Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 2590–2595
Marseille, 11–16 May 2020

Morphology-rich Alphasyllabary Embeddings

Amanuel Mersha, Stephen Wu
Addis Ababa University, UT Health Science Center at Houston
amanuel.negash@aau.edu.et, wu.stephen.t@gmail.com

Abstract

Word embeddings have been successfully trained in many languages. However, both intrinsic and extrinsic metrics are variable across languages, especially for languages that depart significantly from English in morphology and orthography. This study focuses on building a word embedding model suitable for the Semitic language of Amharic (Ethiopia), which is both morphologically rich and written as an alphasyllabary (abugida) rather than an alphabet. We compare embeddings from tailored neural models, simple pre-processing steps, off-the-shelf baselines, and parallel tasks on a better-resourced Semitic language – Arabic. Experiments show our model's performance on word analogy tasks, illustrating the divergent objectives of morphological vs. semantic analogies.

Keywords: word embeddings, word2vec, Amharic, morphologically rich, alphasyllabary

1. Introduction

Word embeddings are ubiquitous in today's language-related applications. A long history of distributional semantics efforts (Deerwester et al., 1990; Blei et al., 2003; Padó and Lapata, 2007) have largely given way to efficient neural network-inspired word embedding methods (Mikolov et al., 2013; Pennington et al., 2014), which have been applied across dozens of tasks with great success. Word embeddings have also been studied in bilingual (Zou et al., 2013) and multilingual (Ammar et al., 2016; Bojanowski et al., 2017) settings, but evaluations in lesser-resourced languages have been cursory and highly variable.

This work focuses on word embeddings in the Amharic language, the second-most-spoken Semitic language[1] (Eberhard et al., 2020). Similar to other Semitic languages like Arabic or Hebrew, it is morphologically rich, with templatic morphology that is fusional rather than agglutinative, so that morphemes are difficult to separate:

አይመለስም	**እመለሳለሁ**
`almelesem`	`imelesalehu`
return.1sg.neg.past	*return.1sg.future*

Amharic is written with the Fidel alphasyllabary (also known as an abugida). Each character in Amharic Fidel represents both a consonant (35) and a vowel (7 + 5 diphthongs) and is unique, yet there are similarities for each onset (consonant) and for each nucleus (vowel) of syllables. Below, note the similarities in shape corresponding to consonants as well as vowels: Note that

ሀ	**ሁ**	**ሂ**	**ሃ**	**ሄ**	**ህ**	**ሆ**
ha	hu	hi	ha	hē	hə	ho
ለ	**ሉ**	**ሊ**	**ላ**	**ሌ**	**ል**	**ሎ**
le	lu	li	la	lē	lə	lo

spoken Amharic does not follow the syllabic structure of the Fidel characters strictly; for example, the 6th form may have an ə] vowel or no vowel at all.

As we will show, word embedding models face unique challenges in this lesser-studied language. An abugida orthographic system as described above (besides Fidel, see also e.g., Devanagari) is distinct from an abjad (requisite glyphs only represent consonants, e.g., Arabic), syllabary (glyphs represent syllables with no visual similarity between related sounds, e.g., Japanese hiragana or katakana), or alphabet (glyphs give vowels equal status to consonants).

Thus, this work introduces both simple (Section 3.1.) and complex (Section 3.2.) embedding models designed for Amharic. In addition, it releases two Amharic language resources[2]: a cleaned but unlabeled corpus of digital Amharic text used for training word embeddings, and an Amharic evaluation resource for morphological and semantic word analogies (Section 4.).

Our results (Section 5.) show that word embeddings in the Amharic language benefit from linguistically informed treatment of the orthography, and that simple pre-processing architectures outperform complex models. We note that the utility of embeddings in this lesser-resourced language is hampered by the need for large corpora, especially on heavily semantic tasks.

2. Related Work

Mikolov et al. (2013) have shown that words can be represented using distributed vectors by implicitly factorizing the co-occurrence of the tokens (Levy and Goldberg, 2014) and their `word2vec` algorithm that employs the asynchronous gradient update has been extensively used. Similarly, Pennington et al.'s GloVe representation (2014) performed their matrix factorization on windows of words explicitly. These well-known methods were able to demonstrate effectiveness in inferring morphological and semantic regularities, despite ignoring morphological regularities be-

[1] `https://www.ethnologue.com/language/amh`

[2] `https://github.com/leobitz/amharic_word_embedding`

tween words. However, challenges such out of vocabulary and complex morphological structure led the exploration of subword or character-level models. Closest to this work are the character-aware neural language modeling approach of Kim et al. (2016) and the `fastText` sub-word skip-gram (swsg) models of Bojanowski et al. (2017). Kim et al. used a convolutional neural network (CNN) as their word-level representation, directly feeding dense vectors (embeddings) to a long short-term memory (LSTM) language model.

Bojanowski et al.'s `fastText` (2017) introduced a simple subword model on top of the skip-gram model (2017): subword character n-grams, alongside the original target word, have their own vectoral representations that are summed to form a word (e.g., for `slant` with $n = 3$, the set of vectors would include {`<sl`, `sla`, `lan`, `ant`, `nt>`}). This subword model improved embeddings for the morphologically rich languages of Arabic, German, and Russian in a word similarity task. Other work on morphologically rich languages has applied subword models to tasks like machine translation (Vylomova et al., 2017).

Training models across more languages leads to further insights on subword modeling; in subsequent work on `fastText`, Grave et al. (2018) trained models in 157 languages. This massive study showed that a large training corpus for embeddings (the Common Crawl) was extremely important in building more accurate embeddings.

Significant efforts in aligning embeddings across multiple languages (Zou et al., 2013; Ammar et al., 2016) are only tangentially related to ours. Multilingual embeddings, though, have been jointly developed with other tasks such as language identification (Jaech et al., 2016).

Instead of translated evaluation sets (Zahran et al., 2015; Abdou et al., 2018), we are interested in building mono-lingual word/subword embeddings validated within their own linguistic context. Most non-English models with significant monolingual evaluation efforts (other than those described above) are relatively well-resourced compared to Amharic, e.g., Chinese (Chen et al., 2015) and Arabic (Dahou et al., 2016). We employ Arabic's larger resources for comparison, since it is also in the Semitic language family.

3. Methods

3.1. Alphabetization and abjadization

Our first subword models combine two simple linguistically-motivated pre-processes with the Bojanowski et al.'s model (2017). First, we normalized Amharic Fidel consonants and represented vowels separately, approximating an alphabet. Second, we normalized consonants and eliminated the vowels, approximating an abjad. Below, we chose the 6th form of each Fidel to represent the consonant, since that form is also used for a consonant with a null vowel.

ሰላም ⟶ ስeልaምɨ ⟶ ስልም

se la m ⟶ s e l a m ⟶ s l m

Abugida ⟶ Alphabet ⟶ Abjad

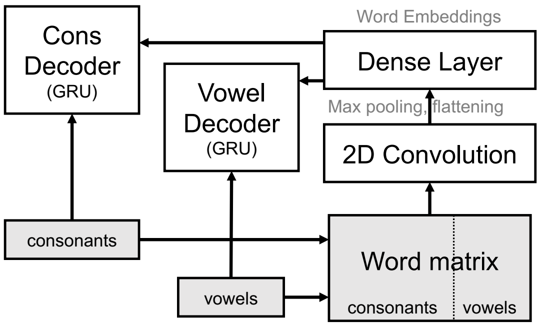

Figure 1: Consonant-Vowel subword (cvsw) model, with CNN character embedding and separate RNN-modeled consonant and vowel sequences.

Note that this is computationally different from other purported abugidas. Hindi's Devanagari script has vowel markers that are separate Unicode code points, easily distinguished by computational systems; Amharic's Fidel script has a single Unicode code point for each consonant–vowel character. Indeed, only full syllabaries are as opaque from the standpoint of digitalized orthographies.

3.2. Consonant-Vowel subword model

We introduce a consonant-vowel subword model (cvsw) for alphasyllabary text, shown in Figure 1. Here, an input word is represented in 3 ways: a sequence of one-hot consonants, a sequence of one-hot vowels, and a static matrix where each row is a two-hot vector (one-hot consonant concatenated with a one-hot vowel) representing an alphasyllabary character in the word.

The model is similar to autoencoder which learns efficient data encoding. It contains an "encoder" CNN (followed by max pooling and a dense layer) which embeds the sequence of the alphasyllabary characters in a given word. We chose this CNN to encode the input to the latent space after preliminary experiments with alternative architectures, e.g., one-hot or removing RNN input sequences, with the intuition that an embedding of a word is directly formed from its consonant and vowel. The dense layer is considered the embedding of a given sequence. Two recurrent neural networks (RNNs) using gated recurrent units (GRUs) "decode" the sequence of the input characters. One RNN predicts the consonant sequence; the other predicts the vowel.

The probability of predicting a character l is given by:

$$\begin{aligned} P(l_1...l_t \mid x_1...x_t) &= \prod P(c_t \mid u, c_1...c_{t-1}) \\ ] \qquad & \cdot \prod P(v_t|u, v_1...v_{t-1}) \\ \log(P(l_1...l_t \mid x_1...x_t)) &= \log(P(c_t|u, c_1...c_t - 1)) \\ &\quad + \log(P(v_t|u, v_1...v_t - 1)) \end{aligned}$$

where l denotes the alphasyllabary character sequence to be predicted, c and v are the consonant and vowel features of the letter, and finally, u is the fixed length embedding (latent feature) of the input sequence.

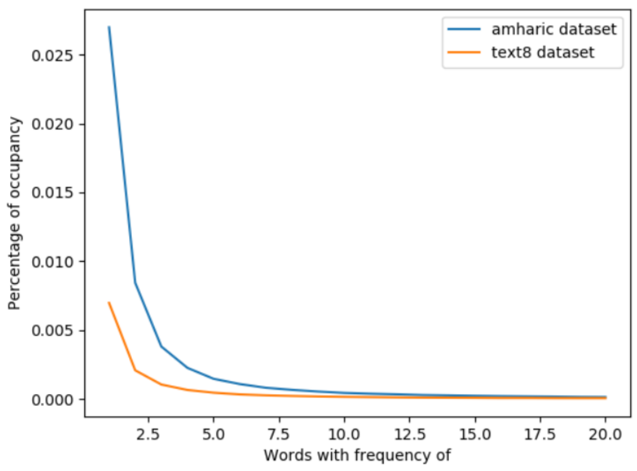

Figure 2: Our dataset for the morphologically rich, fusional language of Amharic, in comparison with the English text8 benchmark, has more words that are used few times (left side of graph).

4. Experiments

4.1. Training Data

We built a new, unlabeled training corpus by using our own crawler on four Amharic news websites from a variety of genres. As a pre-process, we removed non-Amharic words and letters and replaced all digits with the # character. After pre-processing, the corpus contained 16,295,955 tokens consisting of 855,109 unique tokens with average frequency of 19.05; this is similar in size to the English text8 benchmark (17 million tokens, with an average frequency of 66). The resulting corpus was used for all evaluated Amharic embeddings. For comparison, Figure 2 shows the distribution for 17M tokens in both the collected Amharic dataset vs. the well known `text8` (first 1×10^8 characters in English Wikipedia). The Amharic line shows a much higher prevalence of words with 1–5 counts than the `text8` line. This illustrates the typological differences between English vs. the fusional and morphologically rich Amharic.

For Arabic, we trained embeddings with the Arabic Wikipedia dump from 20th December, 2018, and followed standard pre-processing procedures.[3] See Table 1.

Although Amharic contains 12 vowels, 2 of them never appear in our corpus, so we choose to model the language as having only 10 vowels in all models. A further 3 (diphthongs) are relatively less-frequently used. On the consonants side, we include 40 characters: 39 typical Amharic characters, plus one reserved for miscellaneous other characters.

4.2. Word Analogy Task

Rather than translating an English corpus, we collected language-specific evaluation data. We created a web interface where native Amharic-speaking student volunteers were presented with random words from the Amharic corpus (and a refresh button).

Users created *word analogies* by dragging & dropping 4 words in a $A : B :: C : D$ pattern. With this relatively unbiased process, we collected 1731 Amharic analogies; 568 were semantic (e.g., `nephew : niece :: son : daughter`), while 1163 were morphological (e.g., `building : buildings :: road : roads`). Following Mikolov et al. (2013), each word in these analogies was iterated through for evaluation (i.e., 4562 morphological analogies), using vector arithmetic ($A \approx C - D + B$) and the top n closest words (Levy and Goldberg, 2014).

Extrinsic evaluation is out of the scope of this work due to the poor availability of resources in Amharic. Instead, we compare with Arabic embeddings on the Google word analogy task (Zahran et al., 2015), trained on a similarly-sized corpus (Section 5.).

[3]Available at: https://github.com/motazsaad/process-arabic-text/blob/master/clean_arabic_text.py

Pre-processed corpus	**Amharic**	**Arabic**
Total tokens	16,295,955	16,284,308
Unique tokens	855,109	584,204
Avg. token frequency	19.057	27.874
After min-thresholding at frequency of 5		
Total tokens	15,129,549	15,523,827
Unique tokens	155,427	131,014
Avg. token frequency	97.342	118.499

Table 1: Corpus statistics on web news (Amharic) and Wikipedia subset (Arabic). Min-thresholding at frequency of 5 (default in word2vec tool) removes rare words.

4.3. Experimental setup

Our evaluations report the grid-search-maximized combinations of the following variables: written language $l \in$ {Amh-Fidel, Amh-Alphabet, Amh-Abjad, Arabic}; model $m \in$ {skip-gram, `fastText`, CNN}; embedding dimension $d \in \{100, 200, 300\}$; window size $w \in \{0, 1, 2\}$; subword size $n \in$ {2–4, 2–6} for `fastText`.

“Order” of co-occurrence. As noted in Artetxe et al. (2018), word embeddings depend on word co-occurrence. Tuning the order of these co-occurrences, even through a simple linear transformation, can have a profound effect on whether the embeddings perform well in morphological vs. semantic tasks. Our experiments thus sweep the co-occurrence “order” parameter of $\alpha \in [-1.0, 1.0]$.

CNN model parameters. For input in the CNN-based model, consonants are represented by a size-40 vector; vowels are size-10. Words are at longest 11 characters in our corpus, and two more characters are added to signal the beginning and ending of a sequence. Hence, our input is a 13×50 matrix. The encoder is composed of a 16 (5, 5) filters. It is followed by max pooling with a (3, 3) filter. Finally, the flattened output of the pooling layer is passed through a dense layer which will

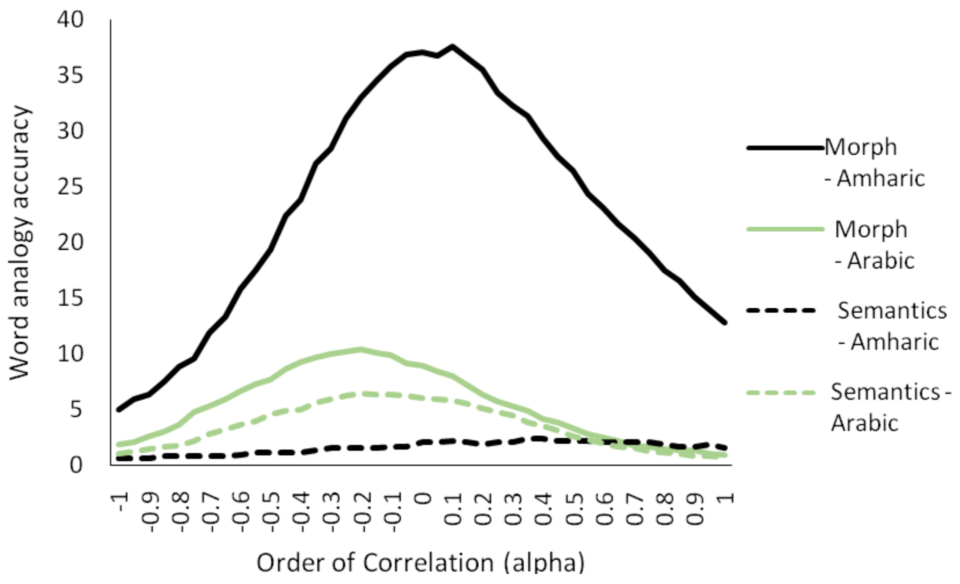

Figure 3: The "order of co-occurrence" for embeddings (Artetxe et al., 2018) controls optimization for morphological vs. semantic embeddings.

	Morph	Sem
No-subword sg (Fidel)	8.44	**5.87**
swsg (Abjadized)	26.87	2.12
swsg (Fidel)	37.64	2.31
swsg (Alphabetized)	**49.85**	2.84
cvsw (Fidel)	4.21	(+sg) 2.95
No-subword sg (Arabic)	7.31	10.85
swsg (Arabic)	10.40	6.97

Table 2: Accuracy (best parameters) of subword models vs. skip-gram (word2vec) on morphological vs. semantic analogies.

map it to 100 length vector. This vector is the latent representation of the input with all its morphological intricacies. We defined and trained the model using the `keras` deep learning library, with an initial learning rate of 0.0003, the Adam optimizer, and 500-sample batches, applying Nesterov momentum for 10 epochs.

5. Results and Discussion

5.1. Word Analogy Results

Table 2 reports the main results of our work.

First, subword models were able to improve the performance on morphological word analogies, but surprisingly, subword information proved detrimental to semantic analogy performance. Furthermore, off-the-shelf `fastText` made a more pronounced improvement on morphological performance in Amharic (from 8.44 to 37.64) than in Arabic (7.31 to 10.40).

Focusing on the morphological test, note that swsg subword models all outperformed a no-subword skip-gram model (8.44). With swsg, abjadization dispenses with some of the available information, harming performance (26.87) relative to the full Fidel representation (37.64), while alphabetized swsg achieves the highest overall performance (49.85).

The performance of our proposed neural cvsw is overall a negative result, underperforming the baseline non-subword model. On morphology, this model considers only internal character patterns and is not robust to small variations for morphologically similar words. When cvsw is inserted into a skip-gram model, it exhibits the best performance among the subword models on the semantic task; the fact that this underperforms a basic sg model highlights the difficulty of improving the semantic performance via morphological information.

5.2. Order of co-occurrence

Amharic subword models have differential effects on morphological vs. semantic performance; these are more pronounced than in Arabic. Diving into these divergent objectives, Figure 3 illustrates linear post-transformations of the embeddings (Artetxe et al., 2018) based on the "order of co-occurrence" (already maximized upon in Table 2's results). Artetxe et al. found that embedding models can often support both semantic and morphological tasks, but semantic tasks are often better addressed by lower values of α and morphological tasks by higher values.

Figure 3 has maxima for morphology and semantics at similar α values. This implies that Arabic and Amharic embeddings are not as affected by "order of co-occurrence" — e.g., are not able to 'recover' performance on the opposite type of word analogy. We expect that this is due to the fusional, morphology-rich nature of Semitic languages, which suggests that producing useful embeddings in these languages is challenging.

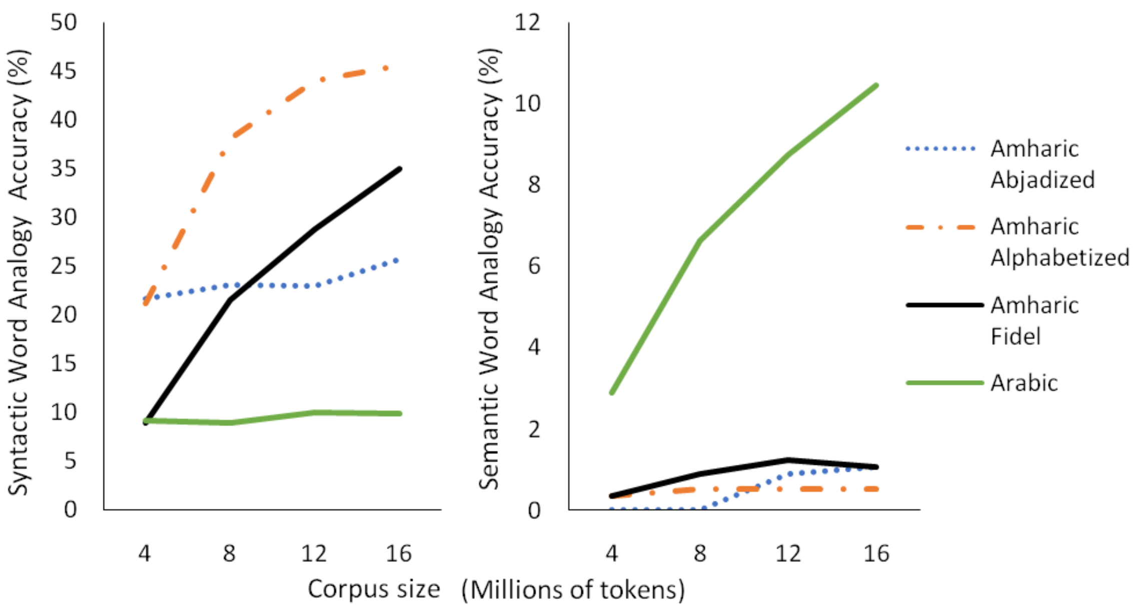

Figure 4: Training curve on the morphological & semantic word analogy tasks, showing accuracy of top Amharic and Arabic models.

5.3. Training Curves

Figure 4 shows training curves for the best Amharic vs. the best Arabic models, up to the Amharic corpus size of 16M (in tokens, before pre-processing).

Interestingly, the Arabic and abjadized Amharic models improve slowly with corpus size. It appears that morphologically important information is stored in vowels for both languages, and the abjad orthographies fail to surface that information for computation. On the other hand, Amharic's Fidel orthography and its alphabetized counterpart improve rapidly with new data, making use of this vowel information.

The training curve for semantic word analogies shows the expected characteristic for Arabic and even for the poor-performing Amharic subword models. At minimum, the upward training curves show that large data is important for high embedding quality, reiterating previous lessons learned (Grave et al., 2018).

6. Conclusion

We have presented methods and challenges for word embeddings in Amharic, and released our training and evaluation data at `github.com/leobitz/amharic_word_embedding/`. Accounting for the consonant–vowel alphasyllabary in Amharic text has a significant effect on the quality of produced embeddings. Other Amharic tasks may benefit from a simple alphabetization of its digital orthography.

Our future work includes further resource development for Amharic and related languages, rigorous evaluations of word embeddings and language models on extrinsic NLP-related tasks, and transfer learning with newer pre-trained models.

7. Bibliographical References

Abdou, M., Kulmizev, A., and Ravishankar, V. (2018). Mgad: Multilingual generation of analogy datasets. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018).

Ammar, W., Mulcaire, G., Tsvetkov, Y., Lample, G., Dyer, C., and Smith, N. A. (2016). Massively multilingual word embeddings. arXiv preprint arXiv:1602.01925.

Artetxe, M., Labaka, G., Lopez-Gazpio, I., and Agirre, E. (2018). Uncovering divergent linguistic information in word embeddings with lessons for intrinsic and extrinsic evaluation. Proceedings of the 22nd Conference on the Computational Natural Language Learning (CoNLL), pages 282–291.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. Journal of Machine Learning Research, 3:993–1022.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5:135–146.

Chen, X., Xu, L., Liu, Z., Sun, M., and Luan, H.-B. (2015). Joint learning of character and word embeddings. In IJCAI, pages 1236–1242.

Dahou, A., Xiong, S., Zhou, J., Haddoud, M. H., and Duan, P. (2016). Word embeddings and convolutional neural network for arabic sentiment classification. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2418–2427.

Deerwester, S., Dumais, S., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6):391–407.

Eberhard, D. M., Simons, G. F., and Fennig, C. D. (2020). Ethnologue: Languages of the World. SIL International, Dallas, Texas, 23 edition.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).

Jaech, A., Mulcaire, G., Hathi, S., Ostendorf, M., and Smith, N. A. (2016). Hierarchical character-word models for language identification. In Conference on Empirical Methods in Natural Language Processing, page 84.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In AAAI, pages 2741–2749.

Levy, O. and Goldberg, Y. (2014). Linguistic regularities in sparse and explicit word representations. In Proceedings of the eighteenth conference on computational natural language learning, pages 171–180.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. ICLR Workshop.

Padó, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. Computational Linguistics, 33(2):161–199.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In EMNLP, volume 14, pages 1532–1543.

Vylomova, E., Cohn, T., He, X., and Haffari, G. (2017). Word representation models for morphologically rich languages in neural machine translation.

Zahran, M. A., Magooda, A., Mahgoub, A. Y., Raafat, H., Rashwan, M., and Atyia, A. (2015). Word representations in vector space and their applications for arabic. In International Conference on Intelligent Text Processing and Computational Linguistics, pages 430–443. Springer.

Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1393–1398.