of the other dialect. This means that word 1 from sphere 1 is placed in the same position as word 2 from sphere 2 if word 1 (of dialect 1) has the same meaning as word 2 (of dialect 2). This can be accomplished by augmenting the training sets of both dialects by cross-dialect synonyms. We do not need an extensive cross-dialect dictionary to construct this training set, as the extensive within-dialect synonyms and antonyms lists will have considerable contributions in the placement of the vector locations. This is the advantage of this proposed approach, as comprehensive cross-dialect dictionaries are not available for many dialect pairs.

Once trained, a query is given for a word in dialect 1. The vector of this word is retrieved and the vectors of the other dialect's sphere that are close to this vector will be the translated meanings. Antonyms can also be obtained by obtaining the vectors of sphere 2 that are polar opposites to the given vector. Consider Egyptian and Levantine dialect translation as an example, Fig. 5 shows an illustration of the parallel spheres. For example the word pairs ("كويس" and "منيح") both mean "good", the first in Egyptian dialect while the second in Levantine dialect. The vectors of this pair are nearly at the same location each in its sphere, leading to a dot product close to 1. The same applies for the word pairs ("تعمل" and "تسوي"), ("أبداً" and "منوب") and ("وحش" and "عاطل"). Consider the word "كويس" (good) in Egyptian dialect, the antonym of this word can be easily obtained in the Levantine

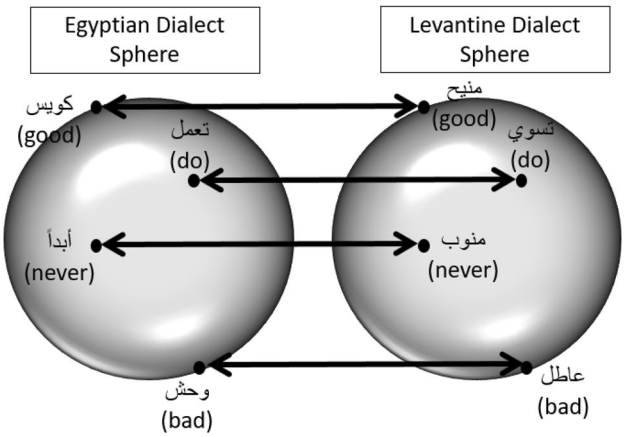

Fig. 5 Parallel spheres

Table 5 ASTD sentiment analysis

Model	Precision	Recall	Weighted F1-Score	Accuracy
ArSphere (our approach)	66	73	62	72.95
ArWordVec-SG3 (Fouad et al., 2020)	51.44	45.25	43.57	69.28
ArWordVec-SG5 (Fouad et al., 2020)	52.45	47.18	45.95	69.93

dialect which is "عاطل" (bad) as the vectors of these words lie at opposite poles each in its sphere having dot product close to − 1. To conclude, any number of parallel spheres can be learned, each corresponding to a certain dialect or language. Corresponding words (or words of similar meaning) in different dialects or languages can be obtained by simply computing the dot products between vectors of words.

6 Conclusion

In this work we propose a new word vector embedding approach for the Arabic language. The purpose of this approach is to tackle the issue of polarity. Antonyms should be clearly distinguished from unrelated word pairs. This would help further the utilization of word embedding methods in applications, such as sentiment analysis and text understanding. The problem with existing approaches is that they are not very successful in this issue. With an unrestricted space for the embedding, existing approaches have a hard time pinpointing suitable positions of the vectors to express antonyms and unrelated words in a different manner. We propose here the sphere as a suitable space for the embedding, with vectors on opposite poles signifying antonym relation. The proposed approach is compared with other Arabic language embedding approaches, and the results appear more sensible. Antonyms are clearly distinguished in most cases from unrelated words, while this is not the case for the competing approaches. Also, synonyms are captured well, compared to the other approaches. The other advantage of the proposed method is that it is a semi-supervised approach. This allows us to benefit from the existing lexicons produced by true linguistic experts. Moreover, it gives the designer control over the features of the embedding. For example, one can design a specialized embedding for the purpose of opinion mining, which focuses on the specifics of the product being reviewed. Also, in some cases the antonym/synonym relations depend on the context. For example "black" and "white" can be considered opposites. In other contexts they can be considered as similar, because they are both colors. The supervised feature allows us to engineer the embedding according to the context and the application. However, word vector methods that are unsupervised have some other advantages. For example, this allows training on any text without the need of expert supervision, i.e. without the need for constructing synonym/antonym tables. However, it is possible to add this unsupervised element as a (secondary) part of our method, in order to enrich the corpus.

In summary, in this work we have developed an approach for Arabic word embedding that utilizes the sphere as an embedding space, and as such is able to distinguish between synonyms, antonyms, and unrelated words. The proposed method uses some supervised aspects, and relies on word resources developed by expert linguists. We therefore believe that the proposed work has merit, and will provide additional advancement for Arabic word embeddings.

References

Al-Ayyoub, M., Essa, S. B., & Alsmadi, I. (2015). Lexiconbased sentiment analysis of arabic tweets. *IJSNM, 2*(2), 101–114.

Al-Azani, S., & El-Alfy, E. S. M. (2017a). *Hybrid deep learning for sentiment polarity determination of arabic microblogs* (pp. 491–500). New York: Springer.

Al-Azani, S., & El-Alfy, E. S. M. (2017b). Using word embedding and ensemble learning for highly imbalanced data sentiment analysis in short arabic text. *Procedia Computer Science, 109*, 359–366.

Al-Rfou, R., Perozzi, B., & Skiena, S. (2013). Polyglot: Distributed word representations for multilingual nlp. arXiv preprint arXiv: 13071662.

Alashri, S., Alzahrani, S., Alhoshan, M., Alkhanen, I., Alghunaim, S., & Alhassoun, M. (2019). Lexi-augmenter: Lexicon-based model for tweets sentiment analysis. In *2019 IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC)*, IEEE (pp. 7–10).

Altowayan, A. A., & Elnagar, A. (2017). Improving arabic sentiment analysis with sentiment-specific embeddings. In *2017 IEEE international conference on big data (big data)*, IEEE (pp. 4314–4320).

Altowayan, A. A., & Tao, L. (2016). Word embeddings for arabic sentiment analysis. In *2016 IEEE international conference on big data (big data)*, IEEE (pp. 3820–3825).

Aly, M., & Atiya, A. (2013). Labr: A large scale arabic book reviews dataset. In *Proceedings of the 51st annual meeting of the association for computational linguistics* (Volume 2: Short Papers) (Vol. 2, pp. 494–498).

Artetxe, M., Labaka, G., & Agirre, E. (2017). Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th annual meeting of the association for computational linguistics* (Volume 1: Long Papers, pp. 451–462).

Baly, F., Hajj, H., et al. (2020). Arabert: Transformerbased model for arabic language understanding. In *Proceedings of the 4th workshop on open-source arabic corpora and processing tools, with a shared task on offensive language detection* (pp. 9–15).

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

Boudad, N., Ezzahid, S., Faizi, R., & Thami, R.O.H. (2020). Exploring the use of word embedding and deep learning in arabic sentiment analysis. In M. Ezziyyani (Ed.), *Advanced intelligent systems for sustainable development (AI2SD'2019)*, Springer, Cham (pp. 243–253).

Boujelbane, R., Khemekhem, M. E., & Belguith, L. H. (2013). Mapping rules for building a tunisian dialect lexicon and generating corpora. In *Proceedings of the sixth international joint conference on natural language processing* (pp. 419–428).

Dahou, A., Xiong, S., Zhou, J., Haddoud, M. H., & Duan, P. (2016). Word embeddings and convolutional neural network for arabic sentiment classification. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 2418–2427).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805.

Diab, M., Al-Badrashiny, M., Aminian, M., Attia, M., Dasigi, P., Elfardy, H., Eskander, R., Habash, N., Hawwari, A., & Salloum, W. (2014). Tharwa: A large scale dialectal arabic-standard arabic-english lexicon. In *9th international conference on language resources and evaluation, LREC 2014*, European Language Resources Association (ELRA) (pp. 3782–3789).

Dou, Z., Wei, W., & Wan, X. (2018). Improving word embeddings for antonym detection using thesauri and sentiwordnet. In *CCF international conference on natural language processing and Chinese computing*. Springer (pp. 67–79).

El Bazi, I., & Laachfoubi, N. (2019). Arabic named entity recognition using deep learning approach. *International Journal of Electrical & Computer Engineering, 9*(3), 2088–8708.

El-Beltagy, S. R., & Ali, A. (2013). Open issues in the sentiment analysis of arabic social media: A case study. In *2013 9th international conference on innovations in information technology* (IIT), IEEE (pp. 215–220)

El-Beltagy, S. R., Khalil, T., Halaby, A., & Hammad, M. (2016). Combining lexical features and a supervised learning approach for arabic sentiment analysis. In *International conference on intelligent text processing and computational linguistics*, Springer (pp. 307–319).

Fouad, M. M., Mahany, A., Aljohani, N., Abbasi, R. A., & Hassan, S. U. (2020). Arwordvec: Efficient word embedding models for arabic tweets. *Soft Computing, 24*(11), 8061–8068.

Ghoniem, R. M., Alhelwa, N., & Shaalan, K. (2019). A novel hybrid genetic-whale optimization model for ontology learning from arabic text. *Algorithms, 12*(9), 182.

Gomaa, W. H., & Fahmy, A. A. (2014). Automatic scoring for answers to arabic test questions. *Computer Speech & Language, 28*(4), 833–857.

Gomaa, W. H., Fahmy, A. A., et al. (2013). A survey of text similarity approaches. *International Journal of Computer Applications, 68*(13), 13–18.

Habibi, M., Weber, L., Neves, M., Wiegandt, D. L., & Leser, U. (2017). Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics, 33*(14), i37–i48.

Hammo, B., Abuleil, S., Lytinen, S., & Evens, M. (2004). Experimenting with a question answering system for the arabic language. *Computers and the Humanities, 38*(4), 397–415.

Hasanzadeh, S., Fakhrahmad, S., & Taheri, M. (2020). Based recommender systems: A proposed rating prediction scheme using word embedding representation of reviews. *The Computer Journal*. https://doi.org/10.1093/comjnl/bxaa044

Helwe, C., & Elbassuoni, S. (2019). Arabic named entity recognition via deep co-learning. *Artificial Intelligence Review, 52*(1), 197–215.

Kolyvakis, P., Kalousis, A., & Kiritsis, D. (2018). Deepalignment: Unsupervised ontology matching with refined word vectors. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: Human language technologies*, Volume 1 (Long Papers) (pp. 787–798).

KS. (2018). Alkhawarizmy software. https://eg.linkedin.com/company/alkhawarizmy-software?trk=public_profile_experience-item_result-card_subtitle-click.

Kumar, C. S. P., & Babu, L. D. D. (2020). Evolving dictionary based sentiment scoring framework for patient authored text. *Evolutionary Intelligence* 1–11.

Lachraf, R., Echahid, Y., Lakhdar, H., Abdelali, A., Schwab, D., et al. (2019). Arbengvec: Arabic-english crosslingual word embedding model. In *Proceedings of the fourth Arabic natural language processing workshop.*

Mahgoub, H. E., Hashish, M., & Hassanein, A. T. (1990). A matrix representation of the inflectional forms of arabic words: A study of co-occurrence patterns. In *Proceedings of the 13th conference on computational linguistics*-Volume 3, Association for Computational Linguistics (pp. 419–421).

Malhas, R., Torki, M., & Elsayed, T. (2016). Qu-ir at semeval 2016 task 3: Learning to rank on arabic community question answering forums with word embedding. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)* (pp. 866–871).

Medved, M., & Hor´ak, A. (2018). Sentence and word embedding employed in open question-answering. In *ICAART* (2) (pp. 486–492).

Mezghanni, I. B., & Gargouri, F. (2017). Deriving ontological semantic relations between arabic compound nouns concepts. *Journal of King Saud University-Computer and Information Sciences, 29*(2), 212–228.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). Advances in pre-training distributed word representations. arXiv preprint arXiv:171209405

Mohammad, S. M., Salameh, M., & Kiritchenko, S. (2016). How translation alters sentiment. *Journal of Artificial Intelligence Research, 55*, 95–130.

Moussa, M. E., Mohamed, E. H., & Haggag, M. H. (2020). A generic lexicon-based framework for sentiment analysis. *International Journal of Computers and Applications, 42*(5), 463–473.

Nabil, M., Aly, M., & Atiya A. (2015). Astd: Arabic sentiment tweets dataset. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 2515–2519).

Nakov, P., Màrquez, L., Moschitti, A., & Mubarak, H. (2019). Arabic community question answering. *Natural Language Engineering, 25*(1), 5.

Nguyen, K. A., im Walde, S. S., & Vu, N. T. (2016). Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th annual meeting of the association for computational linguistics* (Volume 2: Short Papers) (pp. 454–459).

Omar, A. M. (2008). *Modern Arabic language dictionary* (معجم اللغة العربية المعاصرة). Alam El-Kutub.

Ono, M., Miwa M., & Sasaki, Y. (2015). Word embeddingbased antonym detection using thesauri and distributional information. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 984–989).

Pennington J., Socher R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Rizkallah, S., Atiya, A., Mahgoub, H. E., & Heragy, M. (2018). Dialect versus msa sentiment analysis. In *International conference on advanced machine learning technologies and applications*, Springer (pp. 605–613).

Rizkallah, S., Atiya, A. F., & Shaheen, S. (2020a). Learning spherical word vectors for opinion mining and applying on hotel reviews. In *International conference on intelligent systems design and applications*, Springer (pp. 200–211).

Rizkallah, S., Atiya, A. F., & Shaheen, S. (2020b). A polarity capturing sphere for word to vector representation. *Applied Sciences, 10*(12), 4386.

Rizkallah, S., Atiya, A. F., & Shaheen, S. (2021). New vectorspace embeddings for recommender systems. *Applied Sciences, 11*(14), 6477.

Rizkallah, S., Atiya, A., & Shaheen, S. (2022). Arcoq: Arabic closest opposite questions dataset. Working Paper

Saad, M. K., & Ashour, W. (2010). Osac: Open source arabic corpus. In *Proceedings of the 6th international symposium on electrical and electronics engineering and computer science* (pp. 557–562).

Salama, R. A., Youssef, A., & Fahmy, A. (2018). Morphological word embedding for arabic. *Procedia Computer Science, 142*, 83–93.

Salameh, M., Mohammad, S., & Kiritchenko, S. (2015). Sentiment after translation: A case-study on arabic social media posts. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 767–777).

SemEval. (2017). Semeval-2017 task 1. http://alt.qcri.org/semeval2017/task1/.

Shaalan, K. (2014). A survey of arabic named entity recognition and classification. *Computational Linguistics, 40*(2), 469–510.

Shen, Y., Rong, W., Jiang, N., Peng, B., Tang, J., & Xiong, Z. (2017). Word embedding based correlation model for question/answer matching. In *Thirty-first AAAI conference on artificial intelligence.*

Singh, S. K., & Sachan, M. K. (2019). Sentiverb system: Classification of social media text using sentiment analysis. *Multimedia Tools and Applications, 78*(22), 32109–32136.

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

Taj, S., Shaikh, B. B., & Meghji, A. F. (2019). Sentiment analysis of news articles: A lexicon based approach. In *2019 2nd international conference on computing, mathematics and engineering technologies* (*iCoMET*), IEEE (pp. 1–5).

Talafha, B., Ali, M., Za'ter, M. E., Seelawi, H., Tuffaha, I., Samir, M., Farhan, W., & Al-Natsheh, H. T. (2020). Multidialect arabic bert for country-level dialect identification. arXiv preprint arXiv:200705612.

Tubishat, M., Idris, N., & Abushariah, M. A. (2018). Implicit aspect extraction in sentiment analysis: Review, taxonomy, oppportunities, and open challenges. *Information Processing & Management, 54*(4), 545–563.

Vasile, F., Smirnova, E., & Conneau, A. (2016). Metaprod2vec: Product embeddings using sideinformation for recommendation. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 225–232).

Wu, Y., Xu, J., Jiang, M., Zhang, Y., & Xu, H. (2015). A study of neural word embeddings for named entity recognition in clinical text. In *AMIA annual symposium proceedings*, American Medical Informatics Association (Vol. 2015, p. 1326).

Xing, C., Wang, D., Liu, C., & Lin, Y. (2015). Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 1006–1011).

Ye, Z., Li, F., & Baldwin, T. (2018). Encoding sentiment information into word vectors for sentiment analysis. In *Proceedings of the 27th international conference on computational linguistics* (pp. 997–1007).

Yih, W., Zweig, G., & Platt, J. C. (2012). Polarity inducing latent semantic analysis. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, Association for Computational Linguistics (pp. 1212–1222).

Zahran, M. A., Magooda, A., Mahgoub, A. Y., Raafat, H., Rashwan, M., & Atyia, A. (2015). Word representations in vector space and their applications for arabic. In *International conference on intelligent text processing and computational linguistics*, Springer (pp. 430–443).

Zhang, J., Salwen, J., Glass, M., & Gliozzo, A. (2014). Word semantic representations using Bayesian probabilistic tensor factorization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1522–1531).

Zitouni, I. (2014). *Natural language processing of semitic languages*. Springer.

Zou, W. Y., Socher, R., Cer, D., & Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1393–1398)

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com