AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP

Article *in* Procedia Computer Science · November 2017

DOI: 10.1016/j.procs.2017.10.117

3 authors:

Abu Bakr Soliman Mohammad
Nile University
3 PUBLICATIONS 200 CITATIONS
SEE PROFILE

Kareem Eissa
Nile University
3 PUBLICATIONS 182 CITATIONS
SEE PROFILE

Samhaa R. El-Beltagy
Newgiza University
116 PUBLICATIONS 1,554 CITATIONS
SEE PROFILE

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2017) 000–000

Procedia
Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Arabic Computational Linguistics, ACLing 2017, 5-6 November 2017, Dubai, United Arab Emirates

AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP

Abu Bakr Soliman, Kareem Eissa, Samhaa R. El-Beltagy1

Center for Informatics Science, Nile University, Giza 12588, Egypt

Abstract

Advancements in neural networks have led to developments in fields like computer vision, speech recognition and natural language processing (NLP). One of the most influential recent developments in NLP is the use of word embeddings, where words are represented as vectors in a continuous space, capturing many syntactic and semantic relations among them. AraVec is a pre-trained distributed word representation (word embedding) open source project which aims to provide the Arabic NLP research community with free to use and powerful word embedding models. The first version of AraVec provides six different word embedding models built on top of three different Arabic content domains; Tweets, World Wide Web pages and Wikipedia Arabic articles. The total number of tokens used to build the models amounts to more than 3,300,000,000. This paper describes the resources used for building the models, the employed data cleaning techniques, the carried out preprocessing step, as well as the details of the employed word embedding creation techniques.

Keywords: Word2Vec; Word Embeddings; Arabic; NLP

1. Introduction

In the last few years, the benefits of using distributed word representations (embeddings) have been illustrated and highlighted in many different NLP tasks including but not limited to, sentiment analysis Tang et al. [1], named entity recognition Sien [2], and part of speech tagging Lin et al. [3] While these benefits were accompanied by the provision

* Abu Bakr Soliman. Tel.: +0-202-3854-1760.
E-mail address: ab.soliman@nu.edu.eg

of several open source word representation models in English, the same cannot be said to be true with respect to Arabic. AraVec is a distributed word representation (word embeddings) open source project which aims to provide the Arabic NLP research community with free to use, powerful word embedding models. The presented models[2] were built carefully using multiple different Arabic text resources to provide wide domain coverage. Specifically, the models were built using web pages collected from World Wide Web, text harvested from social platforms and text obtained from encyclopedia entries.

The work presented in this paper describes the various steps followed for the creation of these models. Data collection, data cleaning and preprocessing, and hyper-parameter tuning are among the steps described.

The rest of this paper is organized as follows: in section 2 we review related work, in Section 3 we describe the data sources and our preprocessing techniques, in section 4 we describe the used word embedding techniques and how we used those to build our models, in Section 5 we use qualitative and quantitative measures to evaluate our models and finally, in Section 6 we conclude this paper and discuss future work.

2. Related Work

Providing distributed word representations for words using large datasets contributes to improved performance across many NLP tasks as demonstrated by Ma and Hovy [4]. The construction of distributed word representations in Arabic has been carried out by several researchers. For example, the work of Al-Rfou et al. [5] has resulted in the creation of word embedding models for 117 different languages including Arabic. Soricut and Och [6] provided an unsupervised, language agnostic technique for inducing morphological transformations between words. They tried to discover a wide set of morphological rules for building morphological analyzers. They evaluated their technique across six different languages including Arabic. Their Arabic model was built using the Arabic GigaWord corpus from Parker, Robert et al. [7]

Several researchers have used Arabic word representations as features for common NLP tasks. Zirikly and Diab [8] for example, explored the impact of using word embeddings in detecting named entities in Arabic. In their work, the authors demonstrated that this novel representation scheme can replace the use of dictionaries and gazetteers and still result in better performance despite the fact that the authors used a small Twitter corpus of only 3,646 tweets in the form of the Dialectal Arabic Dataset (DA-EGY) [9] which contains just about 40K tokens targeting the Egyptian dialect. Zahran et al. [10] collected a large Modern Standard Arabic dataset from Wikipedia, Arabic Gigaword [7], some Arabic Newswires as well as from other resources. They compared different techniques to build vectorized space representations for Arabic and evaluated them using the standard word similarity task in Arabic. They also evaluated those on two tasks: Information Retrieval and Short Answer Grading.

3. AraVec Data Collection and Preparation

3.1. Data Collection

The main goal of this work is to provide efficient distributed word representation models for different NLP tasks across different text domains. Towards this end, we have decided to collect data for building the various distributed word representation models from three completely different data sources which are: Twitter, the World Wide Web, and Wikipedia. The justification for collecting this data as well as the steps taken to collect data from each of these resources is detailed in the following subsections.

3.1.1. Twitter

Many recent NLP researchers targeting social media analysis and applications have used Twitter as a main data resource for carrying out their work. This could be attributed to a number of reasons including the fact that Twitter can be easily queried within specific time spans. Other factors are related to the fact that tweets are often tagged with

[2] All models are downloadable from the following link: https://github.com/bakrianoo/aravec

geo locations, have user information and that many tools exist for carrying out topic and sentiment derivation from them.

When addressing Arabic within the context of social media, we could claim that we are facing many different linguistic domains; not just one. This is largely due to that fact that social media users express themselves using a variety of dialects and sub-dialects. Examples include: Modern Standard Arabic (MSA فصحى), Egyptian or more specifically Cairene, Gulf, Moroccan, Tunisian, Algerian, and Levantine dialects. Capturing as many of these linguistics domains within the context of daily social interactions was one of the main motivations for creating word representation models using tweets.

To gain access to a wider range of tweets than those provided by the standard twitter API, we designed a crawler to collect tweets. The crawler fully complies with the rules[3] which specify what can be queried from Twitter. The crawler was used to query twitter based on two main parameters: (1) the language which was set to be Arabic and (2) the time span across which to conduct the search. More than 2100 search queries were used to collect our final twitter dataset. At the end we collected more than 77,600,000 Arabic tweets posted between 2008 and 2016 obtained from different random geo locations.

3.1.2. World Wide Web

Arabic is the fifth most used language in the world with more than 420 million native speakers[4]. More than 41% Arabic speakers use the internet[5] constituting about 4% of the Internet context by the end of 2016[6]. Public websites found in the World Wide Web, cover a wide spectrum of topics spread across news sites, blogs, service sites, social forums, among many others. When selecting a model for collecting data from these many various sites, we had to make sure that the privacy policy for each, is respected. To ensure that, we made use of the Common Crawl project. Common Crawl[7] is a nonprofit organization that maintains an open repository of web crawl data. The web crawl data encompasses 40+ languages which include Arabic. In our model, we have used a subset of the January 2017 crawl dump. The dump contains more than 3.14 billion web pages and about 250 Terabytes of uncompressed content. The Common Crawl project provides crawled data in three different formats:

- WARC files which are raw crawl data
- WAT files which are computed metadata for the data stored in WARC
- WET files which are the extracted plain text from the data stored in WARC

We used WET files as we were only interested in plain text for building the distributed word representation models. Due to the size of the dump, which requires massive processing power and time for handling, we only used 30% of the data contained in it. As this subset comprises about one billion web pages (written in multiple language), we believed that it was large enough to provide sufficient Arabic Web pages from which we can build a representative word embeddings model. Here it is important to note that the Common Crawl project does not provide any technique for identifying or selecting the language of web pages to download. So, we had to download data first, and then discard pages that were not written in Arabic. The Arabic detection phase was performed using some regex commands and some NLP techniques to distinguish Arabic from other languages. After the completion of this phase we succeeded in obtaining 4,379,697 Arabic web pages which were then segmented into more than 180,000,000 paragraphs/documents for building our models.

3.1.3. Wikipedia

Wikipedia[8] is an encyclopedia which has been collaboratively written by users all over the world. As a resource, it provides more than 45M categorized articles targeting 285 languages including Arabic. Arabic was the first Semitic language to exceed 100,000 articles in Wikipedia. The Arabic portion of Wikipedia now has more than 520,000

[3] The rule can be found at: https://twitter.com/robots.txt
[4] https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers
[5] http://www.internetworldstats.com/stats19.htm
[6] http://www.internetworldstats.com/stats5.htm
[7] http://commoncrawl.org
[8] https://www.wikipedia.org

articles[9]. As an open source project licensed under the Creative Commons Attribution-ShareAlike 3.0 License (CC-BY-SA), Wikipedia provides free copies of its databases to be used in for a variety of purposes. To build our models from Wikipedia, we downloaded the Arabic dump dated January 2017. After segmenting the articles into paragraphs, we ended up with 1,800,000 paragraphs, each representing a document to be used for building our model.

3.2. Preprocessing

Text preprocessing is an important step in building any word embedding model as it has the potential to significantly affect the final results. In the following sub-sections, the main employed preprocessing steps are described.

3.2.1. Filtering Non-Arabic Content

The first pre-processing step that was carried out on the collected data was filtering out non-Arabic content. This is particularly important when dealing with data from the "Web" or "Twitter". Although Arabic can be easily recognized using its alphabet ("أ,ب,ت,.."), there are other languages whose alphabet contains characters that overlap with the Arabic alphabet such as Urdu and Persian. The challenge was to detect Arabic text only and filter out anything else. To do so, we used a Python library for language detection[10] which can detect 55 languages including Arabic, Persian and Urdu. The library provides an estimate of the confidence in the language. We used regular expressions to filter out languages that use other alphabets. Still in many cases the library would classify a paragraph into multiple languages for example 60% Arabic and 40% Persian. In cases like this, pages with mostly non-Arabic content were discarded.

3.2.2. Normalization

Normalization of Arabic characters is a common preprocessing step when dealing with Arabic text. In this step, letters "إ" ,"أ" and "آ" are replaced with "ا" while the letter "ة" is re-placed with "ه", and the letter "ى" is replaced with "ي". Diacritics are also removed in this step. In this step we also normalize mentions, URLs, emojis and emoticons. Mentions are normalized by replacing their text with a single non-standard Arabic word such as "حسابشخصي". URLs are normalized in a similar manner (the term used to replace a URL is "رابطويب"). Similarly, positive emojis or emoticons are replaced with the term "ايموشنموجب" while negative emojis/ emoticons are replaced with "ايموشنسالب". In addition, elongated words are converted back to their original form (example "سلاااام" will be converted to "سلام").

3.2.3. X-Rated Content Filtering:

When carrying out simple analysis on a sample of the Arabic World Wide Web dataset, we discovered that a large portion of this is dominated by X-Rated content. By "X-Rated" content we mean text content from adult websites. While it may be desirable to have coverage for this content for its semantic value, we did not want this kind of content to dominate our dataset as this would results in a dataset that is skewed towards this kind of content.

To address this problem, we built a simple word embeddings model using the this very same dataset. Then, by using a seed of some common X-Rated words, we utilized the power of word embeddings, to get the most similar terms to those. After revising the list to guarantee that there's no overlap between those and other commonly used words, we got a list of 85 words that could help classify X-Rated content. This list can be found in a separate file within the project's repository. We have utilized this list to detect the percentage of the occurrence of the X-Rated words in a paragraph. To filter out paragraphs, we calculate the percentage of X-Rated words contained within them. If this percentage is greater that some value α, we discard that paragraph. In our system, and after some experimentation, we have set α to 20%

[9] https://en.wikipedia.org/wiki/List_of_Wikipedias
[10] https://pypi.python.org/pypi/langdetect?

4. Word Embeddings

4.1. Techniques

Vector space models (VSMs) are one of the oldest and most well-known schemes for text representation. Traditionally, the vector space model has been primarily used for document representation, with more recent works extending this model to word or term representation. In such recent work, words are represented in continuous space where semantically similar words have a high similarity measure in that space. VSMs rely on the 'Distributional Hypothesis'[11], which states that words that occur in the same contexts tend to have similar meanings. The two main approaches to build these representations are: count-based approaches and predictive methods. Count-based approaches calculate co-occurrence statistics between words then map these statistics into a dense vector for each word. Predictive methods try to predict a word from its neighbors in terms of a learned dense vector for each word. The term "word embeddings" was first coined by Bengio et al. [11] . The model they proposed was based on the idea of obtaining the values for word vectors or embeddings by training a neural language model. By 2008, Collobert and Weston [12] demonstrated word embeddings as an effective tool in many downstream tasks. It was Mikolov et al. [13] who brought the idea to the forefront of research and contributed to its widespread use through the creation of the Word2Vec toolkit which can be easily be used and tuned to generate embeddings. Mikolov et al. [13], proposed two different model architectures for representing words in a multidimensional vector space namely the continuous bag-of-words (CBOW) model and the skip-gram model. The CBOW model aims to learn the embeddings by predicting the center word in a context given the other words in the context without regard to their order in the sentence. The Skip-Gram model is the reverse of the CBOW as it aims to predict the surrounding context words given the center word.

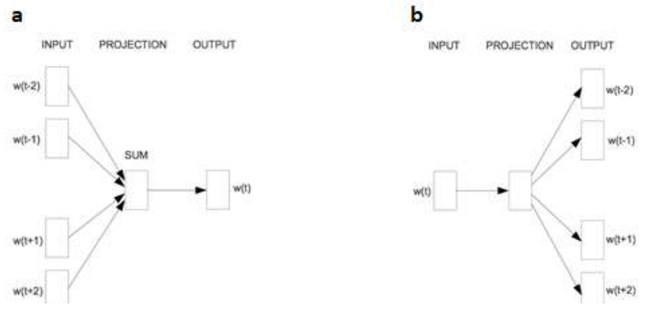


Figure 1 (a) CBOW, (b) Skip-gram

4.2. Building the models

The models that we have constructed were built using the Gensim[12] tool developed by Radim Rehurek [14], which is an efficient toolkit created for addressing many common NLP tasks and which includes an implementation for the Word2Vec model.

AraVec provides six different word embedding models, where each text domain (Tweets, WWW and Wikipedia) has two different models; one built using CBOW technique and the other using the Skip-Gram technique. To build these models, we ran a large set of experiments to tune the hyperparameters (minimum count, window size, . For the minimum word count, we noticed that a low threshold e.g. 5 results in increasing the size of the vocabulary without adding that much value to the model. We have thus set this value to 20 for the Wikipedia corpus. However, we noticed that there is a common problem faced for both the Twitter and Common Crawl dataset, which is the occurrence of misspelled words at a high frequency due to the size of the corpora. To overcome this problem, we set the minimum

[11] https://en.wikipedia.org/wiki/Distributional_semantics#Distributional_hypothesis
[12] https://radimrehurek.com/gensim/about.html

count value to 500 for both datasets. With respect to the window size we used a small window size of 3 for Twitter as the maximum length of a tweet is 140 characters. We increased this to 5 for the Wikipedia and Common Crawl datasets as paragraphs in both are much longer. All generated models had a vector dimension of 300.
The Wikipedia model took 10 hours to train on a Quad core Intel i7-3770 @3.4 GHz PC with 32 GB of RAM running Ubuntu 16.04, while the Twitter model took 1.5 days to train and the Common Crawl 4 days. Table 1, shows hyper-parameters used for each model as and the used corpus size.

Table 1 Model Configurations

Model Name	# Documents (Millions)	# Tokens (Millions)	Min Word Freq. Count	Window size	Technique
Twt-CBOW	66.9	1090	500	3	CBOW
Twt-SG					Skip-gram
WWW-CBOW	132.7	2225.3	500	5	CBOW
WWW-SG					Skip-gram
Wiki-CBOW	1.8	78.9	20	5	CBOW
Wiki-SG					Skip-gram

5. Evaluation

For evaluating the generated models, we have used both qualitative and quantitative methods each of which is presented in the following subsections.

5.1. Qualitative Evaluation

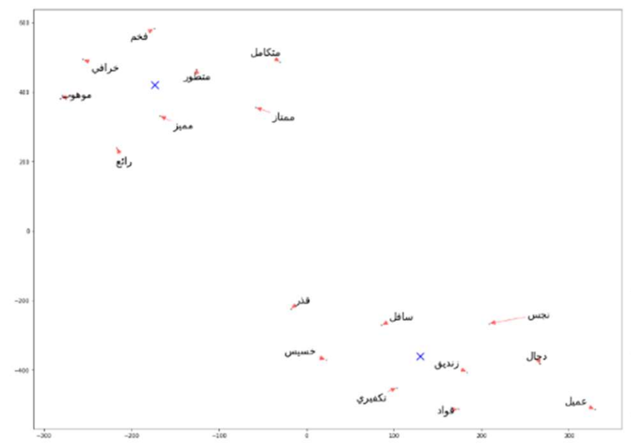
The purpose of carrying out qualitative evaluation for our models was to examine how well they capture similarities among words. To do, we used word vectors for a very small subset of sentiment words and applied a clustering algorithm to see whether words of the same polarity cluster together or not. We did the same thing with a set of randomly selected named entities of known types. Each of these tasks is explained in further details in the following subsections.

5.1.1. Clustering of Sentiment words

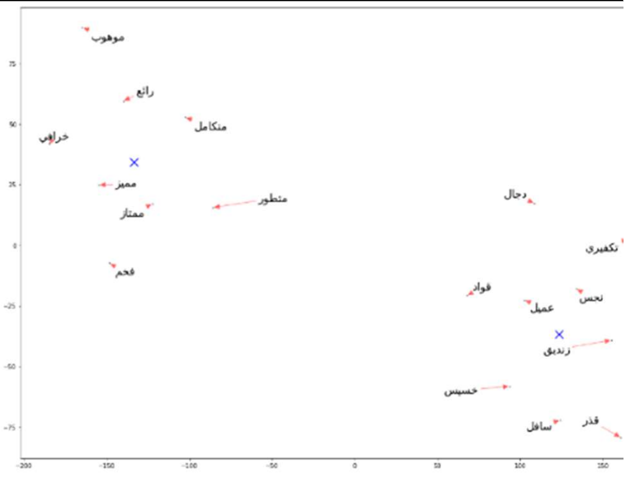
As stated above, we have selected a small subset of words taken from a sentiment lexicon. The subset used can be found in Table 2. We obtained the vectors representing each of those words from each of our generated models and ran the K-Means clustering algorithm on those (setting k=2). We then used the t-Distributed Stochastic Neighbour Embedding (t-SNE) tool by Ulyanov [15], to visualize and examine the results in a 2D graph. The results are shown in Figure 2. Examining these results, it is easy to see that in most cases, words of the same polarity have been clustered together. We note that the Wikipedia model produces the worst clusters. This can be explained by the factual nature of Wikipedia where many words that carry a negative or positive meaning in spoken language, are actually neutral in Wikipedia (ex.عميل).

Table 2 Set of sentiment words

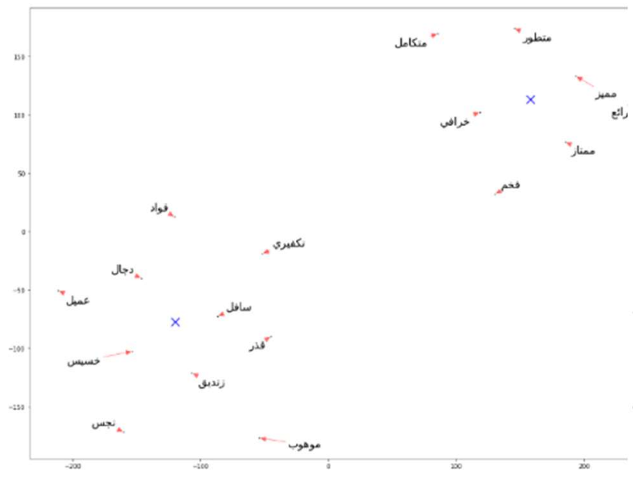
Positive	فخم	خرافي	موهوب	متطور	مميز	رائع	متكامل	
Negative	قذر	خسيس	تكفيري	سافل	زنديق	قواد	نجس	عميل



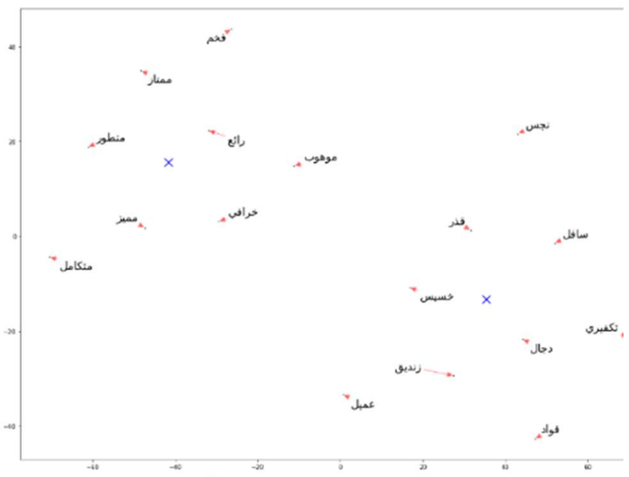
Twitter CBOW Model



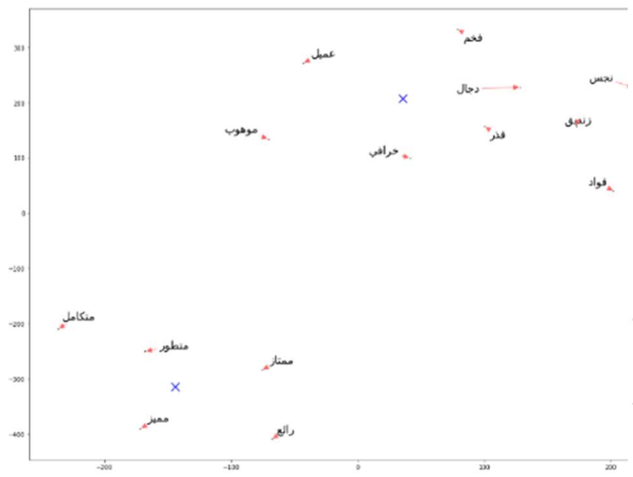
Twitter Skip-Gram Model



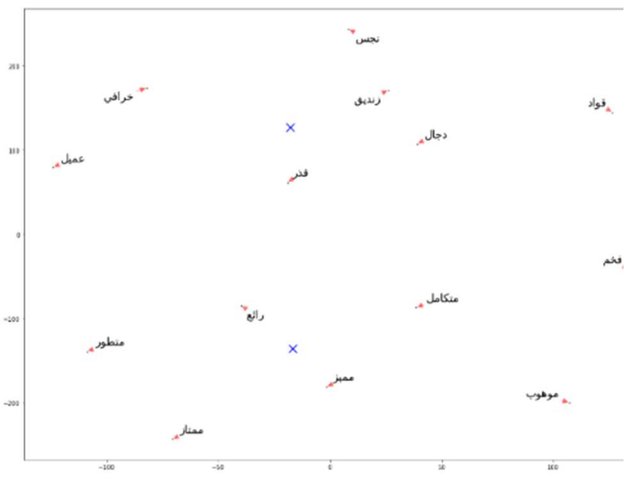
WWW CBOW Model



WWW Skip-Gram Model



Wikipedia CBOW Model



Wikipedia Skip-Gram Model

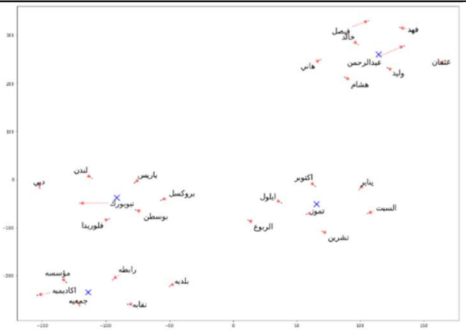
Figure 2: Results of Clustering Sentiment Words

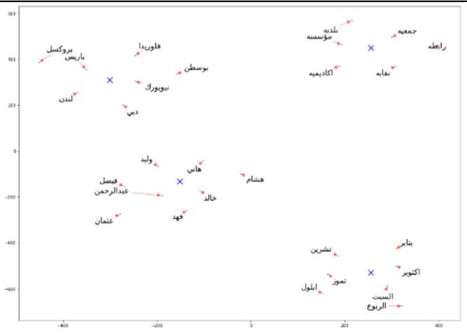
5.1.2. Clustering of Named Entities

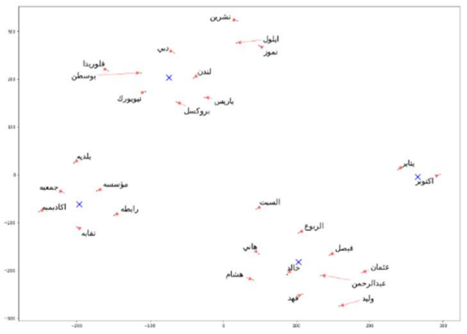
Similar to what we have done when clustering sentiment words, here we have also selected a random set of named entities each of which falls under one of four main categories: Person, Location, Organization and Time/Date as shown in Table 4. The K-Means clustering algorithm was applied on the vectors of these words with k set to 4. Examining the results in Figure 3 we can see that the models capture the similarities amongst Named Entities to a great extent.

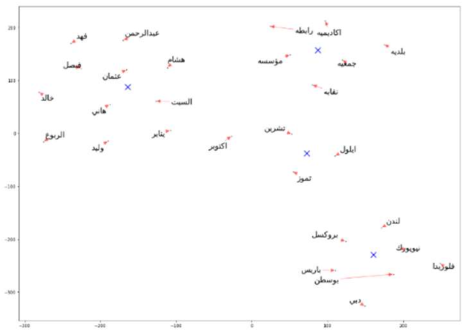
Table 3 Set of named entities

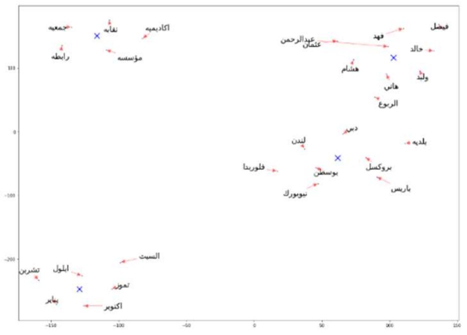
Person	فهد	فيصل	خالد	عبدالرحمن	وليد	هشام	هاني	عثمان
Location	باريس	بروكسل	بوسطن	دبي	فلوريدا	نيويورك	لندن	
Organization	رابطة	مؤسسة	اكاديمية	جمعية	نقابة	بلدية		
Time/Date	أيلول	الربوع	السبت	يناير	أكتوبر	تموز	تشرين	


Twitter CBOW Model


Twitter Skip-Gram Model


WWW CBOW Model


WWW Skip-Gram Model


Wikipedia CBOW Model

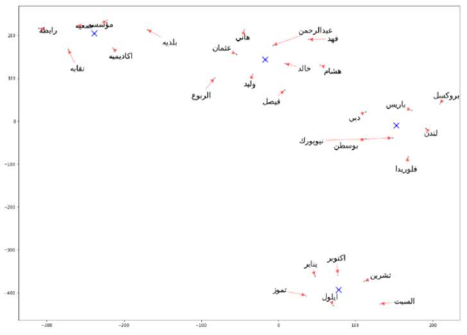

Wikipedia Skip-Gram Model

Figure 3: results of clustering named entities

5.2. Quantitative Evaluation

For evaluating the models quantitatively, we used the SemEval-2017 Semantic Textual Similarity Task 1[13] which aims to measure the degree of equivalence among paired snippets of text. SemEval enhanced the 2017 subtask by providing an Arabic dataset for this task. The test data for Arabic had just 250 pair of snippets and the challenge was to predict the similarity probability of the two snippets. Our aim was not to fully address this task, but rather to demonstrate that just by using a good word embedding model, we could obtain reasonable baseline scores for such a task. To do so, we've calculated a vector for all snippets by taking the mean of the vectors for the words in the snippet after multiplying each vector by its TF–IDF value (Term Frequency – Inverse Document Frequency[14]). Then we calculated the cosine similarity between the vectors of each two snippets to estimate the textual similarity probability. We then used the official evaluation tool[15] to evaluate each model as shown in Table 6. The results show that this very naïve approach provides results that are comparable to the competition's average scores; this is achieved with any feature engineering or the utilization of any complex machine learning models

Table 4 Results for SemEval Task

Model Name	**Score**
Twt-CBOW	0.56813
Twt-SG	**0.58459**
WWW-CBOW	0.57268
WWW-SG	0.56135
Wiki-CBOW	0.52842
Wiki-SG	0.54533
Competition Worst	0.0033
Competition Best	**0.7440**
Competition Average	0.52033

6. Conclusion and Future Work

In this paper, we have presented how we have built 6 different word embedding models for the Arabic language using three different resources: Wikipedia, Twitter and Common Crawl webpages crawl data. We have provided two models for each resource; one based on the Continuous bag-of-words and another on the Skip-gram model. We have evaluated our models using qualitative and quantitate measures on several tasks to show their ability to capture similarity among words. We believe that these pretrained models can be used by other researchers in the field of NLP to enhance the performance of various NLP task.

In future work, we would like to experiment with character level embeddings as well as apply these models to enhance many of our previously addressed problems amongst which are Arabic sentiment analysis and named entity recognition.

References

[1] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu and B. Qin, "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification.," in *ACL (1)*, 2014.

[2] S. K. Siencnik, "Adapting word2vec to Named Entity Recognition," in *NODALIDA*, 2015.

[3] C.-C. Lin, W. Ammar, C. Dyer and L. Levin, "Unsupervised pos induction with word embeddings," *arXiv preprint arXiv:1503.06760,*

[13] http://alt.qcri.org/semeval2017/task1/
[14] https://en.wikipedia.org/wiki/Tf%E2%80%93idf
[15] http://alt.qcri.org/semeval2017/task1/data/uploads/sts2017-trial-data.zip

2015.

[4] X. Ma and E. Hovy, "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* , Berlin, 2016.

[5] R. Al-Rfou, B. Perozzi and S. Skiena, "Polyglot: Distributed word representations for multilingual nlp," *arXiv preprint arXiv:1307.1662,* 2013.

[6] R. Soricut and F. J. Och, "Unsupervised Morphology Induction Using Word Embeddings.," in *HLT-NAACL*, 2015.

[7] R. e. a. Parker, "Arabic Gigaword Fifth Edition LDC2011T11," Web Download. Philadelphia: Linguistic Data Consortium, 2011.

[8] A. Zirikly and M. T. Diab, "Named Entity Recognition for Arabic Social Media.," in *VS@ HLT-NAACL*, 2015.

[9] K. Darwish, "Named Entity Recognition using Cross-lingual Resources: Arabic as an Example," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, 2013.

[10] M. A. Zahran, A. Magooda, A. Y. Mahgoub, H. Raafat, M. Rashwan and A. Atyia, "Word Representations in Vector Space and their Applications for Arabic," in *Computational Linguistics and Intelligent Text Processing: 16th International Conference, CICLing 2015, Cairo, Egypt, April 14-20, 2015, Proceedings, Part I*, A. Gelbukh, Ed., Cham, : Springer International Publishing, 2015, pp. 430-443.

[11] Y. Bengio, R. Ducharme, P. Vincent and C. Janvin, "A Neural Probabilistic Language Model," *J. Mach. Learn. Res.,* vol. 3, pp. 1137-1155, #mar# 2003.

[12] R. Collobert and J. Weston, "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning," in *Proceedings of the 25th International Conference on Machine Learning*, New York, NY, USA, 2008.

[13] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781,* 2013.

[14] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010.

[15] D. Ulyanov, *Muticore-TSNE,* GitHub, 2016.

[16] C. R. García-Alonso, L. M. Pérez-Naranjo and J. C. Fernández-Caballero, "Multiobjective evolutionary algorithms to identify highly autocorrelated areas: the case of spatial distribution in financially compromised farms," *Annals of Operations Research,* vol. 219, pp. 187-202, Aug 2014.

[17] C. O. Alm, D. Roth and R. Sproat, "Emotions from Text: Machine Learning for Text-based Emotion Prediction," in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Stroudsburg, 2005.