# Optimized Training and Evaluation of Arabic Word Embeddings

Jordan King, Lisa Singh, Eric Wang

November 16, 2015

## Abstract

Word embeddings are an increasingly important tool for NLP tasks, especially those that require semantic understanding of words. Methodologies and properties of English word embeddings have been extensively researched, however little attention has been given to the production and application of Arabic word embeddings. Arabic is far more morphologically complex than English due to the many conjugations, suffixes, articles, and other grammar constructs. Arabic words in text are often carry more contextual information about objects, tense, gender, and definiteness than English, meaning that Arabic unigrams occur less frequently on average than English unigrams. This has a significant effect on the training and application of Arabic word embeddings, as the embeddings are trained on unigram tokens. While there are a number of techniques to break down Arabic words through lemmatization and tokenization, the quality of resulting word embeddings must be investigated to understand the effects of these transformations. In this work, we investigate a number of preprocessing methods and training parameterizations to establish best practice strategies for training Arabic word embeddings. To enable this research, we required a semantic similarity task to evaluate the Arabic word embeddings. There is little work done to provide a large semantic similarity task in Arabic so we created a list of 1000 similarity scores for given Arabic word pairs using native Arabic speakers. We evaluated all training parameterizations by using the embeddings from a given parameterization to obtain a similarity score to evaluate against the task. Additionally, we evaluated the word embeddings' ability to capture syntactic properties of words using a part of speech tagging task. Using these tasks, we were able to identify

the training strategies that produce the best results for each task. We also offer a suite of Arabic NLP tools developed alongside this work that helps fill the void of accessible open source Arabic NLP tools. Altogether, this work provides best practices for training Arabic word vectors, an open semantic similarity task developed by native Arabic speakers, and a python package of Arabic text processing tools.

# 1 Introduction

Arabic word embeddings are numerical vector representations of a words meaning - both semantic meaning and syntactic meaning. These embeddings are obtained using machine learning algorithms - word2vec - that utilize the context a word appears in to infer its meaning. This works very well as words with similar meanings tend to be used in similar contexts, which are defined by the preceeding and following $n$ words. For example the sentences *I eat bread every night* and *I eat rice every night* are examples of how food words may appear in similar contexts. With enough text to process, we can train numerical vectors to learn that bread and rice appear in these *common-for-food* contexts. Similarly, we can learn syntactic relationships because different parts of speech appear in certain context patterns as well.

High quality word embeddings provide a representation of the meaning of a word, without ever translating or referencing a dictionary. We can obtain the semantic and syntactic meaning straight from a corpus of natural written language. With accurate word embeddings, we can perform powerful operations to investigate the relationships between words. A few of the possible operations are measuring the similarity of two words, identifying which word from a set is least similar, and solving basic analogies. The classic party trick for word embeddings is to take (the embeddings of) *king*, subtract *man*, and add *woman*. The resulting embedding is closest to the embedding for *queen*! Intuitively, this allows us to subtract the male gender meaning from kings embedding, add the female gender meaning, and end up with an embedding equivalent to queens embedding.

We would like accurate Arabic word embeddings so we can interpret the general topics of discussion in Arabic media without using translation or ignoring some words belonging to a topic. An example application would be to use the embeddings to learn what words are highly similar to words similar to fear (in Arabic), and then compute the degree to which some media is using fearful language in the context of a threatened city. This information could help us understand when

people feel forced to migrate from the city.

# 2 Related Literature

In progress.

# 3 Experiments

We performed a broad parameter sweep over various preprocessing techniques and word2vec parameterizations to determine the optimal word embedding methods. We evaluated each parameterization on two tasks - one for semantic similarity, and one for syntactic understanding. The text corpus for training the embeddings is a cleaned Arabic Wikipedia dump. Each part of this experiment is described in the following subsections.

## 3.1 Semantic Understanding Evaluation

The semantic similarity task consists of 1000 Arabic word pairs and a similarity score in the range 0-10, where 10 represents words that are extremely related. As no task existed of the size that we required for our parameter sweep, we developed this task ourselves.

The words began by selecting 1250 of the most common words from the Arabic Wikipedia, excluding words that occur in more than $5\%$ of sentences. These words word then translated into English with Google translate, queried against the big huge thesaurus API for either synonyms or antonyms, and translated back to Arabic. The original word and the resulting word are then paired up. One half of the pairs are synonyms, one quarter are antonyms, and one quarter are shuffled with other pairs to be randomly matched. The proportions were chosen because the synonyms database is more extensive than the antonym database. The various APIs involved introduce a large amount of noise, to the point that some synonym pairs will be completely unrelated Arabic words. We take advantage of this noise to distribute the relatedness of words across the 0-10 scale, while ensuring some pairs are related.

This list of word pairs was then given to 10 fluent Arabic speakers, along with simple instructions to evaluate the relatedness of the words on a scale of 1 to 5. The values that they provided were then averaged and scaled to 0 to 10. When

evaluating a parameterization, we performed the same preprocessing on the word list as we did to the corpus prior to training. Each word pair's embeddings were compared for a similarity score, and the parameterization recieved a score for its squared error off the task's similarity score.

## 3.2 Syntactic Understanding Evaluation

The syntactic understanding of the word embeddings was evaluated via a part-of-speech tagging task. A selection of Arabic documents were first tagged with part-of-speech values using Madamira's NLP analysis, once for each preprocessing method. For each parameterization, a simple recurrent neural network is trained to predict the part-of-speech of a word using its embedding. One document was held out as a test set for the network, and the accuracy of the network on this set was taken as the syntactic understanding score for the parameterization.

## 3.3 Preprocessing Options

The three main preprocessing options that we consider for this task are as-is, tokenized, and lemmatized. As-is leaves the corpus as it is. Tokenization breaks each word into simple grammatical tokens, separating the definite article and pronoun suffixes from the word. Lemmatization completely removes such affixes from the corpus, mapping each word to a base word that represents the simple meaning of the word. It reduces words to a single tense, gender, and definiteness while preserving grammatical forms.

Further considered preprocessing parameters are normalizing Arabic text, removing diacritics and reducing multiple forms of equivalent letters to a single letter.

## 3.4 Parameterizations

The main parameters of word2vec that are considered are window size, dimension, and algorithm. The window sizes considered are 5 and 8. The dimensions considered are 100 and 200. Both CBOW and Skipgram algorithms are considered.

### 3.5 Implementation

All operations defined above utilize the Arapy package we developed, which is released as an open source utility. All preprocessing options were precomputed first, generating multiple copies of the Arabic Wikipedia corpus. Then word vectors were generated for each parameterization. The vectors were then ran through both evaluation tasks, recording their performance.

### 3.6 Results

The final results are shown here. Notably, parameter xxx was the best performer on both tasks, suggesting that the method of preprocessing was much better.

## 4 Arapy

We developed a package of Arabic text processing tools while working on this research. This packages includes the following tools: Python Madamira wrapper, Arabic text normalization functions, Google Translate wrapper functions, Arabic thesaurus simulation, Arabic Wikipedia cleaning functions, and word embedding training wrappers.

## 5 Conclusion

This work provides best practices for training Arabic word vectors, an open semantic similarity task developed by native Arabic speakers, and a python package of Arabic text processing tools.