METHODS TO OVERCOME CHALLENGES WHEN LEARNING ARABIC WORD EMBEDDINGS FOR TEXT MINING TASKS

A Senior Thesis
submitted to the Faculty of the
Graduate School of Arts and Sciences
of Georgetown University
in partial fulfillment of the requirements for the
degree of
Bachelor of Science
in Computer Science

By

Jordan King, Computer Science Undergraduate

Washington, DC
May 5, 2016

TODO: spellcheck FUTURE DRAFT: Freq/DomFreq/NeighborFreq/WeightedBuzz tasks

METHODS TO OVERCOME CHALLENGES WHEN LEARNING ARABIC WORD EMBEDDINGS FOR TEXT MINING TASKS

Jordan King, Computer Science Undergraduate

Senior Thesis Advisor: Dr. Lisa Singh

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 thesis, 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 then apply the best models from these experiments to a text mining task where we measure the buzz about a subject in a corpus of Arabic news documents. This application highlights the utility of word embeddings for data mining tasks. We also offer a suite of accessible open source Arabic NLP tools to increase the use of Arabic word embeddings within the research community. 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, an application of Arabic word embeddings for the buzz detection text mining task, and a python package of Arabic text processing tools.

INDEX WORDS: Arabic, Natural Language Processing, Word Embeddings, Word Vectors, Buzz, Text Mining

TABLE OF CONTENTS

CHAPTER

LIST OF FIGURES

List of Tables

CHAPTER 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 [17, 18]. This approach 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 [18]. The classic demonstration of word embeddings is to take (the embeddings of) *king*, subtract *man*, and add *woman*. The resulting embedding should be near 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 [19].

While word embeddings have been analyzed and evaluated for different tasks in English corpa [17, 8], 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 [2]. 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 [1]. While the number of sensical words is estimated to be between 12 and 500 million words by unofficial sources [28, 5], the vocabulary is much larger than the 1 million word vocabulary of English [21]. 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. We evaluate the quality of our embeddings using word similarity and analogy tasks. These evaluation metrics are supported by the work of Schnabel et al. that details the process of evaluating word embeddings [27].

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 a particular concept, such as fear, and then compute the degree to which some media sources use fearful language during 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*. Because of these types of differences between Arabic and English, a need exists to conduct a more detailed analysis of Arabic word embeddings.

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 demonstrate the utility of high quality Arabic word embeddings on a novel buzz detection task. 4) 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. After establishing practices to train word embeddings, we apply

Arabic word embeddings to a buzz detection task in Section 7. Following this we describe the software created to perform our analyses in Section 8. Section 9 offers our thoughts on further work and conclusions.

CHAPTER 2

RELATED LITERATURE

Word embeddings have gained popularity over the past few years since Mikolov et al. published the word2vec algorithms in 2014 [18, 17]. 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. [9]. Al-Rfou et al. computed word embeddings for 100 languages using Wikipedia articles [2]. 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, Farghaly et al. 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 [32]. 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 [3]. 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 [16], 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 [13]. Diab published the Amira software as fast and robust option for phrase chunking and POS tagging [6]. 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 [22]. 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. [10]. However, few of these are available in Arabic. Faruqui refers to two data sets that have been translated to Arabic by Hassan et al. [15], the 353 word WordSimilarity-353 and the 30 word Miller-Charles datasets [11, 20]. However, this translation was done by a single Arabic speaker using the English semantic similarity scores [15]. 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.

Word embeddings have been used in many tasks, from sentiment classification to semantic translation. Zhang et al. have shown that word embeddings capture enough information to classify sentiment [31]. Dickinson et al. have used word embeddings with neural networds to classify the sentiment of tweets related to publicly traded companies such that it correlates with stock prices [7]. By aligning words in corpuses from two languages, Wolf et al. have shown that word embeddings from multiple languages can be used for translation. Beyond these unique applications, word embeddings enable many numerical analysis tools to be applied to text.

Methods to measure various qualities and phenomena of text are well researched. Our task is fairly unique in its application of word embeddings to measure buzz, but is inspired and influenced by a many well-explored text mining tasks. One of the closest related works to our application of word embeddings for buzz detection is Rekabsaz's work showing improvements in text document querying when using semantic information from word embeddings to retrieve domain specific documents [26]. Other similar work is done in the field of event detection, where trends in text documents are identified with machine learning, data mining, and graph theory [24]. Similar works develop measures of semantic meaning without the use of word embeddings, using many sources of text to identify events and storylines [29]. The buzz application looks at the domain specific retrieval problem in a time series setting similar to event detection.

CHAPTER 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 [23]. 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 high level process of training Arabic word embeddings is illustrated in Figure 3.1. 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.

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 base text unedited (base), 2) tokenize the text to make affixes individual words (tokens), and

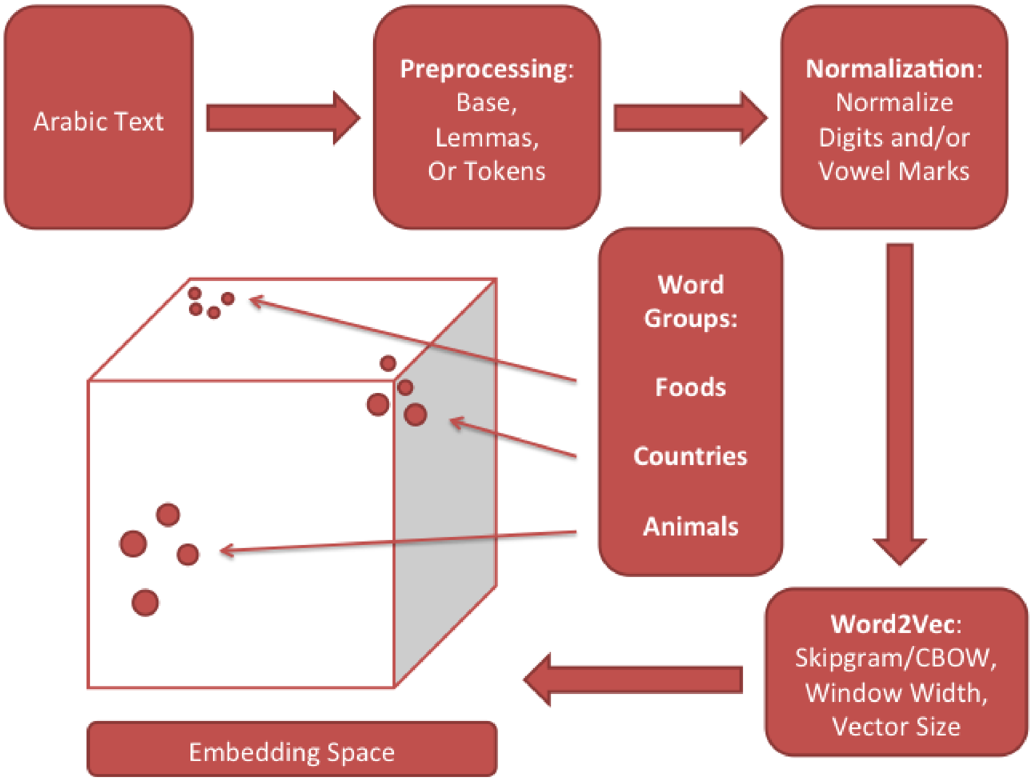
Arabic Text
Preprocessing: Base, Lemmas, Or Tokens
Normalization: Normalize Digits and/or Vowel Marks
Word2Vec: Skipgram/CBOW, Window Width, Vector Size
Word Groups:
Foods
Countries
Animals
Embedding Space

Figure 3.1: Training Arabic Word Embeddings

3) lemmatize the text to drop most affixes and preserve only the core idea of each Arabic word (lemmas). 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 Skip-gram algorithms are considered [17]. The defining difference between these methods is that CBOW trains embeddings by attempting to predict a word given its context while Skip-gram tries to predict a word's context given the word. The embedding dimensions considered are 100 and 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

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 3.1: Training Parameters

for context words. 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 3.1. 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 [18].

CHAPTER 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 [11, 15].

We also created semantic similarity tasks from a set of 1250 word pairs with similarity scores. Between 1 and 4 fluent Arabic speakers labeled each word pair with a similarity score between 0-1, where pairs with a score of 1 indicates that the words are extremely related. We distinguish three tasks, one with pairs created from 4 labels, on from pairs that have 2 or more labels, and one task consisting of all pairs given labels. To begin creating these tasks, we selected 1250 of the most common words in the Arabic Wikipedia dump [30] 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 [12, 4]. 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. 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 tasks, 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.

4.2 Analogy Task

The analogy task is a standard for evaluating word vectors first used by Mikolov et al. [17]. 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

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

Table 4.1: Analogy Examples

the Google Translate API to translate the 19544 English analogies to Arabic [12]. 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 4.1 for examples of these analogies in English taken from Mikolov's originating paper [17].

CHAPTER 5

WORD EMBEDDING EXPERIMENTS

In order to better understand the effects of different parameters on the quality of the final embeddings, we evaluate the multiple parameter combinations of each of the parameter selections outlined in Section 3. The text corpus for training the embeddings is an Arabic Wikipedia dump from `https://dumps.wikimedia.org/arwiki/20150901/` [30], 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 run 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 5.1 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 publicly available pre-trained model trained on a 100 billion word Google News Corpus [18]. The metric that we choose to base our evaluations on is the Spearman correlation between the model similarity estimates the evaluation task similarity values. Both models show a high correlation with the evaluation task scores. The Google News vectors display an impressive .6979 Spearman correlation score to the task,

Data Set	**Similarity Correlation**	**Analogy Accuracy**
Google News	0.6979	0.7359
5 Million	0.5458	0.0452

Table 5.1: English Baseline Results

providing a high score to aim for. We consider the 0.5458 correlation score of the default English task to be the baseline for our Arabic word embeddings because it is the same size training corpus as the Arabic one.

Table 5.2 shows the results of the models with the 10 highest correlation scores on the Word Similarity 353 task [11, 15]. 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 5.1 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 that performs as well as the English baseline, and outperforms it. This improvement can be considered even more significant due to the translation of the task, but it is difficult to quantify how much loss is occured from translation. Interestingly, the unprocessed base text scores higher than lemmatization on this task, possibly due to the comparitively simpler, noun dominated terms in the task. Lemmatization likely over simplifies the words, while in our similarity task, the higher complexity of the terms benefits more from lemmatization. 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 5.2: Top Results on Arabic Word Similarity 353

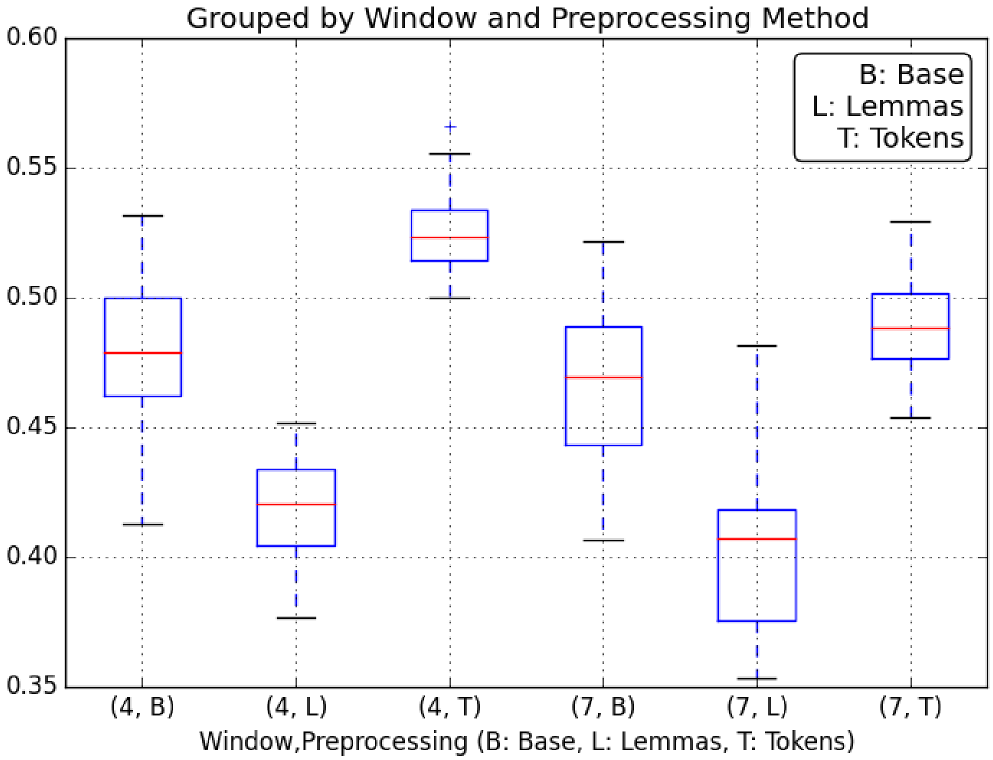

Figure 5.1: Results on Arabic Word Similarity 353

5.2 Our Similarity Task

Tables 5.3, 5.4, and 5.5 show the results of the models with the 10 highest correlation scores on the task we developed. Each table displays results on different subsets of our data, selected by the minimum number of fluent Arabic evaluators that contributed labels to the word pairs. These subsets are the collections where each word pair has at least one evaluator, at least two evaluators, and at least four evaluators (maximum is four). These results all demonstrate similar trends in Figures 5.2, 5.3, and 5.4. The Kendall Tau scores between these three rankings are .335 between the full set of word pairs and the set with four votes, .482 between the full set and the set with two or more votes, and .572 between the task with two votes and the task with four votes (all highly significant). Tables 5.3, 5.4, and 5.5 do demonstrate that with more evaluators, the models become better correlated with the task. Interestingly, the base Arabic without tokenization or lemmatization produces the highest correlation scores on our task. This observation, along with the improvements with more voters, suggests that as our task draws the most common words from the same Wikipedia corpus that the models are trained on, the embeddings are able to directly learn the semantic similarity of these base words without the aid of lemmatization or tokenization. As the word pair similarities become more correlated with the model as we add fluent evaluators, these results also suggest that the control model is able to predict a similarity score more consistently accurate than our human labelers, assuming that the similarity scores will converge to a true label as we add human labelers. Additionally, Figures 5.2, 5.3, and 5.4 support our findings that the smaller window size does indeed have a strong positive impact on the quality of the word embeddings.

These similarity task experiments have shown that certain preprocessing and training decisions can substantially change the performance of Arabic word embeddings on similarity tasks. Some choices did not demonstrate any significant impact on any of our results,

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 5.3: Top Results on our whole Arabic Task

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 5.4: Top Results on our Arabic Task - 2+ Votes

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 5.5: Top Results on our Arabic Task - 4 Votes

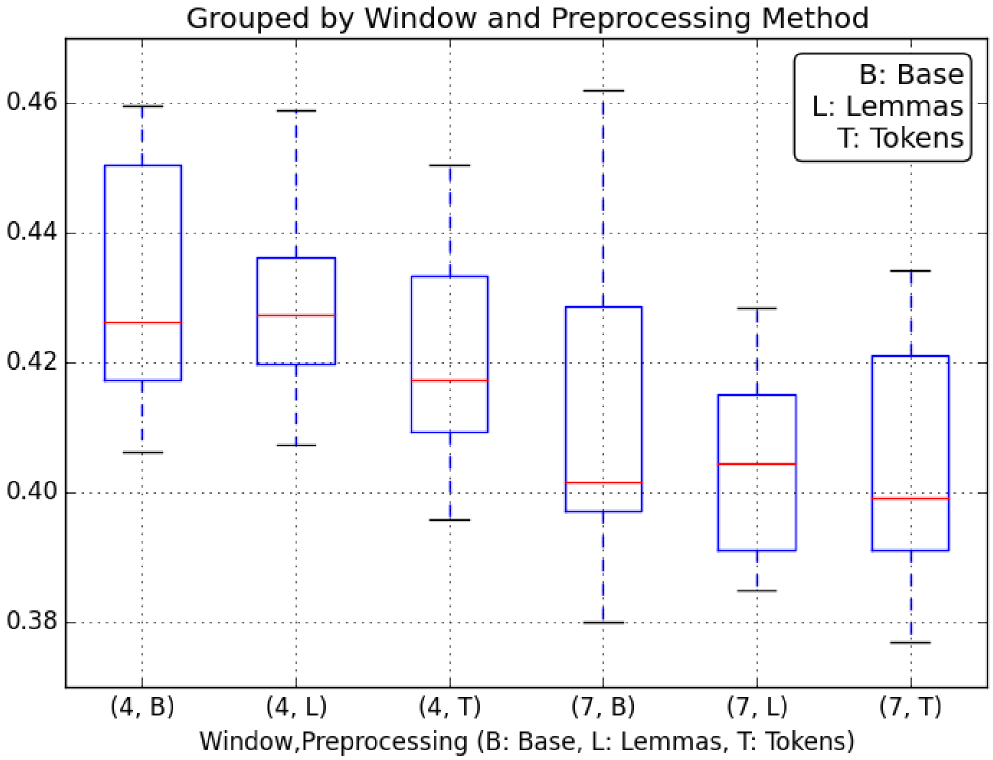

Figure 5.2: Results on Our Whole Arabic Task

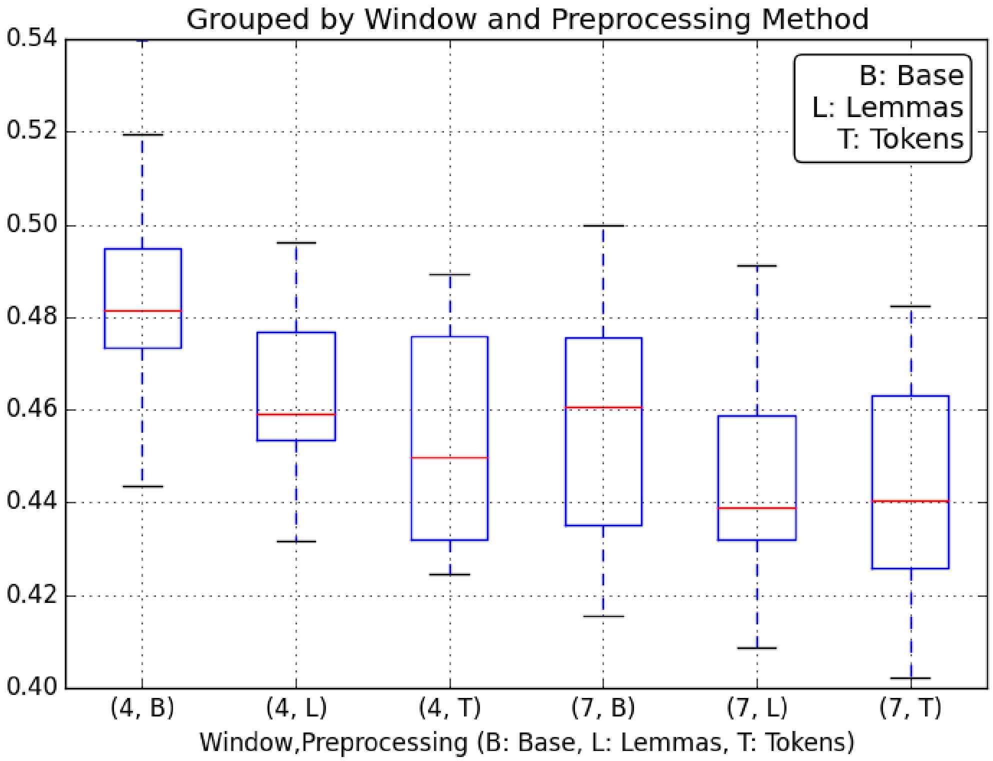

Figure 5.3: Results on Our Arabic Task - 2+ Votes

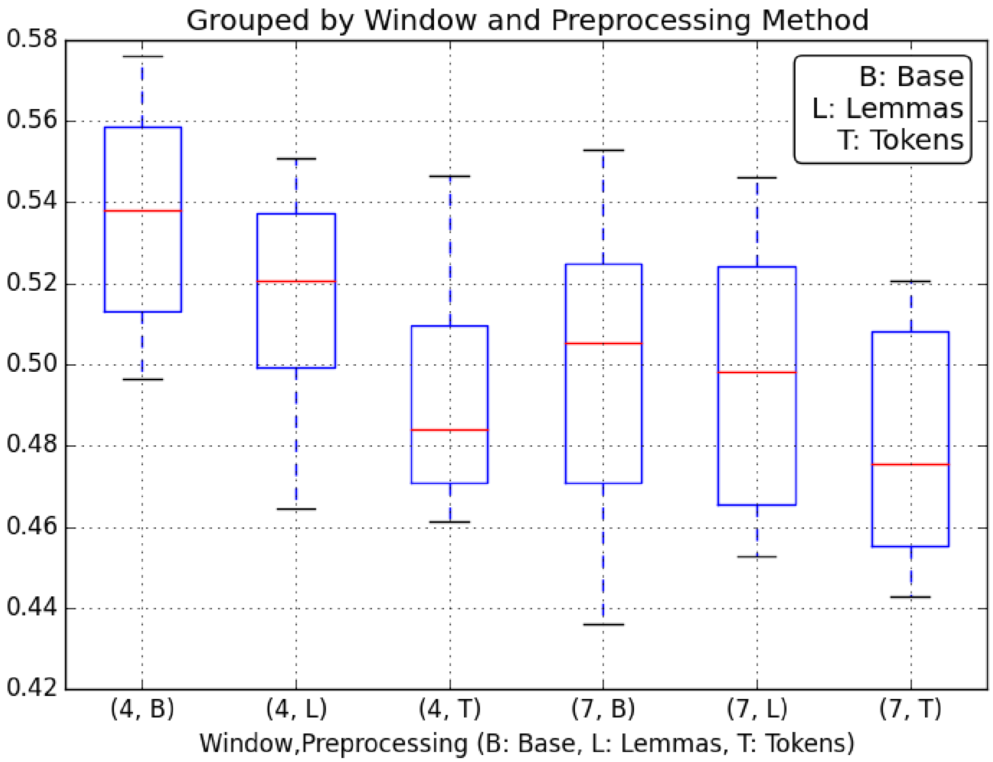

Figure 5.4: Results on Our Arabic Task - 4 Votes

Model	**Accuracy**
Google News	0.73587801882930826
5 Million	0.04522560880627239

Table 5.6: English Analogy Results

such as choosing between CBOW and Skip Gram algorithms. Some methods were 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, and our training set has approximately 5 million Arabic words while the Google News set has about 100 billion words [18].

5.3 Analogy Task

Table 5.6 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. We leave trying a larger model as future work.

Table 5.7 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 5.5 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,

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 5.7: Top Results on Arabic Analogy Task

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.

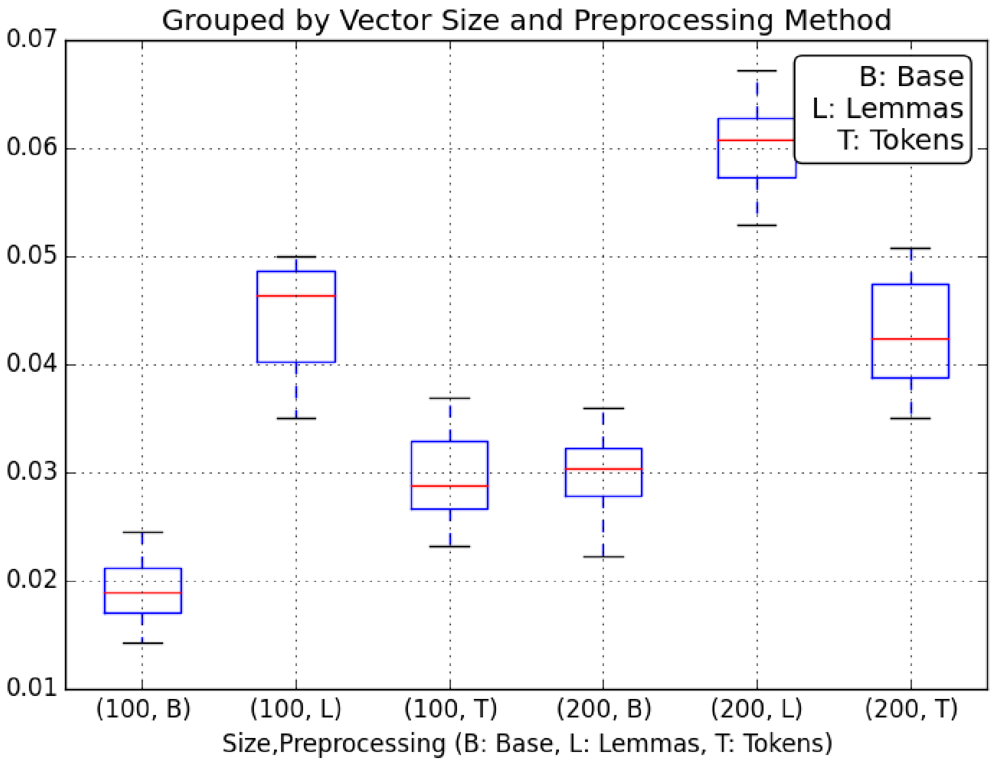

Figure 5.5: Arabic Analogy Task Results

Chapter 6

Recommendations for Arabic Word Embeddings

Our results have affirmed that there is no free lunch when choosing methods to train Arabic word embeddings. This is result supports the findings of Schnabel et al. when they investigated how to evaluate word embeddings and found that different evaluations provided useful but different results [27]. 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 is situationally useful, we suggest not normalizing either of these. 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.

CHAPTER 7

APPLYING WORD EMBEDDINGS TO BUZZ DETECTION

7.1 BUZZ DETECTION METHODOLOGY

The properties of high quality word embeddings can used in unique text mining applications. In this work we apply the best models from our training experiments to a buzz detection task, where we harness the power of Arabic word embeddings to demonstrate their power. We define buzz to be the level of discussion about a semantic topic in a set of documents. We define the buzz detection task as given a set of documents and a topic, accurately identify the level of buzz about the topic within them. Our goal in the buzz task is to detect buzz over a series of months in a set of Arabic news documents relevant to Iraq. We will detect buzz for each month over which we have data, measuring how much buzz about the topic of *violence* is in the news each month. Our ground truth for evaluation is the publicly available Iraq Death Count data from `https://www.iraqbodycount.org/database/` that aims to document how many violent deaths occur in Iraq on a monthly basis [14]. We will look at five methods to measure buzz to demonstrate the benefits of using Arabic word embeddings, each outlined in Table 7.1. We score the results from each method against the ground truth by finding the Spearman correlation between the method's scores and the ground truth counts. We propose that we can obtain improved results on the buzz detection task through the use of Arabic word embeddings. Below we outline three baseline buzz detection methods that do not use embeddings and two buzz detection methods that utilize Arabic word embeddings.

Type	**Target Words**	**Uses Embeddings**	**Weighted**
$frequency$	*violent, violence*	No	No
$domain$	*expert domain*	No	No
$synonym$	*expert domain + synonyms*	No	No
$similarity$	*expert domain + similar embeddings*	Yes	No
$weighted$	*expert domain + similar embeddings*	Yes	Yes

Table 7.1: Methods to Measure Buzz

Our three baselines for buzz detection are the frequency method, the domain method, and the synonym method. The frequency method is to simply count the number of times the Arabic words for violent or violence occur per month in the corpus, and normalize this score by the total number of words in the corpus per month. The domain method is the same as the frequency method, but we use a list of violence-related Arabic words instead of just violent and violence. This list of words in the violence domain is provided by experts on social science in the Middle East. The synonym method expands this domain list by adding 5 synonyms obtained from a thesaurus (through translation) for every expert provided word. When expanding words in this way, we only keep the intersection of each of these expansions leaving us with $|expandedset| \leq |originalset| * 6$. The similarity method expands the domain method by using the 5 most similar word embeddings to the word embedding for each word in the expert domain list. The weighted similarity method adds weights to each word in the similarity method by how similar it is to the word embedding for the original expert-provided word. Both methods that use word embeddings are evaluated with three different models to demonstrate differences between training methods. The three models are the top performing models on the Word Similarity 353 task, our similarity task, and the analogy task. This provides us with representation for each of the three preprocessing methods.

Method	Model	Buzz	Normalized Buzz
frequency	NA	-0.093631914230652966	-0.093631914230652966
domain	NA	0.060914562733731664	0.10179965592643478
similarity	control	0.060914562733731664	0.10179965592643478
similarity	tokens	0.060914562733731664	0.10179965592643478
similarity	lemmas	0.060914562733731664	0.10179965592643478
weighted similarity	control	0.060914562733731664	0.10179965592643478
weighted similarity	tokens	0.060914562733731664	0.10179965592643478

Table 7.2: Correlation Scores Between Method Scores and Ground Truth

7.2 Buzz Detection Experiments

Table 7.2 shows the correlations between each of our buzz methods and the ground truth values. The results shown below are bad and led us to discover that the corpus did not overlap the ground truth correctly. Soon to be replaced with appropriate data from new experiments on a better corpus.

Here we can see that any method utilizing the expert domain list outperforms the base frequency method by the same amount. This is only because the corpus had only a few occurances of the target words in it.

CHAPTER 8

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 also hope to increase the utilization of Arabic word embeddings within the research community with this software. 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 [25, 22, 12, 4].

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`.

Chapter 9

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.

BIBLIOGRAPHY

[1] Mohamed Attia Mohamed Elaraby Ahmed. *ALarge-SCALE COMPUTATIONAL PROCESSOR OF THE ARABIC MORPHOLOGY, AND APPLICATIONS*. PhD thesis, Faculty of Engineering, Cairo University Giza, Egypt, 2000.

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

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

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

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

[6] Mona Diab. 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, 2009.

[7] Brian Dickinson and Wei Hu. Sentiment analysis of investor opinions on twitter. *Social Networking*, 4(03):62, 2015.

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

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

[10] Manaal Faruqui and Chris Dyer. 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 2014. Association for Computational Linguistics.

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

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

[13] Nizar Habash, Owen Rambow, and Ryan Roth. 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, 2009.

[14] John Sloboda Hamit Dardagan. Iraq body count. `https://www.iraqbodycount.org/database/`, April 2016. (Accessed on 04/27/2016).

[15] Samer Hassan and Rada Mihalcea. 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, 2009.

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

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

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

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

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

[21] NPR. 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 2010. (Accessed on 04/04/2016).

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

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

[24] Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. Learning causality for news events prediction. In *Proceedings of the 21st international conference on World Wide Web*, pages 909–918. ACM, 2012.

[25] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[26] Navid Rekabsaz. On using statistical semantic on domain specific information retrieval.

[27] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *Proc. of EMNLP*. Citeseer, 2015.

[28] Lameen Souag. 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 2013. (Accessed on 04/04/2016).

[29] Yifang Wei, Lisa Singh, Brian J. Gallagher, and David Butler. Overlapping target event and storyline detection of online newspaper articles. Unpublished Technical Paper, April 2016.

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

[31] Dongwen Zhang, Hua Xu, Zengcai Su, and Yunfeng Xu. Chinese comments sentiment classification based on word2vec and svm perf. *Expert Systems with Applications*, 42(4):1857–1863, 2015.

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