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`

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 best practice strategies for training Arabic word embeddings. Using a new semantic similarity task created by fluent Arabic speakers we are able to identify the training strategies that produce the best results for each task. We also offer a suite of accessible open source Arabic NLP tools. Together, 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 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., 2013; **?**). 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. 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.

We would like accurate Arabic word embeddings so we can interpret the general topics of discussion in Arabic media without using translation, or to help understand which words are relevant to a topic by similarity. 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. This has a significant effect on the training and application of Arabic word embeddings, as the embeddings are trained on unigram tokens.

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 a syntactic similarity task. We show that standard parameters for English word embeddings lead to poor Arabic word embeddings. 2) We develop and open-source a simple software package that provides easy access to important Arabic natural language processing tools. 3) We present a semantic similarity task using a team of native Arabic speakers. This task is larger than other published tasks and uses multiple native speakers to manually provide high quality human labels. 4) 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.

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., 2013; **?**). 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. 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. Elaborate other training methods a bit. The main decisions to be made when training word embeddings in Arabic are how to preprocess the text, how to normalize the text, and how to parameterize the word2vec algorithms.

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 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 ,,and Ł – arabic letters not showing up?). The options we consider variable are removing diacritics and reducing both English and Arabic numerical characters to the number sign.

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	Heirarchical samp.
$negative$	0	Negative samp.
$iterations$	5	Training iterations

Table 1: Training Parameters

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. The embedding dimensions considered are 100 and 200. We chose these dimensions as the typical range is between 100 and 300, where 300 requires more time and data to train. We lack the gratuitous amount of data that is available freely for English, so we chose to keep only the smaller two dimensionalities. The window sizes considered are 5 and 8, which is how far to either side of the word being trained we look for context. For the other parameters, refer to Table 1.

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.

As we perform a large parameter sweep in 5, we want a large semantic similarity task to accurately evaluate the Arabic word embeddings. The largest Arabic semantic similarity task that we could find is a manually translated version of the WordSimilarity-353 task (Finkelstein et al., 2001; Hassan and Mihalcea, 2009). As we wanted a larger list generated specifically for Arabic and evaluated by multiple fluent Arabic speakers, we created a list of 1000 similarity scores for given Arabic word pairs using fluent Arabic speakers. The semantic similarity task consists of 1000 Arabic word pairs given a similarity score in the range 0-1. 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 find out how to cite this, fix the following citations, excluding words that occur in more than 5% of sentences. These words were then translated into English with Google translate (Google,), queried against the big huge thesaurus API for either synonyms or antonyms, and translated back to Arabic (Labs,). The original word and the resulting synonym or antonym are 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 synonym database is more extensive and accurate 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 attempt to distribute the relatedness of words across the 0 to 1 scale.
This list of word pairs was then given in parts to fluent Arabic speakers such that each pair is scored by multiple evaluators. We provided simple instructions to evaluate the relatedness of the words on a scale of 0 to 5. The values that they provided were then scaled from 0 to 1 and averaged. 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 an absolute cosine similarity score and the parameterization is given a score for its mean absolute difference from the task's similarity score.

5 Experimental Results

We performed a broad 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 cleaned with our Arapy package.
All preprocessing, normalization, and training processes utilize the Arapy package we developed, which is released as an open source utility. It utilizes various software resources, including gensim, Madamira, the Google Translate API, the Big Huge Thesaurus API, etc. CITE THESE. 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 evaluation tasks, recording various performance statistics.
The baseline results for English vectors are shown in Table 2. There are two models shown, each evaluated on the WS353 English word similarity task. The first is an English model trained under a default parameterization on the same number of words as our Arabic models. The second is the publically available pre-trained vectors trained on the 300 billion word Google News Corpus. The metrics that we choose to display are Mean Error and Correlation with the evaluation task. The mean error is obtained as the mean absolute distance between the vector estimate and the evaluation task estimate for word pair similarity. The default vectors exhibit an impressive .268 mean error, and both models show a high correlation with the evaluation task scores.

Method	**Mean Error**	**Correlation**
english default	0.268	0.549
google news corpus	0.313	0.65

Table 2: Baseline English Results

Tables 3 and 4 show the results of models trained on differently preprocessed text. Table 3 contains results from the similarity task that we developed, and Table 4 contains the results from the WS353 CITE Arabic task. None of the Arabic models reach either the level of coorelation or accuracy that the default English model does. These tabels are sorted by mean error, revealing traits that improved their performance on evaluation tasks. We analyze these results in Section 6.

6 Recommendations for Arabic Word Embeddings

The least accurate results for both Arabic semantic similarity tasks were those models trained on unprocessed Arabic text. Additionally, the accuracies of unprocessed text were far below the English default baseline. This goes to show that Arabic word embeddings do benefit in semantic accuracy when preprocessed to tokens and lemmas. Additionally, tokenized Arabic tends to have a higher accuracy on the tasks than lemmatized Arabic.
Also to note is the effect of normalizing digits and

Method	Mean Error	Correlation
tokens	0.304	0.446
tokens+dig	0.304	0.445
lemmas+tash	0.306	0.473
lemmas+dig+tash	0.307	0.454
tokens+dig+tash	0.31	0.419
lemmas+dig	0.311	0.462
tokens+tash	0.311	0.412
lemmas	0.313	0.446
control+dig	0.313	0.485
control+tash	0.317	0.475
control+dig+tash	0.318	0.478
control	0.319	0.477

Table 3: Results on Our Task

Method	Mean Error	Correlation
tokens+dig	0.306	0.54
tokens+dig+tash	0.308	0.512
tokens+tash	0.314	0.517
tokens	0.317	0.513
lemmas+tash	0.324	0.413
lemmas+dig+tash	0.328	0.438
lemmas	0.33	0.383
lemmas+dig	0.333	0.406
control+dig	0.341	0.483
control	0.342	0.489
control+dig+tash	0.343	0.47
control+tash	0.347	0.494

Table 4: Results on ws353 Task

tashkil (voweling). When lemmatizing the Arabic, there seems to be a consistent increase in performance if you then remove tashkil. Tokenization and unprocessed Arabic do not seem to have the same improvements with lemmatization.

We are currently training embeddings with the word2vec parameters considered. These results are only the tip of the parameters we are considering. With the algorithm parameters changed, these preprocessing effects should reveal greater effects. However, there was a bug discovered in the training code and these embeddings require many days to train again.

7 Arapy

We developed a package of Arabic text processing tools while working on this research. Arapy includes many useful tools for simple natural language processing 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, largely as a wrapper for gensim methods. Arapy also includes a modules for Arabic text normalization, translation with Google Translate, simulation of an Arabic thesaurus using translation and the Big Huge Thesaurus API, and cleaning the Arabic Wikipedia dump.

The 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,

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 a syntactic similarity task. We show that standard parameters for English word embeddings lead to poor Arabic word embeddings. 2) We develop and open-source a simple software package that provides easy access to important Arabic natural language processing tools. 3) We present a semantic similarity task using a team of native Arabic speakers. This task is larger than other published tasks and uses multiple native speakers to manually provide high quality human labels. 4) 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.

References

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.

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.

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. 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.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. 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.

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

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*.

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