Optimized Training and Evaluation of Arabic Word Embeddings

Jordan King, Lisa Singh, Eric Wang

Abstract

Word embeddings are an increasingly important tool for NLP tasks, especially those that require semantic understanding of words. Methodologies and properties of English word embeddings have been extensively researched, however little attention has been given to the production and application of Arabic word embeddings. Arabic is far more morphologically complex than English due to the many conjugations, suffixes, articles, and other grammar constructs. 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 and a part of speech tagging task, we were 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. 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 operations to investigate the relationships between words. A few of the possible operations are measuring the similarity of two words, identifying which word from a set is least similar, and solving basic analogies. The classic demonstration of word embeddings is to take (the embeddings of) *king*, subtract *man*, and add *woman*. The resulting embedding is closest to the embedding for *queen*! Intuitively, this allows us to subtract the male gender meaning from 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 ignoring some words belonging to a topic. An example application would be to use the embeddings to learn what words are highly similar to words similar to fear (in Arabic), and then compute the degree to which some media is using fearful language in the context of a threatened city. This information could help us understand when people feel forced to migrate from the city.

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.

2. RELATED LITERATURE

Word embeddings have gained popularity over the past few years since Mikolov et al. published the word2vec algorithms in 2014 [11, 12]. 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. [4]. Al-Rfou et al. computed word embeddings for 100 languages using Wikipedia articles [1]. This work inspired our system of semantic and syntactic evaluation, but 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. Zirikly et al. utilized Arabic word vectors to improve named-entity recognition

*ODDx*3*'15,* August 10th, 2015, Sydney, AU.
Copyright 2015 ACM 978-1-4503-3831-8 ...$15.00.

performance, normalizing hamzas, elongated words, and number normalization [15]. 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 [2]. 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 [8]. Diab published the Amira software as fast and robust option for phrase chunking and POS tagging [3]. 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 [14]. 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. [5]. However, few of these are available in Arabic. Faruqui refers to two data sets that have been translated to Arabic by Hassan et al. [9], the 353 word WordSimilarity-353 and the 30 word Miller-Charles datasets [6, 13]. however this translation was done by a single Arabic speaker using the English semantic similarity scores [9]. 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. EXPERIMENTS

We performed a broad parameter sweep over various preprocessing techniques and word2vec parameterizations to determine the optimal word embedding methods. To evaluate the methods of training word embeddings, we required a method of measuring both semantic and syntactic accuracy. For the semantic similarity, we chose to use a semantic similarity task. As we were performing a large programmatic parameter sweep, we desired a large semantic similarity task to evaluate the Arabic word embeddings. The largest Arabic semantic similarity task is a manually translated version of the WordSimilarity-353 task [6, 9]. As we wanted a larger list entirely generated in Arabic and evaluated by multiple Arabic speakers, we created a list of 1000 similarity scores for given Arabic word pairs using fluent Arabic speakers. We evaluated all training parameterizations by using the embeddings from a given parameterization to obtain a similarity score to evaluate against the task. Additionally, we wanted to measure how different preprocessing techniques preserved the embeddings' ability to capture syntactic information. We evaluated the embeddings' ability to capture syntactic properties of words using a part of speech tagging task, using training and testing lables generated by Madamira [14]. The text corpus for training the embeddings is a cleaned Arabic Wikipedia dump. Each part of this experiment is described in the following subsections.

3.1 Semantic Understanding Evaluation

The semantic similarity task consists of 1000 Arabic word pairs and a similarity score in the range 0-10, where 10 represents words that are extremely related. As no task existed of the size that we required for our parameter sweep, we developed this task ourselves. The words began by selecting 1250 of the most common words from the Arabic Wikipedia, excluding words that occur in more than 5% of sentences. These words word then translated into English with Google translate [7], queried against the big huge thesaurus API for either synonyms or antonyms, and translated back to Arabic [10]. The original word and the resulting word are then paired up. One half of the pairs are synonyms, one quarter are antonyms, and one quarter are shuffled with other pairs to be randomly matched. The proportions were chosen because the synonyms database is more extensive than the antonym database. The various APIs involved introduce a large amount of noise, to the point that some synonym pairs will be completely unrelated Arabic words. We take advantage of this noise to distribute the relatedness of words across the 0-10 scale, while ensuring some pairs are related.

This list of word pairs was then given to 10 fluent Arabic speakers, along with simple instructions to evaluate the relatedness of the words on a scale of 1 to 5. The values that they provided were then averaged and scaled to 0 to 10. When evaluating a parameterization, we performed the same preprocessing on the word list as we did to the corpus prior to training. Each word pair's embeddings were compared for a similarity score, and the parameterization recieved a score for its squared error off the task's similarity score.

3.2 Syntactic Understanding Evaluation

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

3.3 Preprocessing Options

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

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

3.4 Parameterizations

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

3.5 Implementation

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

3.6 Results

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

4. ARAPY

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

5. CONCLUSION

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

References

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

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

[3] M. 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.

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

[5] M. Faruqui and C. 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.

[6] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

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

[8] N. Habash, O. Rambow, and R. 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.

[9] S. Hassan and R. 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.

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

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

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

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

[14] A. Pasha, M. Al-Badrashiny, M. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. 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.

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