Optimized Training and Evaluation of Arabic Word Embeddings

Jordan King and **Lisa Singh**
Georgetown University
3700 O Street NW
Washington, DC, 20057, USA
jwk67@georgetown.edu and
lisa.singh@cs.georgetown.edu

Eric Wang and **David Buttler**
Lawrence Livermore National Laboratory
7000 East Ave
Livermore, CA, 94550, USA
ericxwang.py@gmail.com and
buttler1@llnl.gov

TODO: Clean up our task's experiment, do we want the tables, do we want Mean Squared error?, spellcheck

FUTURE: Freq/DomFreq/NeighborFreq/Buzz tasks, finalize tex style

Abstract

Word embeddings are an increasingly important tool for NLP tasks 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. This has a significant effect on the training and application of Arabic word embeddings. 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 guideline methodologies for training high quality Arabic word embeddings. Using various evaluation tasks, including a new semantic similarity task created by fluent Arabic speakers, we are able to identify training strategies that produce high quality results for each task. We also offer a suite of accessible open source Arabic NLP tools. To summarize, the main contributions of this work include improved methodologies for training Arabic word vectors, a 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 word's 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 (Mikolov et al., 2013a; Mikolov et al., 2013b). 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 directly from a corpus of natural written language. With accurate word embeddings, we can perform powerful vector operations to investigate the relationships between words in a corpus. 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 (Mikolov et al., 2013b). The classic demonstration of word embeddings is to take (the embeddings of) *king*, subtract *man*, and add *woman*. The resulting embedding should be near to the embedding for *queen* in the embedding vector space. Intuitively, this allows us to subtract the male gender meaning from king's embedding, add the female gender meaning, and end up with an embedding equivalent to queen's embedding (Mikolov et al., 2013c).

While word embeddings have been analyzed and evaluated for different tasks in English corpa (Mikolov et al., 2013a; dos Santos and Gatti, 2014), word embeddings in other languages have received less attention. Arabic word embeddings have been included in multi-lingual work on embeddings like generic part-of-speech tagging (Al-Rfou et al., 2013). However, the process of training word vectors for a language so morphologically different from English has not been explored. Every language has different levels of morphological complexity. This complexity may have a significant effect on how the word embeddings should be trained and the tasks that the word embeddings can be used for. According to one of the only papers attempting to quantify the Arabic vocabulary, Arabic itself has around 250 prefixes, 4500 regular derivative roots, 1000 derivative regular forms, and 550 suffixes to build words from, meaning there are around $6 * 10^{10}$ possible Arabic words (Ahmed, 2000). While the number of sensical words is estimated to be between 12 and 500 million words by unofficial sources (Souag, 2013; Da'na, 2012), the vocabulary is much larger than the 1 million word vocabulary of English (NPR, 2010). In this paper, we focus on different preprocessing and training parameters to bring the performance of Arabic word embeddings closer to that of English word embeddings.

Having word embeddings in different languages allows us to avoid translation when using these embeddings for different natural language processing and machine learning tasks. These embeddings can allow for the interpretation of topics in media, or provide an understanding of how similar words are to a subject. An example application would be to use the embeddings to learn what words are highly similar to words representing fear, and then compute the degree to which some media is using fearful language in the context of political or economic termoil.

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. Written Arabic words 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. For example, the sentence هو حمله الي بيتها translates to *He carried it to her house* The word بيتها for example, is a combination of بيت (*house*) and ها (*her*). This has a significant effect on the training and application of Arabic word embeddings as the embeddings are trained on unigram tokens. The complex words then occur with less frequency and more semantic meaning than the English counterparts of *house* and *her*.

The contributions of this work are as follows: 1) We perform a comparative empirical evaluation of Arabic and English word vectors using both a semantic similarity task and an analogy solving task. We show that standard parameters for English word embeddings can lead to poor Arabic word embeddings. 2) We present an empirical analysis identifying the parameters that are most effective for our tasks, identifying a set of best practices for training Arabic word embeddings. 3) We developed an open-source software package that provides easy access to important Arabic natural language processing tools.

The remainder of this paper is organized as follows. In Section 2 we present work related to training, evaluating, and utilizing Arabic word vectors. In Section 3 we provide an overview of the process and parameters required to train word vectors in Arabic. Section 4 describes our methodology to measure the quality of word vectors against semantic similarity tasks and an analogy solving task. In Section 5 we present the results of our parameterization experiments and evaluations of Arabic word vectors on the tasks. This leads to Section 6 in which we inspect our results to establish guidelines for creating the best performing Arabic word vectors for a task. Following this we describe the software created to perform our analyses in Section 7. Section 8 offers our thoughts on further work and conclusions.

2 Related Literature

Word embeddings have gained popularity over the past few years since Mikolov et al. published the word2vec algorithms in 2014 (Mikolov et al., 2013b; Mikolov et al., 2013a). While new algorithms and applications have received a great amount of research attention, word embeddings are often considered in the English-like language cases. Arabic differs greatly from English in many ways important to natural language processing. An excellent summary of the most important challenges that come with Arabic is provided by

Farghaly et al. (Farghaly and Shaalan, 2009). Al-Rfou et al. computed word embeddings for 100 languages using Wikipedia articles (Al-Rfou et al., 2013). This work is the closest to ours, as it inspired our system of semantic and syntactic evaluation. However, we believe our use of a semantic similarity task provides a better quantitative evaluation. Additionally, this work does not actually look at Arabic-specific training methods, which we would like to improve Arabic embedding quality. Zirikly et al. utilized Arabic word vectors to improve named-entity recognition performance, normalizing hamzas, elongated words, and number normalization (Zirikly and Diab, 2015). However, this work did not seek out any further improvements for training Arabic word vectors. Belinkov et al. utilize Arabic word vectors in a question answering task, reporting slight improvements when their training data was lemmatized using MADAMIRA (Belinkov, 2015). Further normalization is not performed in their work. Some research has been done to utilize morphology to alter the training algorithms of English word embeddings to learn morphological similarities (Luong et al., 2013), but this work makes no attempt to extend the method beyond English. This work is also focused on utilizing morphological similarities within a language rather than overcome morphological complexity that exists in a language as morphologically complex as Arabic. In summary, Arabic word vectors are being used, but the process of training them has not been explored or optimized as we aim to do with this work.

In English, there are some accessible open source natural language processing tools, especially those made available through Stanford University. However in Arabic, the list of strong NLP tools is a bit shorter. Habash et al. developed Mada+Tokan to perform tokenization, part of speech tagging, and lemmatization (Habash et al., 2009). Diab published the Amira software as fast and robust option for phrase chunking and POS tagging (Diab, 2009). Recently, these tools have been brought together into the MADAMIRA software package, comprised of a suite of Arabic NLP tools that includes tokenization, lemmatization, phrase chunking, and part of speech tagging (Pasha et al., 2014). While powerful and robust, MADAMIRA's lack of open source code and inaccessible input and output make it difficult to use in short NLP experiment scripts. Our python package provides a wrapper to help with this difficulty, providing simple calls to process and access commonly desired output from MADAMIRA.

Word similarity tasks are widely used for NLP experimentation and evaluation, and a long list of semantic similarity data was compiled by Faruqui et al. (Faruqui and Dyer, 2014). However, few of these are available in Arabic. Faruqui refers to two data sets that have been translated to Arabic by Hassan et al. (Hassan and Mihalcea, 2009), the 353 word WordSimilarity-353 and the 30 word Miller-Charles datasets (Finkelstein et al., 2001; Miller and Charles, 1991). However, this translation was done by a single Arabic speaker using the English semantic similarity scores (Hassan and Mihalcea, 2009). In their paper, they cite that with 5 translators on a Spanish task, they obtained unanimous translations 74% of the time, and further rescoring produced a correlation of .86. Our work attempts to alleviate these losses by beginning with Arabic words and evaluating them all with multiple fluent Arabic speakers.

3 Training Word Embeddings in Arabic

There are a number of decisions to be made when training word embeddings in Arabic. We have chosen to use the word2vec framework to train, although there are other proposed methods to obtain word embeddings that are highly similar (Pennington et al., 2014). We chose word2vec as there is more public research available for reference as well as excellent open software support. The main decisions to be made when training word2vec embeddings in Arabic are how to preprocess the text, how to normalize the text, and how to parameterize the word2vec algorithms. The remainder of this section describes different consideration for each of these steps, explains options and training parameters that can be adjusted, and presents the specific training parameters we used in our evaluation. The high level process of training Arabic word embeddings is illustrated in Figure 1.

3.1 Preprocessing Options

Preprocessing is very important when analyzing Arabic text. Much of the linguistic information in the grammar is contained in various affixes to words. This is very different from English, where information is often contained in stand-alone pronouns and articles. Word2vec captures information at a word level, so separating these affixes into

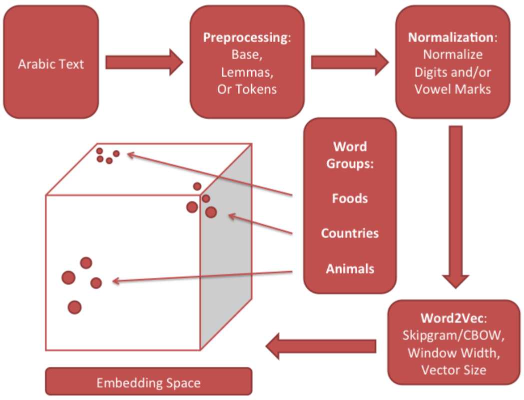

Figure 1: Training Arabic Word Embeddings

individual words greatly changes what is learned during training.
The three main preprocessing options that we consider for this task are 1) leave the text unedited, 2) tokenize the text to make affixes individual words, and 3) lemmatize the text to drop most affixes and preserve only the core idea of each Arabic word.Tokenization breaks each word into simple grammatical tokens and creates separate words from affixes such as the definite article and the various pronouns. Lemmatization completely removes such affixes from the corpus, mapping each word to a base word that represents the core meaning of the word. It reduces words to a single tense, gender, and definiteness, but preserves the basic grammatical form. An English equivalent would be to map both *he jumped* and *she jumps* to *he jumps*.

3.2 Normalization

Normalizing Arabic text can greatly reduce the sparsity of the word space in Arabic. We always normalize the corpus by removing English characters, reducing all forms of the letters alif, hamza, and yaa to single general forms (respectively ا,ء,and ي). The options we consider variable are removing diacritics and reducing both English and Arabic numerical characters to the number sign.

3.3 Parameterizations

The main parameters of word2vec that are considered are algorithm, embedding dimension, and window size. Both CBOW and Skipgram algorithms are considered (Mikolov et al., 2013a). The embedding dimensions considered are 100 and

Parameter	Value	Explanation
sg	$[0, 1]$	Algorithm
$size$	$[100, 200]$	Dimensionality
$window$	$[4, 7]$	Context window
$mincount$	5	Filters rare words
$sample$	$1e-5$	Downsampling
$seed$	1	Random seed
hs	1	Hierarchical softmax
$negative$	0	Negative sampling
$iterations$	5	Training iterations

Table 1: Training Parameters

200. We chose these vector sizes as the typical range is between 100 and 300, where more dimensions require more time and data to train well. We believe we lack sufficient Arabic text data to fully benefit from higher dimensions, so we chose to keep only smaller dimensionalities. The window sizes considered are 4 and 7, which is how far to either side of the word being trained we look for context. For both the vector sizes and the window widths we were limited to two values for the sake of time. Training models for all combinations of parameters listed above results in 96 models, each requiring well over an hour to train. We believe that our choices provide us with sufficient granularity to understand how Arabic text can be best be used to train high quality word embeddings. For a complete list of the Word2Vec parameter choices, including the static parameters, refer to Table 1. Hierarchical softmax and negative sampling are methods to sample training data efficiently. Downsampling is used to decrease the influence of high frequency words in the corpus. We use hierarchical sampling and some downsampling as together they have been shown to perform well on complex vocabularies with infrequently represented words and phrases (Mikolov et al., 2013b).

4 Evaluating Arabic Word Embeddings

It is a complex problem to evaluate the quality of word embeddings. The word2vec methods produce unsupervised vectors that maximize the probability of predicting a word given the context that it appears near in the training corpus. We evaluate the embeddings on semantic similarity tasks as well as an analogy solving task.

4.1 Semantic Similarity Tasks

The semantic similarity tasks consist of pairs of words associated with a human-labeled similarity value. The largest Arabic semantic similarity task that we could find is the WordSimilarity-353 task, which was developed in English and then manually translated into Arabic (Finkelstein et al., 2001; Hassan and Mihalcea, 2009).

We also created a semantic similarity task consisting of 1000 word pairs with similarity scores. Between 2 and 5 fluent Arabic speakers labeled each word pair with a similarity score in the range 0-1, where pairs with a score of 1 indicates that the words are extremely related. To begin creating this task, we selected 1250 of the most common words in the Arabic Wikipedia dump (Wikipedia-Meta, 2016) at `https://dumps.wikimedia.org/arwiki/20150901/`, excluding words that occur in more than 5% of the sentences. The remaining words were then translated into English with Google translate, queried against the Big Huge Thesaurus API for either synonyms or antonyms, and translated back to Arabic (Google, 2016; Big-Huge-Labs, 2016). The original word and the resulting synonym or antonym were then paired up. Half of the pairs are at this point synonyms, one quarter are antonyms, and one quarter are shuffled with other pairs to be randomly matched. This distribution is synonym heavy because the Big Huge Thesaurus database has more data on synonyms than antonyms. The various APIs involved introduce a large amount of noise, to the point that some synonym pairs end up as unrelated Arabic words. We take advantage of this noise to distribute the relatedness of words across the 0 to 1 scale.

This list of 1250 word pairs was then distributed to fluent Arabic speakers such that each pair is scored by textcolorredmultiple evaluators. We provided simple instructions to evaluate the relatedness of the words on a scale of 0 to 5 for ease of labeling. The values that they provided were then scaled from 0 to 1 and averaged. We computed an average inter-rater reliability score of 0.7022 using Pearson correlation between pairs of raters.

When evaluating a model parameterization with the WordSimilarity-353 task or our similarity task, we perform the same preprocessing on the word pairs as we do on the training corpus for each model. Each word pair's embeddings are first obtained from the model, and then an absolute cosine similarity score is obtained between them. The cosine similarity is compared against the similarity task's score. The model is scored on both the mean absolute difference between the scores and the correlation between the task scores and model scores.

Type	**Query**	**Query**	**Query**	**Answer**
Capital city	Athens	Greece	Oslo	Norway
Gender	brother	sister	grandson	granddaugl
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars

Table 2: Analogy Examples

4.2 Analogy Task

The analogy task is a standard for evaluating word vectors first used by Mikolov et al. (Mikolov et al., 2013a). It consists of analogy questions each composed of three query words and one answer word, in the form of an analogy such that $query_1$ is to $query_2$ as $query_3$ is to $answer$. We used the Google Translate API to translate the 19544 English analogies to Arabic (Google, 2016). This translated model is available with our code. For each model that we trained, we performed matching preprocessing to each item in each analogy. The first three analogy items are then coverted to two positive vectors and one negative vector and averaged to obtain a fourth result vector. A correct answer on this task is one for which the closest vector to the result in the model matches the fourth analogy item.

This task is composed of categories of analogies, with a mix of syntactic and semantic analogies. This allows this task to evaluate the syntactic abilities of our models to complement our semantic similarity evaluations. See Table 2 for examples of these analogies in English taken from Mikolov's originating paper (Mikolov et al., 2013a).

5 Word Embedding Experiments

Using the parameter selections outlined in Section 3, we perform a partial parameter sweep over the various preprocessing techniques, normalization options, and word2vec parameterizations to determine the optimal word embedding methods. The text corpus for training the embeddings is an Arabic Wikipedia dump

Method	Similarity Correlation	Analogy Accuracy
Google News	0.6979	0.7359
5 Million	0.5458	0.0452

Table 3: English Baseline Results

from `https://dumps.wikimedia.org/arwiki/20150901/` (Wikipedia-Meta, 2016), cleaned by dropping Wikipedia markup, punctuation, and non-Arabic characters. All preprocessing options are precomputed first, generating multiple versions of the Arabic Wikipedia corpus. Then word vectors are trained for each parameterization. The vectors are then ran through the evaluation tasks, recording performance statistics.

5.1 Word Similarity 353

The results for English vectors on the semantic similarity tasks are shown in Table 3 for comparison. There are two models shown, each evaluated on the WS353 English word similarity task. The first is an English model trained under the default Word2Vec parameterization (skipgram, window of 7, 100 dimensions) on the same number of words as our Arabic models. The second is the publically available pre-trained vectors trained on a 100 billion word Google News Corpus (Mikolov et al., 2013b). The metric that we choose to base our evaluate on is the Spearman correlation between the model similarity estimates the evaluation task similarity values. We also provide the mean squared error as the mean squared difference between the model estimate and the evaluation task value. The default vectors exhibit an impressive .268 MSE, and both models show a high correlation with the evaluation task scores. The Google News vectors display an impressive .6978 Spearman correlation score to the task, providing a high score to aim for. We consider the 0.5457 correlation score of the default English task to be the baseline for our Arabic word embeddings.

Table 4 shows the results of the models with the 10 highest correlation scores on the Word Similarity 353 task (Finkelstein et al., 2001; Hassan and Mihalcea, 2009). Do these tables add? Standing out is the lack of any preprocessing method but tokenization in this list. Additionally, these best performing models were primarily trained using a window of 4 words. Figure 2 shows box plots over all results grouped by the two most significant preprocessing measures, Window size and Preprocessing method. These results show that tokenization is the only method not only as good as teh English baseline, but able to perform better. This improvement can be considered even more significant due to the translation of the task, but it is difficult to quantify this effect. Interestingly, the unprocessed Arabic scores higher than lemmatization on this task. textcolorredInterpret. The window size is very interesting, as this parameter is highly dependent on the grammar of the training language. A sentence structure that uses complex words more often has related words nearer to each other than English does, so Arabic word embeddings may benefit from having a smaller window to not look beyond the relevant information.

Rank	Preprocessing	Window	Size	Correlation
1	tokens	4	200	0.5662
2	tokens	4	100	0.5557
3	tokens	4	200	0.5486
4	tokens	4	100	0.5418
5	base	4	200	0.5317
6	tokens	4	100	0.5315
7	tokens	4	100	0.531
8	tokens	7	100	0.5295
9	base	4	200	0.5248
10	tokens	4	200	0.5248

Table 4: Top Results on Word Similarity 353

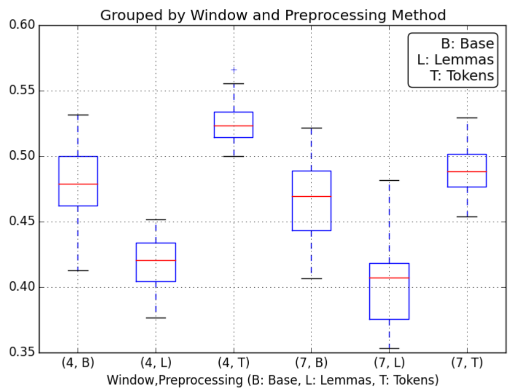

Figure 2: WS353 Results

5.2 Our Similarity Task

Table 5 shows the results of the models with the 10 highest correlation scores on the task we developed Pick one of the three model options. The subset with 4 votes has the highest correlation but only 250 word pairs. The results on our data sup-

Rank	Preprocessing	Window	Size	Correlation
1	base	7	200	0.4619
2	base	4	200	0.4596
3	base	4	100	0.4594
4	lemmas	4	100	0.4589
5	base	4	200	0.4544
6	base	7	200	0.4543
7	base	4	100	0.4506
8	tokens	4	100	0.4505
9	base	4	100	0.4505
10	base	4	200	0.4474

Table 5: Top Results on our whole Task

Rank	Preprocessing	Window	Size	Correlation
1	base	4	200	0.576
2	base	4	100	0.5733
3	base	4	200	0.5687
4	base	4	200	0.5654
5	base	4	100	0.5566
6	base	7	200	0.5529
7	base	4	200	0.5527
8	lemmas	4	200	0.5509
9	lemmas	4	200	0.5508
10	tokens	4	200	0.5467

Table 7: Top Results on our Task-4 Votes

Rank	Preprocessing	Window	Size	Correlation
1	base	4	100	0.5398
2	base	4	200	0.5194
3	base	4	200	0.5022
4	base	4	200	0.5009
5	base	7	100	0.5
6	lemmas	4	100	0.496
7	base	4	100	0.493
8	base	4	200	0.4923
9	base	4	100	0.4917
10	lemmas	7	100	0.4912

Table 6: Top Results on our Task-2+ Votes

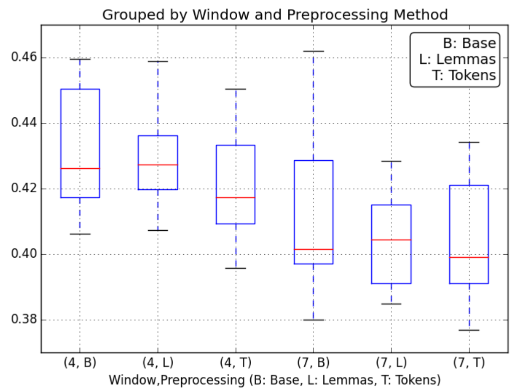

Figure 3: Our Whole Task Results

port our findings that the smaller window size does indeed have a strong positive impact on the quality of the word embeddings. We suspect that the high performance of the control group without preprocessing is due to the evaluation task being developed from the same corpus as the vector training data. Interpret, verify?

We have shown that preprocessing and training decisions can substantially change the performance of Arabic word embeddings on similarity tasks. Some methods were even able to surpass the English baseline. While the best performing models were still significantly below the scores of the English embeddings trained on the Google News corpus, this is to be expected considering the strong correlation between the quantity of training data and the quality of the word embeddings. Our training set has approximately 5 million Arabic words while the Google News set has about 100 billion words (Mikolov et al., 2013b).

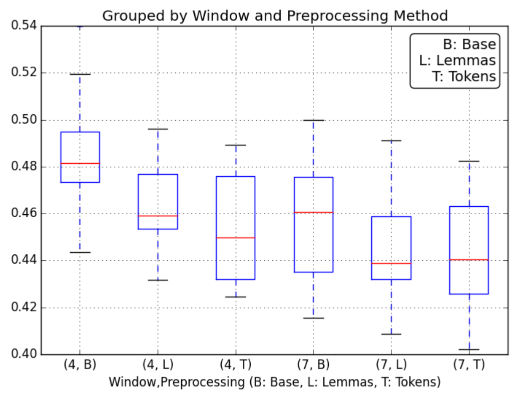

Figure 4: Our Task 2+ Votes Results

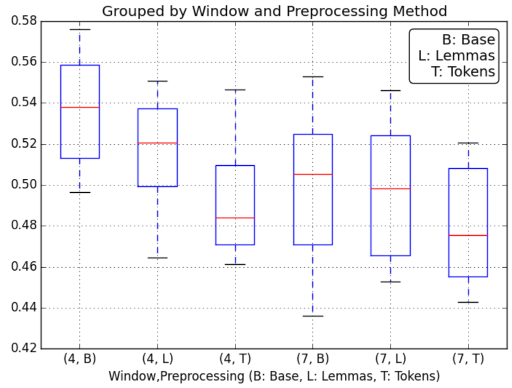

Figure 5: Our Task 4 Votes Results

Model	Accuracy
Google News	0.735878018829330826
5 Million	0.04522560880627239

Table 8: English Analogy Results

5.3 Analogy Task

Table 8 shows the baseline results of the English embeddings on the Analogy task. These results demonstrate the extreme difference in quality between vectors trained on 5 million words and 100 billion words. The lower accuracy of the 5 million word model at 0.04522, or 4.522 percent correct of the 19544 analogies, will be used as a comparitive baseline for this task.

Table 9 shows the top 10 analogy task results from the Arabic models. Here it seems models preprocessed to lemmas and trained to 200 dimensions seem to dominate. We also see less difference between models with different window sizes. Figure 6 confirms these trends with boxplots with groups by significant factors. This plot illustrates the dramatic improvements that are obtained with proper preprocessing and parameterization for the task. The lemmatized 200 dimensional models consistently outperformed all other models, including the baseline English model. In the best case, one ideally parameterized model is nearly 50% better than the English baseline. Of lesser note, the tokenization method also delivers significantly higher accuracies than the models that received no preprocessing on the Arabic. These results demonstrate that preprocessing and training decisions can greatly improve the performance of Arabic word embeddings on analogy solving tasks, improving scores from as low as half of the English baseline to as high as 150% of the baseline.

Rank	Preprocessing	Window	Size	Accuracy
1	lemmas	4	1	0.0671
2	lemmas	4	2	0.0647
3	lemmas	7	3	0.0637
4	lemmas	4	4	0.0634
5	lemmas	7	5	0.0626
6	lemmas	7	6	0.0623
7	lemmas	7	7	0.0621
8	lemmas	4	8	0.061
9	lemmas	4	9	0.0607
10	lemmas	7	10	0.0591

Table 9: Top Results on Arabic Analogy Task

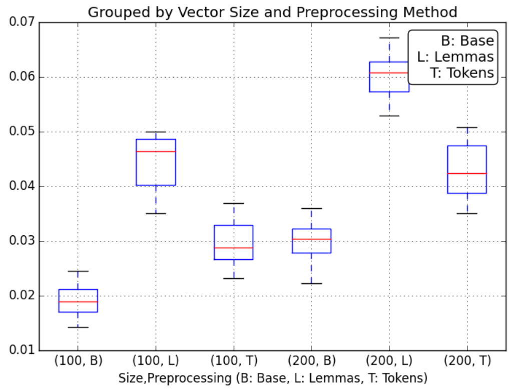

Figure 6: Arabic Analogy Task Results

6 Recommendations for Arabic Word Embeddings

Our results have affirmed that there is no free lunch when choosing methods to train Arabic word embeddings. There is no parameterization that outperformed all others on our tasks, but we have shown that within a task different models can offer dramatically different performance. When developing Arabic word embeddings that will be used in a context requiring performance at a level comparable to English word embeddings, it is necessary to inspect all preprocessing methods available. However, we do offer some guidelines based on our results.

For preprocessing, we believe that unprocessed Arabic may perform well if it is trained on the same data on which it is applied. Due to the emphasis on syntactic analogies in the analogy task, we suggest trying lemmatization for tasks requiring syntactic analysis. We suspect that it performs well as it reduces complex words to simpler forms the retain their basic syntactic structure. For semantic-heavy analysis, we suggest trying tokenization as it performed so well on the Word Similarity 353 similarity task. Tokenization likely performs well as it does not reduce the text, but isolates each core word in a broken down context. However, we reiterate that we believe it is essential to try at least one model from each of these methods on a specific application, as they have been shown to perform very differently across different tasks.

For normalization, we saw nearly no difference when we removed vowels or normalized numerical digits. As much written Arabic does not have vowels, and leaving digits could be useful, we suggest not normalizing these things at all. For training, we did not find a dominant training algorithm between Skip-Gram and CBOW. However, we do believe the smaller window size of 4 demonstrated significantly better results globally. We also found large improvements on the analogy task for 200 dimensional embeddings, and no evidence of draw backs on other tasks. With more data and time, it may be possible to obtain even better performance with 300 dimensions, as the Google News embeddings showed on the analogy task. Other parameters did not show significant improvements on any of the evaluation tasks.

7 Arabic NLP Package

We have developed a simple python package for all of the utilities that we required to complete this research. As the software for Arabic NLP is somewhat difficult to acquire and apply, we believe that the utilities and wrappers we provide are greatly useful to anyone wanting to perform common NLP tasks in Arabic. We call this open source package Arapy, and it will be available at `https://github.com/jordanking/arapy` upon completion of this work.

All preprocessing, normalization, and training processes outlined in this research utilize Arapy. The package itself utilizes various software packages and resources, including gensim, MADAMIRA, the Google Translate API, and the Big Huge Thesaurus API (**?**; Pasha et al., 2014; Google, 2016; Big-Huge-Labs, 2016).

Arapy includes many useful tools for simple NLP tasks that can be difficult when working with Arabic. The first is a module providing a MADAMIRA wrapper that provides access to the part-of-speech tagging, base phrase chunking, tokenization, and lemmatization features of MADAMIRA. There is also a wrapper providing various tools for training and evaluating Arabic word embeddings, primarily as a wrapper for gensim tools. Arapy also includes modules for Arabic text normalization, translation with Google Translate, simulation of an Arabic thesaurus with translation and the Big Huge Thesaurus API, and cleaning Arabic Wikipedia dumps. The main dependancies for this package are Java for MADAMIRA, the MADAMIRA jar and a license for MADAMIRA, gensim for word2vec model operations, a Google API key for translation, a Big Huge Thesaurus API key for thesaurus simulation. Complete documentation can be found in the repository at `https://github.com/jordanking/arapy`.

8 Conclusion and Future Directions

The contributions of this work are as follows: 1) We perform a comparative empirical evaluation of Arabic and English word vectors using both a semantic similarity task and an analogy solving task. We show that standard parameters for English word embeddings can lead to poor Arabic word embeddings. 2) We present an empirical analysis identifying the parameters that are most effective for our tasks, identifying a set of best practices for training Arabic word embeddings. 3) We developed an open-source software package that provides easy access to important Arabic natural language processing tools.

We have a few research directions that we believe would extend this work. We would like to expand our analysis with more training data and experiment with more parameter choices, which were heavily restricted by time. We would like to explore different applications for word embeddings and investigate if training parameters produces siginificantly different performances. Finally, we would like to continue to expand our NLP package, Arapy, to help other researchers perform Arabic NLP with ease.

References

Mohamed Attia Mohamed Elaraby Ahmed. 2000. *ALarge-SCALE COMPUTATIONAL PROCESSOR*

OF THE ARABIC MORPHOLOGY, AND APPLICATIONS. Ph.D. thesis, Faculty of Engineering, Cairo University Giza, Egypt.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. *arXiv preprint arXiv:1307.1662*.

Yonatan Belinkov. 2015. Answer selection in arabic community question answering: A feature-rich approach. In *ANLP Workshop 2015*, page 183.

Big-Huge-Labs. 2016. Big huge thesaurus: Synonyms, antonyms, and rhymes (oh my!). `https://words.bighugelabs.com/`. (Visited on 11/30/2015).

Muhammad Da'na. 2012. (5) how many words does the arabic language have? - quora. `https://www.quora.com/How-many-words-does-the-Arabic-language-have`, 12. (Accessed on 04/04/2016).

Mona Diab. 2009. Second generation amira tools for arabic processing: Fast and robust tokenization, pos tagging, and base phrase chunking. In *2nd International Conference on Arabic Language Resources and Tools*. Citeseer.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Ali Farghaly and Khaled Shaalan. 2009. Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(4):14.

Manaal Faruqui and Chris Dyer. 2014. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA, June. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Google. 2016. Translate api - translate api google cloud platform. `https://cloud.google.com/translate/docs`. (Visited on 11/30/2015).

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. Mada+ tokan: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt*, pages 102–109.

Samer Hassan and Rada Mihalcea. 2009. Cross-lingual semantic relatedness using encyclopedic knowledge. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1192–1201. Association for Computational Linguistics.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113. Citeseer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

NPR. 2010. Google book tool tracks cultural change with words : Npr. `http://www.npr.org/2010/12/16/132106374/google-book-tool-tracks-cultural-change-with-words`, 12. (Accessed on 04/04/2016).

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Lameen Souag. 2013. Jabal al-lughat: Does arabic have the most words? don't believe the hype. `http://lughat.blogspot.com/2013/12/does-arabic-have-most-words-dont.html`, 12. (Accessed on 04/04/2016).

Wikipedia-Meta. 2016. Data dumps — meta, discussion about wikimedia projects. [Online; accessed 5-April-2016].

Ayah Zirikly and Mona Diab. 2015. Named entity recognition for arabic social media. In *Proceedings of naacl-hlt*, pages 176–185.