Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 4726–4730
Marseille, 11–16 May 2020

GGP: Glossary Guided Post-processing for Word Embedding Learning

Ruosong Yang, Jiannong Cao, Zhiyuan Wen
The Hong Kong Polytechnic University
Hong Kong, China
{csryang,csjcao,cszwen}@comp.polyu.edu.hk

Abstract

Word embedding learning is the task to map each word into a low-dimensional and continuous vector based on a large corpus. To enhance corpus based word embedding models, researchers utilize domain knowledge to learn more distinguishable representations via joint optimization and post-processing based models. However, joint optimization based models require much training time. Existing post-processing models mostly consider semantic knowledge so that learned embedding models show less functional information. Compared with semantic knowledge sources, glossary is a comprehensive linguistic resource which contains complete semantics. Previous glossary based post-processing method only processed words occurred in the glossary, and did not distinguish multiple senses of each word. In this paper, to make better use of glossary, we utilize attention mechanism to integrate multiple sense representations which are learned respectively. With measuring similarity between word representation and combined sense representation, we aim to capture more topical and functional information. We propose GGP (Glossary Guided Post-processing word embedding) model which consists of a global post-processing function to fine-tune each word vector, and an auto-encoding model to learn sense representations, furthermore, constrains each post-processed word representation and the composition of its sense representations to be similar. We evaluate our model by comparing it with two state-of-the-art models on six word topical/functional similarity datasets, and the results show that it outperforms competitors by an average of 4.1% across all datasets. And our model outperforms GloVe by more than 7%.

Keywords: Word Embedding, Post-processing model, Representation Learning

1. Introduction

Word embedding learning is the task to utilize a continuous vector to represent each word. With the success of Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), which are trained on a large corpus via a simple neural network or matrix factorization, pre-trained word representations are widely used in various Natural Language Processing (NLP) tasks, such as sequence labeling task (Lample et al., 2016), text classification (Kim, 2014), etc. However, typical word embedding models including Word2Vec and GloVe, are based on the Distributional Hypothesis (Harris, 1954), which utilizes the distribution of context words as the target word representation. In practice, the corpus is always limited, which makes it difficult to calculate the actual context word distribution of each target word. For example, synonyms and antonyms are usually hard to distinguish. Meanwhile, the quality of the word embedding highly depends on the frequency of the word in the corpus, rare words are usually discarded.

To improve the quality of the word embedding, various knowledge bases such as WordNet (Miller, 1995), FrameNet (Baker et al., 1998), and Paraphrase Database (Ganitkevitch et al., 2013) are considered. Meanwhile, joint optimization and post-processing are two popular approaches to incorporate domain knowledge, which have achieved better performance. Joint optimization based models design extra constraints according to domain knowledge. And they retrain a new model with integrated objectives on a large corpus and knowledge bases, which usually require much training time. As for post-processing based models, they fine-tune pre-trained word embedding models with new constraints on the knowledge bases, and the data volume of the knowledge bases is much smaller than that of the corpus. So post-processing is a more efficient way. Besides, as many pre-trained word vectors exist, e.g., Google News Vectors [1] (based on Word2Vec), GloVe [2], and FastText [3] (Bojanowski et al., 2016), post-processing approaches are more effective.

Recent works focus on fine-tuning pre-trained word embedding models with word-to-word semantic knowledge such as synonyms and antonyms. Retrofitting (Faruqui et al., 2015) utilized semantic lexicons' relational information and assumed linked words should learn similar representations. ER-CNT (Glavaš and Vulić, 2018) used a deep neural network to fine-tune word vectors by adding constraints on synonyms and antonyms. It shows a significant improvement in topical similarity datasets, while loses the functional (Levy and Goldberg, 2014) information. There are also some works utilize the glossary to learn word representations via extending the original corpus with word definitions. The Dict2vec (Tissier et al., 2017) model constructed word pairs from both the corpus and dictionary entries. Especially for negative sampling, it ignored the words in pairs from the second part. Then Skip Gram (Word2Vec) is used to learn the word embedding model. CPAE (Bosc and Vincent, 2018) proposed a LSTM based auto-encoding model to learn word representations from dictionary definitions which are assumed to be similar to their embedding. However, Dict2vec required to retrain the word embedding model, which was time-consuming. CPAE combined multiple senses of each word into one sense, which is not reasonable. And in CPAE, the authors also introduced a post-processing model, however, it could only fine-tune the words having a definition or that occur in definitions.

Glossary is the collection of glosses, and each gloss con-

[1]https://code.google.com/archive/p/word2vec/
[2]https://nlp.stanford.edu/projects/glove/
[3]https://fasttext.cc/docs/en/english-vectors.html

sists of a word as well as its definition (a word sequence to explain the word). In common sense, looking up the glossary (dictionary) is one important way for humans to learn a new word, and learning from example sentences of the word is another way. However, corpus is always not enough to estimate the real context words distribution, and there are also no effective approaches for semantic composition. Combining the corpus and the glossary is necessary, and they should be modeled in different ways. In Distributional Hypothesis based word embedding models, word vector represents the distribution of context words, which combines context words of different senses. Besides learned word representations are the composition of different senses. So word embedding should be approximate to the linear combination of its senses' representations.
We propose GGP (**G**lossary **G**uided **P**ost-processing) model, which combines a general function to fine-tune all pre-trained word representations, and an auto-encoding model to learn the representation of each sense. By joint optimization, embedding learned by GGP exhibits better topical and functional similarity. In addition, to speed up the convergence of the auto-encoding model, word definitions are also used to pre-train it. To evaluate whether our model learns both topical and functional information, we test our model on six word tpical/functional similarity datasets, i.e., Men, Simverb3500, RW, Simlex999, WS353 and Mturk. Experimental results show that our model outperforms rivals by at least 4.1% in all benchmarks. And compared to GloVe, our model outperforms more than 7%. We also use two glossaries and two pre-trained word embedding models with different vocabulary sizes to show the effectiveness of our model. In summary, our contribution are:

- For multi-sense word, we learn sense representation respectively and utilize attention to integrate them.
- We combine a general post-processing function and sense representation learning model so that each pre-trained word representation could be post-processed.
- Experimental results show that our model learn both topic and functional information and performs much better than previous model.

2. GGP Model

Our model contains two parts as shown in Figure 1. The left part is a sequence to sequence auto-encoding model (Li et al., 2015), which is used to transform each sense entry into a sense vector. The right part is a general mapping function, a multi-layer fully-connected feed-forward network, to fine-tune each word vector. Besides, we expect the general function could preserve learned information so that the word embedding will not change too much. Furthermore, We constrain the linear composition of sense vectors from the left part to be similar to the post-processed vector in the right part.

2.1. Sequence to Sequence Auto-encoding

The auto-encoding model consists of an encoder which transforms a sequence into a vector, and a decoder which decodes the vector to a new sequence. The encoder has two layers, the first layer is a bi-directional Long Short-Term Memory (LSTM) neural network which captures the full context information of each word. The second layer, a vanilla LSTM network, is used to learn a composition function to transform a whole sentence into one vector by utilizing word order information. The decoder is also a two-layer vanilla LSTM neural network which tries to reconstruct the input sequence from the output of the encoder.
Given a word w_t, the definition sequence of i_{th} sense is $in_t^i = \{in_{t,1}^i, in_{t,2}^i, ..., in_{t,n_i}^i\}$, where n_i is the number of words of the sense definition. $Encoder(\cdot)$ and $Decoder(\cdot)$ are the encoder model and the decoder model respectively. The encoder transforms the definition sequence into a sequence of hidden states $he_t^i = \{he_{t,0}^i, he_{t,1}^i, ..., he_{t,n_i}^i\}$ as shown in Formula 1. Meanwhile, he_{t,n_i}^i is used to represent the i_{th} sense. And the decoder utilizes the sense representation he_{t,n_i}^i as the initial hidden state $hd_{t,0}^i$, then generates a new sequence $hd_t^i = \{hd_{t,1}^i, hd_{t,2}^i, ..., hd_{t,n_i}^i\}$ as shown in Formula 2.

$$he = Encoder(in_t^i) = BiLSTM(LSTM(in_t^i)) \quad (1)$$

$$hd = Decoder(he_{t,n_i}^i) = LSTM(LSTM(he_{t,n_i}^i)) \quad (2)$$

The reconstruction loss L_r of the auto-encoding part is defined by $CrossEntropy$ loss shown in Formula 3, where n_t is the number of senses, in_t^i is the input sequence of the i_{th} sense, and hd_t^i is the output sequence of the decoder.

$$L_r = \frac{1}{n_t}\sum_{i=1}^{n_t} CrossEntropy(hd_t^i, in_t^i) \quad (3)$$

2.2. Composition of Sense representations

With all calculated sense representations $he_t = \{he_{t,n_1}^1, he_{t,n_2}^2, ..., he_{t,n_{n_t}}^{n_t}\}$ of the given word, additive attention (Bahdanau et al., 2014) is used to calculate S_{w_t}, the composition representation of all sense vectors, which is shown as Formula 4:

$$S_{w_t} = \sum_{i=1}^{n_t} w_i * he_{t,n_i}^i \quad (4)$$

Where he_{t,n_i}^i is the i_{th} sense representation of the word t, w_i is the weight of the representation of i_{th} sense calculated as Formula 5. v_{w_t} is the word embedding of word t, besides, W and b are parameters to be learned.

$$w_i = \frac{tanh(W[v_{w_t}, he_{t,n_i}^i] + b)}{\sum_{j=1}^{n_t} tanh(W[v_{w_t}, he_{t,n_j}^j] + b)} \quad (5)$$

2.3. Multi-layer Fully-connected Feed-Forward Network

To learn a general post-processing mapping function to fine-tune any pre-trained word vectors, a multi-layer fully-connected feed-forward neural network is adopted. It maps a vector $v_{w_t} \in R^{d*1}$ to a new one $v'_{w_t} \in R^{d*1}$, where d is the dimension size of the word embedding. Each layer is calculated as:

$$h_n = \tanh(W_{n-1}h_{n-1} + b_{n-1}) \quad (6)$$

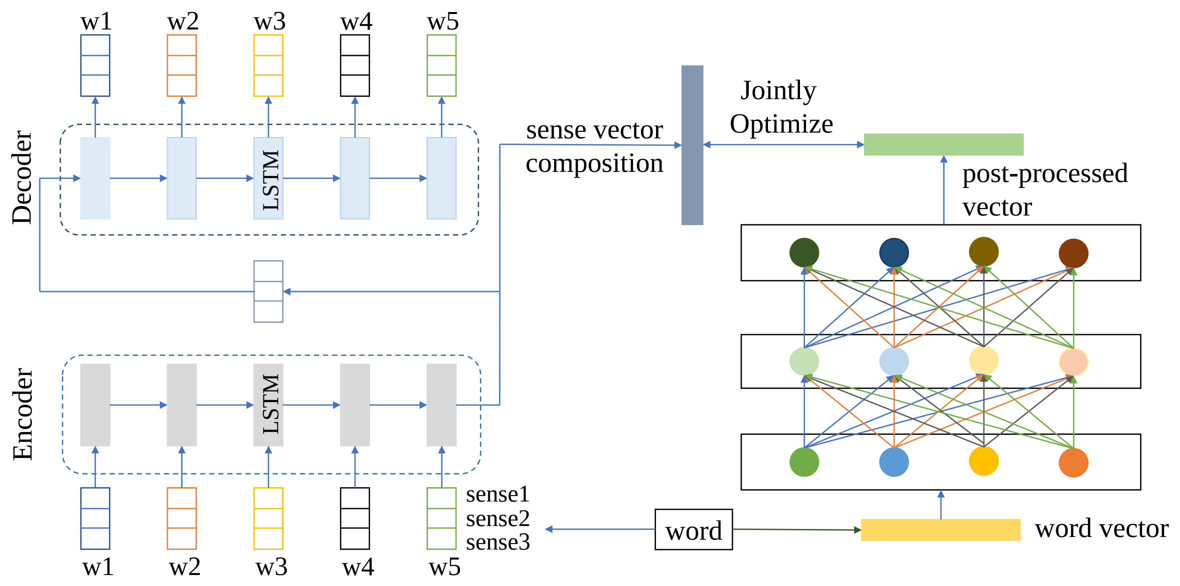

Figure 1: Model Framework (an example of one word with three senses)

Where W and b are parameters to be learned. And the input h_0 is the pre-trained word vector v_{w_t}, the post-processed vector v'_{w_t} is h_n. To preserve the information learned by the pre-trained model, we use F_2 norm as the loss L_n shown in Formula 7.

$$L_n = \|v'_{w_t} - v_{w_t}\|_F^2 \tag{7}$$

2.4. Extra Constraint and Joint Objective

Given post-processed word vector v'_{w_t} and sense composition representation S_{w_t}, we assume the word representation learned from the corpus and that from the glossary should be similar. The loss of the extra constraint L_s is defined as Formula 8, where D means distance measurement.

$$L_s = D(v'_{w_t}, S_{w_t}) \tag{8}$$

All these three objectives are optimized jointly, so the total loss L is defined as Formula 9, where α and β are adjustable weights.

$$L = L_s + \alpha L_r + \beta L_n \tag{9}$$

3. Experiment and Discussion

3.1. Pre-trained Vectors

GloVe has several pre-trained word vectors with different sizes of corpuses, and it outperforms other word vectors in most word similarity tasks. We select two pre-trained word vectors trained on two extremely large corpuses. One is trained on the corpus of 840 billion tokens and the vocabulary has 2.2 million unique words. The other one uses the corpus containing 42 billion tokens and the size of the vocabulary is 1.9 million. All of the word vectors are in 300 dimensions.

3.2. Definition Entries

In our experiment, we construct two dictionaries from WordNet and Opted. We only extract the definitions explaining the words in the two vocabularies. For the dictionary from WordNet (Miller, 1995), we extract the word definitions with multi-senses via the interface provided by NLTK [4]. And Opted is from an open project called The Online Plain Text English Dictionary (Gutenberg, 2009), we parse all the original dictionary files downloaded from the website [5] and construct a unified dictionary, while each definition is separated into multiple senses. We also calculate some statistical properties, including the number of words having senses (NW), the average number of senses for each word (NSW), and the average length for each sense (ALS), shown in Table 1. We find that Opted contains more words in the two vocabularies of the pre-trained models, and provides more words to explain senses.

Vocab	Dict	**NW**	**NSW**	**ALS**
1.9M	WordNet	51621	1.57	5.39
1.9M	Opted	64616	3.49	6.10
2.2M	WordNet	47337	1.62	5.33
2.2M	Opted	58575	3.68	6.04

Table 1: Dictionary Statistics

3.3. Pre-train Auto-encoding Model

In previous work (Howard and Ruder, 2018), researchers proposed a pre-trained language model to improve classification tasks. Motivated by it, we also pre-train the auto-encoding part, then jointly optimize the auto-encoding part as well as the post-processing part. Furthermore, dictionary entries are used to train the auto-encoding model until the loss on the validation set increases.

3.4. Model Parameter Specification

In all experiments, we adopted $softmax$ as a special distance measurement, which is approximated by negative sampling (Mikolov et al., 2013). We separate 20% of word and definition pairs as validation samples. And we use two different word embedding matrices for auto-encoding and

[4] http://www.nltk.org/howto/wordnet.html
[5] http://www.mso.anu.edu.au/~ralph/OPTED/

Model	Corpus	Dict	**SV**	**SL**	**Men**	**RW**	**Mturk**	**WS353**	**WS-S**	**WS-R**
GloVe	42B	-	22.6	37.4	74.3	37.4	64.5	63.2	69.8	57.1
ER-CNT	42B	-	**47.0**	**61.1**	- 64.7	36.3	56.6	59.5	66.0	49.0
CPAE	42B	-	27.5	33.2	52.2	23.0	33.7	40.5	49.5	32.8
GGP	42B	WN	28.6	40.6	80.2	42.5	69.8	73.8	76.2	70.2
GGP + PT	42B	WN	29.3	42.4	80.7	42.8	**70.3**	75.5	78.0	72.7
GGP	42B	Opted	**30.6**	43.8	80.8	43.8	68.2	**75.6**	**78.4**	72.4
GGP + PT	42B	Opted	30.5	**44.2**	**81.2**	**44.6**	68.7	75.0	77.8	**72.7**
Glove	840B	-	28.3	40.8	80.5	45.5	69.3	71.2	80.2	64.4
ER-CNT	840B	-	**47.2**	**59.0**	67.6	43.2	62.3	59.3	70.6	44.6
CPAE	840B	-	33.8	39.7	62.3	32.4	41.7	48.7	60.3	39.3
GGP	840B	WN	32.0	42.7	**82.5**	49.0	71.5	73.8	79.0	67.9
GGP + PT	840B	WN	31.9	44.2	82.4	49.5	**72.3**	**75.5**	**80.2**	68.2
GGP	840B	Opted	32.7	**44.3**	82.2	48.6	69.5	73.0	79.4	65.4
GGP + PT	840B	Opted	**32.9**	44.2	82.1	**50.4**	70.1	74.7	79.9	**68.2**

Table 2: Word Similarity Experiment Results (Spearman's correlation coefficient ρ * 100)

fully-connected networks respectively, where both the embedding size are set to 300. In fully-connected networks, we set 3 hidden layers. α and β are both set to 0.3 for two pre-trained word embedding models. For each word and definition pair, the number of negative samples is 10. And an Adam optimizer is used with an initial learning rate as 0.5. Besides, the loss of validation samples is monitored to know when to stop early.

3.5. Word Similarity

We tested our model on six word similarity datasets, including Men (Bruni et al., 2014), Simverb3500 (SV) (Gerz et al., 2016), RW (Luong et al., 2013), Simlex999 (SL) (Hill et al., 2015), WS353 (Finkelstein et al., 2002) and Mturk (Radinsky et al., 2011). Simberb3500 and Simlex999 are topical similarity datasets (Glavaš and Vulić, 2018). Men, RW, and Mturk are functional similarity datasets. WS353 consists of a topical similarity part WS-S and a functional similarity part WS-R. Spearman's ρ rank correlation between the ground truth and calculated word similarity from word embedding is used to evaluate the performance. We show the experimental results on five models including GloVe, ER-CNT (Glavaš and Vulić, 2018), CPAE (Bosc and Vincent, 2018), and our model (GGP), as well as our model with the pre-trained auto-encoding model (GGP+PT). The former three models are used as baseline models. For ER-CNT model, the paper only reported the performance on SV and SL datasets, we run the source code published by the authors [6] to obtain the performance on all six datasets. CPAE only utilized Word2Vec to evaluate their model and ignored GloVe, since they said that GloVe performs much better than Word2Vec in their paper. We also run the source code published by the authors [7] to test the model on the six datasets. Finally, we tested two pre-trained word embeddings with different vocabulary sizes and two different dictionaries.

[6]https://github.com/codogogo/explirefit

[7]https://github.com/tombosc/cpae

3.6. Analysis and Discussion

According to Table 2, ER-CNT (Glavaš and Vulić, 2018) outperforms other models by a large margin in SV and SL (two topical similarity tasks), however, on the rest functional similarity datasets ER-CNT performs worse than GloVe. CPAE (Bosc and Vincent, 2018) shows a little improvement in SV, and performs worse on all other datasets. Our model enhances GloVe by around 7% in all datasets, which shows that our model could capture topical and functional information simultaneously.

The Influence of Distance Measurement Absolute distance and relative distance are two common distance measurements. Absolute distance including cosine similarity, Euclidean distance, etc., stresses that two words should be similar enough. Relative distance, asks two words should be closer than another two words, for example, $softmax$ constrains that the distance between the target word and context word should be smaller than that between the target word and negative words. In ER-CNT (Glavaš and Vulić, 2018), "Contrasting Objective" is also a relative distance and shows significant improvement. In our model, $softmax$ also shows a surprisingly better performance. The reason is that the relative distance tends to give a partial order to several words (more than two words), while the absolute distance only constrains two words.

The Influence of the Size of the Corpus GloVe learned from the larger corpus performs better. When GloVe trained on the small corpus was fine-tuned with the glossary, it shows a comparable, sometimes even better performance, which shows the effectiveness of utilizing the glossary. Experiments show less improvement when incorporating the glossary with the larger corpus, for the overlapping information between the corpus and the glossary increases. Opted dictionary show better performance than WordNet on the small corpus. Since Opted contains more explanations which provide more extra information.

The Influence of Pre-Trianed AE In most datasets, pre-trained auto-encoding model shows improvement. Maybe the pre-trained auto-encoding model helps the total model to be trained from a better initialization.

4. Conclusion and Future Work

In this paper, we propose a model to incorporate dictionary entries to post-process word embedding to be more topical and functional. Experimental results show the effectiveness of our model. To further improve the work, we will combine various word relationships in a partial order to improve the word embedding models.

5. Acknowledgement

The work was supported by PolyU Teaching Development with project code 1.61.xx.9A5V and Hong Kong Collaborative Research Fund with project code C5026-18G.

6. Bibliographical References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Bosc, T. and Vincent, P. (2018). Auto-encoding dictionary definitions into consistent word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1532. Association for Computational Linguistics.

Bruni, E., Tran, N.-K., and Baroni, M. (2014). Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615. Association for Computational Linguistics.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. (2013). Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Gerz, D., Vulić, I., Hill, F., Reichart, R., and Korhonen, A. (2016). Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*.

Glavaš, G. and Vulić, I. (2018). Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–45. Association for Computational Linguistics.

Gutenberg, P. (2009). The online plain text english dictionary. `http://www.mso.anu.edu.au/~ralph/OPTED/`.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Hill, F., Reichart, R., and Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June. Association for Computational Linguistics.

Levy, O. and Goldberg, Y. (2014). Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Li, J., Luong, M.-T., and Jurafsky, D. (2015). A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Luong, T., Socher, R., and Manning, C. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Radinsky, K., Agichtein, E., Gabrilovich, E., and Markovitch, S. (2011). A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Tissier, J., Gravier, C., and Habrard, A. (2017). Dict2vec: Learning word embeddings using lexical dictionaries. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 254–263.