Word Embeddings for Arabic Sentiment Analysis

A. Aziz Altowayan
Computer Science Department
Pace University
New York, USA
Email: aa10212w@pace.edu

Lixin Tao
Computer Science Department
Pace University
New York, USA
Email: ltao@pace.edu

***Abstract*— Manual feature extraction is a challenging and time consuming task, especially in a Morphologically Rich Language (MRL) such as Arabic. In this paper, we rely on word embeddings as the main source of features for opinion mining in Arabic text such as tweets, consumer reviews, and news articles. First, we compile a large Arabic corpus from various sources to learn word representations. Second, we train and generate word vectors (embeddings) from the corpus. Third, we use the embeddings in our feature representation for training several binary classifiers to detect subjectivity and sentiment in both Standard Arabic and Dialectal Arabic. We compare our results with other methods in literature; our approach—with no hand-crafted features—achieves a slightly better accuracy than the top hand-crafted methods. To reproduce our results and for further work, we publish the data and code used in our experiments[1].**

***Keywords*-arabic sentiment; word embeddings.**

I. Introduction

Sentiment analysis is a very common task in Natural Language Processing (NLP), where the goal is to determine the attitude or feeling conveyed in some text. With the surge in microblogging and similar services, opinionated posts and texts are flooding the Internet. Hence, sentiment analysis has gained much attention and is in demand for many applications (e.g. business analytics) because of its simplicity and efficiency. Various products are being developed around users' opinions ranging from consumer reviews to reactions surrounding political events.

Although English has been the target language in most sentiment analysis research, recent efforts extend the focus to other languages such as Arabic. Basic machine learning techniques—as simple as Naïve Bayes—have been used to achieve baseline results [1], [2]. However, these systems require lots of feature engineering work prior to applying any machine learning method.

Most Arabic sentiment analysis systems still rely on costly hand-crafted features, where features representation requires manual pre-processing in order to obtain the preferred accuracy. For example, Mourad and Darwish (2013) report that *POS tagging* and *word stemming* have major effect in improving their sentiment classification result. Morphology-based features have, also, been shown to improve the system's performance [3]. A manually-prepared list of emotion words *"polarity lexicon"* is another requirement in such systems. And not to mention the efforts spent on handling random sentence structures in dialectal Arabic.

In this work, we present neural word embeddings as an alternative for such hand-crafted features in Arabic sentiment analysis. We embed Arabic words in a continuous vector space. This allows us to represent sentiment features as dense vectors instead of the conventional sparse representations. We consider the sentiment task as a standard binary classification problem, thus we discriminate only between either positive/negative or subjective/neutral sentiment. The contributions of this paper include:

- Employing distributed word representations to embed sentiment features as dense vectors in Arabic sentiment analysis; where we achieve a significant performance.
- An Arabic corpus that we have built carefully from various text collections.
- A pre-trained Arabic word embeddings generated from the aforementioned corpus.
- A labeled (positive/negative) Arabic twitter dataset that we gathered from several published datasets and refined them into a larger dataset.

II. Related Work

We are not aware of a published work that utilize embeddings in a sentiment (or classification) task specifically for Arabic text. However, research on Arabic sentiment analysis is getting more attention from the research community recently. The majority of the existing work rely on manually engineered features in their classification. One of the most prominent features is the existence of certain words in a sentiment lexicon. Lexical-based features have been heavily exploited in almost all reviewed work of Arabic sentiment analysis [3], [4], [2], [5], [6].

In [3], they used both morphology-based and lexical features for subjectivity and sentiment classification of Arabic. Where POS tagging and stemming features were the key strength for [2] to achieves a remarkable performance on the subjectivity of Standard Arabic. As well as a whole corpus of labeled emotion words of Standard Arabic was presented in [7] for subjectivity and sentiment analysis.

[1]code and data available at: https://github.com/iamaziz/ar-embeddings

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

On the other hand, a set of open issues and difficulties in Arabic sentiment analysis has been surveyed in [8].

III. Word Embeddings

Semantics representation is a challenging task in natural language processing. Nonetheless, with the recent advancement in neural word representations models [9], [10], [11], [12], word embedding[2] has emerged as the main spectrum for *distributional semantic models*. For the first time, distributed representations of words make it possible to capture words semantics; even the shift in meaning of words over time [14]. Such capability explains the recent successful switch in the NLP field from linear models over sparse inputs[3], e.g. support vectors machines and logistic regression, to non-linear neural-network models over dense inputs [16]. Consequently, systems that rely on word embedding have been very successful in recent years, across a variety of NLP tasks [17].

Neural word embeddings are prediction-based models. In other words, in order to come up with distributed representations for words, the network learns its parameters by predicting the correct word (or its context) in a text window over the training corpus.

While the point of network training is to learn good parameters, word vector representations follow the notion that similar words are closer together[4]. In linguistics, this is known as "*Distributional Hypothesis*"[5]. This underlying idea is beneficial for extracting features from text represented in such way; especially for understanding the context of word use in sentiment analysis. Since this notion is viable for any natural language, we take advantage of that and apply it to Arabic. Next we describe our approach for collecting an Arabic corpus and generating word vectors from the corpus.

A. Building and Preprocessing Corpus

We build a corpus from a set of publicly available text collections. Text contents are mainly news articles based on a local Arabic newspaper [18] and Arabic editions of international news networks[6] [19]. To enrich the corpus with dialectal vocabulary, we also include a pool of around 63 thousand consumer reviews [20] , which include a mixture of different spoken Arabic. Further, as observed by [2], some social media users may tend to convey their feelings through using some verses from the holy Quran. Therefore, we include the complete text of Quran[7] in our corpus as to account for such sentiments. The complete corpus contains around 190 million words. Table I shows the corpus details with sources.

Table I
Corpus collections and sources

source	word count
Quran-text	751,291
watan-2004	$\sim$ 106 million
cnn-arabic	$\sim$ 24 million
bbc-arabic	$\sim$ 20 million
consumer reviews	$\sim$ 40 million

Next, we need to ensure the formatting of the sentences and words before we generate the embedding. Although most of the text collections are prepared properly since they come from published work, we notice few irregularities in some chunks of the text such as non-arabic letters and misplaced punctuations (e.g a comma immediately preceded a word as in "الوظيفية،"). So we perform further preprocessing (e.g. extracting a sentence tokens then re-joining them) on the complete merged text. Python's NLTK[8] was our perfect assist for sequencing the text. Though we ran into some encoding issues[9] that were eventually cleared.

B. Generating Word Vectors

There are several models available for learning word embeddings from raw text. Among these are GloVe [12] and dependency-based word embeddings[10] [21]. Our choice, however, is the well known and widely used word2vec model [9], [10]. Word2vec describes two architectures for computing continuous vectors representations, the skip-gram and Continuous Bag-Of-Words (CBOW). The former predicts the context-words from a given source word, while the latter does the inverse and predicts a word given its context window. We use CBOW to learn the embeddings; since it is simpler, computationally-efficient, and suitable for larger datasets [22].

1) Embeddings experiment setup: We are not sure about a good choice of hyper-parameters, so we ran several experiments with different settings. With each run, we would apply the generated embeddings to the sentiment classifiers (a sort of brute-force) to test performance. We tried arbitrary choices for window size (5, 10, 15, and 20) and embedding dimensions (200, 300, 500, 700). Based on the classifiers' performance, window size 10 with 300 dimensions seems to be our best choice for learning good embeddings. Since the average size of each tweet in our dataset is 8.5 words, a window size of 10 would make sense.

From the available implementations, we experimented on both the original C implementation of word2vec toolkit[11]

[2]Originally introduced in 2003 [13].

[3]Although, a very recent work shows "re-discovers" the effectiveness of linear models through proposing a simple and efficient linear model for text classification and representation learning [15].

[4]"You shall know a word by the company it keeps." (Firth, J. R. 1957).

[5]"Deep Learning for NLP" by R. Socher, 2016.

[6]Such as CNN and BBC.

[7]Quran text source: http://tanzil.net (we used the simple-text version).

[8]http://www.nltk.org/api/nltk.tokenize

[9]Due to the difference between Python2 and Python3 in handling character unicode.

[10]A modified version of word2vec

[11]https://code.google.com/archive/p/word2vec/

and Python's gensim[12]. Training time takes around 102 minutes on a machine with 3.1GHz CPU and 16GB of RAM. From around 190 million words in our training corpus, the learned embedding size is 159,175 vocabulary.

2) Evaluating the embeddings: For English embeddings, vectors quality can be evaluated using a test set that comprises around 20,000 semantic and syntactic questions. However, for Arabic, we are not aware of any evaluation method. The actual quality measure, as we mentioned earlier, would be the classifiers' performance down the road. We create multiple similarity and analogy queries to see how reasonable our embeddings are. Table II shows examples of the performed semantic analogy queries; while in Table III, we show examples of sentiment-related results of the embeddings.

Table II
SAMPLE ANALOGY QUERIES RESULTS ON THE EMBEDDINGS (NOTE: SAUDI ARABIA IN ARABIC IS ONE WORD NOT A PHRASE).

semantic analogy query (English translation)	**results - top 3**	**similarity**
السعودية الرياض اليابان (saudi_arabia riyadh japan)	طوكيو (tokyo)	0.53
	سنغافورة (singapore)	0.45
	بكين (beijing)	0.44
ملك رجل ملكة (king man queen)	متزوجة (a married woman)	0.35
	لرجل (to a man)	0.33
	امرأة (woman)	0.31

Table III
SAMPLE WORD SIMILARITY RESULTS FOR SENTIMENT-RELATED VOCABULARY IN STANDARD ARABIC.

query term (transliteration)	**top 5 results** (transliteration)	**distance**
جيد (jyd)	ممتاز (mmtAz) excellent	0.62
meaning: good	مميز (mmyz) special	0.51
	جميل (jmyl) beautiful	0.49
	مناسب (mnAsb) suitable	0.48
	كبير (kbyr) great	0.48
سخيف (sxyf)	تافه (tAfh) trivial	0.49
meaning: silly	سطحي (sTHy) shallow	0.44
	سخيفة (sxyfp) silly/feminine	0.44
	مبتذل (mbt*l) vulgar	0.43
	سيء (sy') bad	0.43

Dealing with dialectal Arabic is difficult. One of the problems is the use of different spellings for the same word. For example, some users may violate spelling rules of a certain word by adding or omitting letters. Fortunately, word embeddings work well in this situation. Since the different (misspelled) word variations are mostly used in the same context, they will have similar word vectors. Thus, we know they are semantically the same even though we actually did not specifically handle these violations. In our approach, we take this issue into account; that is why we include dialect text (the consumer reviews) in the training corpus. This will save a lot of efforts spent during preprocessing to handle such variations. Table IV shows similarity results of two words in dialectal Arabic. We include the transliteration[13] in the table to show how each word is written differently.

Table IV
SAMPLE WORD SIMILARITY RESULTS, WORD EMBEDDING CAN HANDLE MIS-SPELLED VARIATIONS OF THE SAME WORD IN DIALECT ARABIC.

query term (transliteration)	**top 3 results** (transliteration)	**distance**
كتير (ktyr)	كتيرة (ktyrp)	0.68
meaning: numerous	كتيير (ktyyr)	0.62
	كتيره (ktyrh)	0.59
رائع (rAAA}E)	رائع (rA}E)	0.62
meaning: splendid	رااائع (rAAAAA}E)	0.56
	رآئع (r—}E)	0.55

IV. SENTIMENT AND SUBJECTIVITY ANALYSIS

A. Datasets

We use three different datasets in our classification experiments. For the sentiment classification, we use twitter and book reviews datasets; we assume these two as "dialectal Arabic" text. Whereas for subjectivity classification we use news articles dataset, which is "Standard Arabic" text. The book reviews is scraped by [20] from a community driven website[14]. While for news articles dataset, we use the labeled news articles of [1] which is based on an automatically translated Arabic version of MPQA corpus[15].

Table V
OUR COLLECTION OF TWITTER DATASETS AND THE SOURCE OF EACH.

Dataset Source	positive	negative	total
ASTD	777	812	1589
ArTwitter	993	958	1951
QCRI	377	377	754
TOTAL	2147	2147	4294

For twitter dataset, we were not able to acquire a sufficient number of labeled tweets. So, we rely on previously published datasets in [23], [5], [2] to compile a relatively larger set of tweets. To easily distinguish these datasets, we refer to them as ASTD, ArTwitter, and QCRI respectively. Originally, ASTD tweets were grouped in four categories (*positive, negative, neutral*, and *objective*); however, we sampled only from the positive and negative tweets. Where QCRI set comprised seven categories, again we only consider the positive and negative tweets. For better training, we balanced

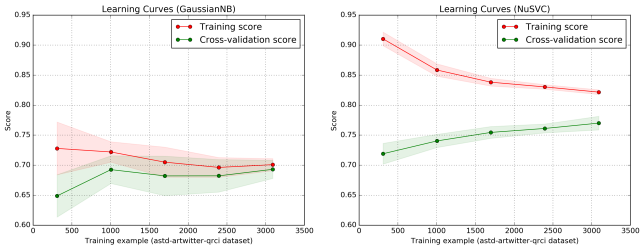
[12]https://radimrehurek.com/gensim/models/word2vec (Note: gensim library is a bit more sensitive to Arabic encoding, and it requires sequencing the input text.)

[13]Transliteration is based on Buckwalter.s See: http://www.qamus.org/transliteration.htm

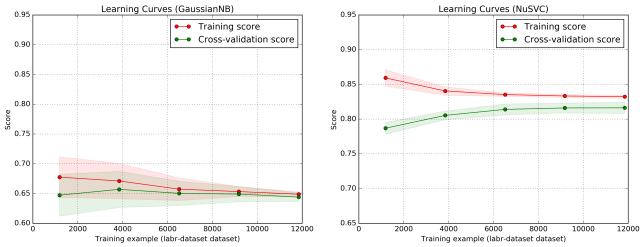
[14]http://www.goodreads.com

[15]Multi-Perspective Question Answering corpus. Based originally on English news articles. Arabic translation is downloadable at: http://lit.csci.unt.edu/research/downloads

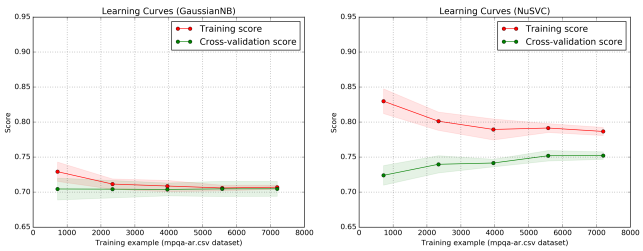
the selected samples. Table V shows number of samples we used from each dataset. After being combined together, we preprocess all tweets to remove non-arabic letters (e.g. handle names and mentions) and special characters (e.g. urls), we preserve the hashtags and emoticons however. The average number of tokens (words) in each tweet is 8.5.



(a) Sentiment of ASTD-ArabicTwitter-QRCI dataset



(b) Sentiment of LABR book review dataset



(c) Subjectivity of MPQA Arabic dataset

Figure 1. Learning curves of NaïveBayes and SupportVectors on Sentiment and Subjectivity datasets.

B. Training Classifiers

We consider the sentiment task as a standard binary classification. Thus, we run our experiments on six different binary classifiers with the three datasets. Table VII include the full list of the classifiers with the performance of each. We use the implementations of [24] in all our classification experiments. We run all classifiers under same training conditions and we perform a 10-fold cross-validation with data split 90% train and 10% test. Classifiers performance will eventually determine the quality of our word representations. Although we do not attempt to tune classifiers for better results, SVM classifier and logistic regression classifier performed better than others in most our of baseline results.

1) Features representation: In one way or another, methods that use manually-extracted features rely on sparse representations; in which each feature (e.g. words or POS tags) is its own unique dimension. In contrary to that, we represent each feature as a fixed-size dense vector. For each input sample, we obtain its feature vector by averaging the retrieved embeddings of that sample. Depending on the quality of the embeddings, similar features will end up having similar vectors. And here where the "powerful" key idea of the dense representations comes, generalization [16]. This enables our model to map (represent) unseen samples to similar feature vectors of those in the training set.

Table VI
METHODS PERFORMANCE COMPARED ON THE MPQA SUBJECTIVITY OF STANDARD ARABIC.

Method	Prec.	Rec.	F-Score	MAcc
Banea et al. [1]	72.30%	71.13%	71.30%	72.22%
Mourad et al. [2]	78.10%	**75.70%**	76.05%	77.20%
Our word embedding method	**79.95%**	72.67%	**76.14%**	**77.87%**

C. Results

Since our main focus is on the quality of the embeddings, we do not attempt to tune the classifiers parameters. Hence, we run classifiers with the simplest configurations possible (i.e. default parameters). Figure 1 shows a comparison between the training of Naïve Bayes classifier and SVM classifier over the three datasets. From the learning curves, we can clearly notice that NB classifier will not benefit from adding more training data; since it converges quickly even with the small twitter dataset. Whereas in the case of SVM classifier, as we see in fig. 1 (a), there is a room to further improve our results by increasing the size of twitter dataset.

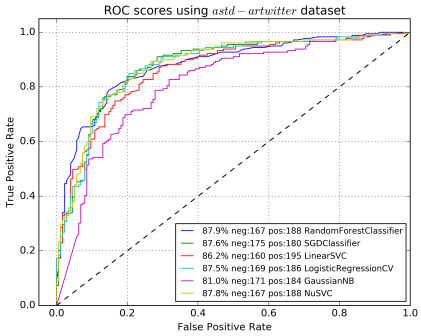
The detailed performance of all classifiers on each of the three datasets are reported in Table VII and figure 2. For the purpose of method comparison, we follow the conventions in reporting our classifiers results. We use the measures recall, precision, F-measure, and macro-accuracy which correspond to Rec., Prec., F1, and MAcc respectively.

1) Comparison with other methods: For a fair comparison, we use the same dataset[16] and task reported in the papers that we compare our work with. Table VI shows our results on the subjectivity classification task compared to top reported work [1], [2] on the Standard Arabic dataset MPQA. Both [1] and [2] use hand-crafted features to achieve their results. The improvement in the latter method is attributed to POS tagging and word stemming they use. As we see in the table, our method achieves a slightly better accuracy than both of the other two. Though, we think, we still could improve upon our result should we use a larger training dataset, as we see from the learning curve of SVM in fig. 1 (c).

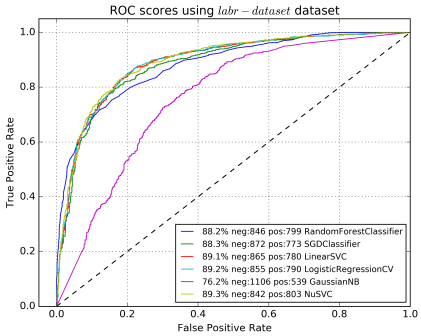
[16]For our comparison, we were able to get MPQA dataset only. Unfortunately, we could not obtain the other dataset "ArabSenti"; although we have contacted the original authors.

Table VII
CLASSIFIERS AND THEIR SCORES ON EACH DATASET. (NOTE: REC., PREC., AND MACC. SCORES ARE THE AVERAGE OF BOTH POSITIVE AND NEGATIVE CLASSES).

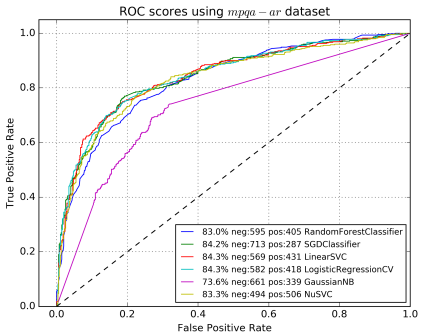
		Classifier					
Dataset	Measure	LinearSVC	Rnd.Forest	GaussianNB	NuSVC	Log.Reg.	SGDClassifier
ASTD-ArTwitter-QRCI (Sentiment)	Rec.	74.19%	71.43%	58.53%	76.50%	77.42%	75.58%
	Prec.	82.14%	75.98%	76.97%	83.00%	81.16%	82.00%
	F1	77.97%	73.63%	66.49%	79.62%	79.25%	78.66%
	MAcc.	78.80%	74.17%	69.86%	80.21%	80.21%	79.53%
LABR-book-reviews (Sentiment)	Rec.	81.48%	80.23%	50.06%	81.73%	82.60%	88.74%
	Prec.	80.27%	79.04%	71.17%	80.82%	80.59%	73.17%
	F1	80.87%	79.63%	58.78%	81.27%	81.58%	80.20%
	MAcc.	81.27%	80.05%	64.85%	81.69%	81.88%	78.60%
MPQA-Arabic (Subjectivity)	Rec.	72.67%	70.28%	58.57%	72.02%	77.87%	81.13%
	Prec.	79.95%	77.70%	75.42%	75.42%	78.30%	71.80%
	F1	76.14%	73.80%	65.93%	75.03%	74.71%	75.40%
	MAcc.	77.87%	76.65%	71.16%	77.60%	75.66%	75.60%



(a) ASTD-ArTwitter tweets



(b) LABR book reviews



(c) MPQA Arabic articles

Figure 2. Classifiers ROC on each dataset.

V. CONCLUSION

We have introduced neural word embeddings as an alternative of hand-crafted features for Arabic sentiment analysis. We exploit state-of-the-art word representations method to compute continuous vector representations of Arabic words. For the purpose of this work, we have built a large Arabic corpus to generate word representations. In our classification experiments, we relied solely on the dense representations of our embedding as the source of features; and yet we achieve a significant performance in compare to existing techniques. Our results showed that such a simple yet powerful method is enough to achieve state of the art performance. This should encourage research in the application of Arabic sentiment analysis to adapt more future-promising techniques. To further contribute to research in this area, we release our code and data used in our experiments.

In future work, we hope to compare the impact of different embedding methods (e.g. GloVe and skip-gram) on the performance of the classifiers. We also hope to see some clear and specific techniques for evaluating the quality of the embeddings.

REFERENCES

[1] C. Banea, R. Mihalcea, and J. Wiebe, "Multilingual subjectivity: are more languages better?" in *COLING '10: Proceedings of the 23rd International Conference on Computational Linguistics*, University of North Texas. Association for Computational Linguistics, Aug. 2010, pp. 28–36.

[2] A. Mourad and K. Darwish, "Subjectivity and sentiment analysis of modern standard arabic and arabic microblogs," in *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis*, 2013, pp. 55–64.

[3] M. Abdul-Mageed, M. T. Diab, and M. Korayem, "Subjectivity and sentiment analysis of modern standard Arabic," in *HLT '11: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language*

Technologies: short papers - Volume 2, Columbia University. Association for Computational Linguistics, Jun. 2011, pp. 587–591.

[4] M. Abdul-Mageed, S. Kübler, and M. Diab, "SAMAR: a system for subjectivity and sentiment analysis of Arabic social media," in *WASSA '12: Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, Columbia University. Association for Computational Linguistics, Jul. 2012.

[5] N. A. Abdulla, N. A. Ahmed, M. A. Shehab, and M. Al-Ayyoub, "Arabic sentiment analysis: Lexicon-based and corpus-based," in *Applied Electrical Engineering and Computing Technologies (AEECT), 2013 IEEE Jordan Conference on*, Dec 2013, pp. 1–6.

[6] R. Bouchlaghem, A. Elkhelifi, and R. Faiz, "A Machine Learning Approach For Classifying Sentiments in Arabic tweets," in *WIMS '16: Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics*. ACM, Jun. 2016.

[7] M. Abdul-Mageed and M. T. Diab, "Awatif: A multi-genre corpus for modern standard arabic subjectivity and sentiment analysis." in *LREC*, 2012, pp. 3907–3914.

[8] S. R. El-Beltagy and A. Ali, "Open issues in the sentiment analysis of Arabic social media: A case study," in *2013 9th International Conference on Innovations in Information Technology (IIT)*. IEEE, 2013, pp. 215–220.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv.org*, Jan. 2013.

[10] T. Mikolov, I. Sutskever, K. C. 0010, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality." *AAAI Spring Symposium AI Technologies for Homeland Security 200591-98*, vol. cs.CL, pp. 3111–3119, 2013.

[11] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents." *AAAI Spring Symposium AI Technologies for Homeland Security 200591-98*, vol. cs.CL, pp. 1188–1196, 2014.

[12] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation." *EMNLP*, pp. 1532–1543, 2014.

[13] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[14] C. D. Manning, "Computational Linguistics and Deep Learning." *COLING*, vol. 41, no. 4, pp. 701–707, 2015.

[15] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," *arXiv.org*, Jul. 2016.

[16] Y. Goldberg, "A Primer on Neural Network Models for Natural Language Processing," *arXiv.org*, Oct. 2015.

[17] T. Luong, R. Socher, and C. D. Manning, "Better word representations with recursive neural networks for morphology." in *CoNLL*, 2013, pp. 104–113.

[18] M. Abbas, K. Smaïli, and D. Berkani, "Evaluation of topic identification methods on arabic corpora." *JDIM*, vol. 9, no. 5, pp. 185–192, 2011.

[19] M. K. Saad and W. Ashour, "Osac: Open source arabic corpora," in *6th ArchEng Int. Symposiums, EEECS*, vol. 10, 2010.

[20] M. A. Aly and A. F. Atiya, "LABR: A Large Scale Arabic Book Reviews Dataset." in *ACL (2)*, 2013, pp. 494–498.

[21] O. Levy and Y. Goldberg, "Dependency-Based Word Embeddings." *ACL*, pp. 302–308, 2014.

[22] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors." *ACL (1)*, 2014.

[23] M. Nabil, M. Aly, and A. F. Atiya, "ASTD: Arabic Sentiment Tweets Dataset," pp. 2515–2519, 2015.

[24] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.