Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2018”

Moscow, May 30—June 2, 2018

LEARNING WORD EMBEDDINGS FOR LOW RESOURCE LANGUAGES: THE CASE OF BURYAT

Konovalov V. P. (vaskonov@yahoo.ru),
Tumunbayarova Z. B. (zhargal@zabgu.ru)
Transbaikal State University, Chita, Russia

Word-vector representations have been extensively studied for rich resource languages with large text datasets. However, only a few studies analyze semantic representations of low resource languages, when only small corpus is available. In this study we introduce a methodology and compare techniques to learn semantic representations of low resource languages. The proposed methodology consists of defining accurate preprocessing steps, applying language-independent stemmer and learning word-vector representations. In addition, we propose a simple word embeddings evaluation scheme that can be easily adapted to any language. By using this methodology we learn word-vector representations for Buryat language. In order to promote further research we make the source code and the resulting word embeddings corpus publicly available.

Keywords: word2vec, word-embeddings, SVD, PMI, GloVe

ВЕКТОРНОЕ ПРЕДСТАВЛЕНИЕ СЛОВ ДЛЯ МАЛОРЕСУРСНЫХ ЯЗЫКОВ: НА ПРИМЕРЕ БУРЯТСКОГО ЯЗЫКА

Коновалов В. П. (vaskonov@yahoo.ru),
Тумунбаярова Ж. Б. (zhargal@zabgu.ru)
Забайкальский Государственный
Университет, Чита, Россия

1. Introduction

Using word embeddings is a standard practice in NLP systems, both in shallow and deep architectures [Goldberg, 2016]. Word embeddings exploits statistical techniques to embed words in a vector space. In this space, words with similar meanings tend to be located close to each other. These techniques are based on the Harris distributional hypothesis [Harris, 1954], which says that words in similar contexts have similar meanings. This statement provides a framework to use semantic relationship between words. Word embeddings has been used in a wide variety of applications such as query expansion [Chen & Chen, 2007], building bilingual comparable corpora [Zhu, Li, Chen, & Yang, 2013], clustering [Di Marco & Navigli, 2013].

Classical count-based methods such as PMI matrices and SVD factorization were very popular to represent words as vectors. However, recently word2vec approach has been proposed to represent words as dense vectors by applying neural embedding methods [Mikolov, Chen, Corrado, & Dean, 2013]. The neural embedding methods are particularly computationally-efficient predictive model for learning word embeddings. It comes in two types, the Continuous Bag-of-Words model (CBOW) and the Skip-Grams with Negative Sampling model (SGNS). In this work we answer the question of which model to choose for low resource languages when only small amounts of data are available.

Word normalization is an important data preprocessing step for learning word-vector representations. It improves the vectors quality by reducing language variability. In order to reduce words to a common base form, most stemmers use the extensive set of linguistic rules developed for specific language [Porter, 1980]. Usually low resource languages lack rule-based stemmers. In this work, we examine to what extend language independent stemmer can improve word embeddings quality when only small training data is available.

In addition, we suggest a new scheme for word vector evaluation. We aim to develop a method that can address the common shortcomings mentioned in [Hill, Reichart, & Korhonen, 2016], at the same time this methods can be easily reproducible for any language.

The remainder of the paper is organized as follows. Section 2 gives information on Buryat language. Then Section 3 provides an overview of the methods for building word-vector representations. Section 4 describes language independent stemming approach. Section 5 specifies the experimental setup and describes the evaluation methodology. Finally, Section 6 provides results and comparisons of various word embedding techniques.

2. Buryat language

Buryat language is one of the Mongolic languages. The majority of Buryat speakers live in Russia along the northern border of Mongolia where it is an official language in the Buryat Republic, Ust-Orda Buryatia and Aga Buryatia. According to the Russian census of 2002, there are 353,113 native speakers in Russia. In addition, there are at least 100,000 native speakers in Mongolia and the People's Republic of China as well. There are regularly published Buryat newspapers, journals, books, films, television and radio

programs, however, according to UNESCO report, Buryat is considered to be an endangered language and at risk of disappearing [Janhunen, 2006]. Implementation of NLP tools is crucial for language preservation and development. The first syntactic treebank for Buryat language based on the Universal Dependencies was developed in [Badmaeva & Francis, 2017]. Buryat language has seven cases and two numbers. Its alphabet is based on the general Cyrillic scripts with three additional letters.

3. Background

There are two major word-vector representation methods: the count-based methods (PMI matrix, SVD factorization) and the neural embeddings methods (SGNS, CBOW).

The previous results suggest that the new embedding methods consistently outperform the traditional methods by a non-trivial margin on many similarity-oriented tasks [Baroni, Bernardi, & Zamparelli, 2014]. However this result was reported only for rich resources languages [Altszyler, Sigman, Ribeiro, & Slezak, 2016]. For example, the model used in [Baroni, Bernardi, & Zamparelli, 2014] was trained on 2.8 billion tokens constructed by concatenating ukWaC[1], the English Wikipedia[2] and the British National Corpus [3]. Unfortunately such big resources are not available for many languages, including Buryat language.

We strictly follow the notation that was used in [Levy, Goldberg, & Dagan, 2015], where $w \in V_W$ is collection of words and $c \in V_C$ their contexts, and V_W and V_C are the word and context vocabularies. The collection of observed word-context pairs Is D. The number of times the pair $(w,)$ appears in D is denoted as $\#(w,c)$. Then, $\#(w) = \sum_{c' \in V_C} \#(w,c')$ and $\#(c) = \sum_{w' \in V_W} \#(w',c)$ are the number of times w and c occurred in D, respectively.

When words and contexts are embedded in a space of d dimensions, each word $w \in V_W$ is represented as a vector $\vec{w} \in \mathbb{R}^d$ and each context $c \in V_C$ is represented as a vector $\vec{c} \in \mathbb{R}^d$.

In this work we focused on fixed-window bag-of-words contexts, where D is obtained by using a corpus $w_1, w_2, ..., w_n$ and defining the contexts of word w_i as the words surrounding it in an L-sized window $w_{i-L}, ..., w_{i-1}, w_{i+1}, ..., w_{i+L}$.

Section 3.1 describes the traditional pointwise mutual information method, which follows by singular values decomposition method in the Section 3.2. In addition, we examined word embeddings learned by GloVe, SGNS and CBOW methods. Due to space limitations we omitted their description here. The GloVe (Global vectors for word representation) method was described in [Pennington, Socher, & Manning, 2014]. The neural based methods SGNS and CBOW were described in [Mikolov, Chen, Corrado, & Dean, 2013].

---

1 http://wacky.sslmit.unibo.it

2 http://en.wikipedia.org

3 http://www.natcorp.ox.ac.uk

3.1. Pointwise mutual information (PMI)

Traditionally, the word-vector representations can be achieved by constructing a high-dimensional sparse matrix M, where each row represents a word w in the vocabulary V_W and each column a context $\in V$. The cell value M_{ij} represents the association between the word w_i and the context c_j. Pointwise mutual information (PMI) is a traditional metric to measure this association [Church & Hanks, 1990].

$$PMI(w,c) = \log \frac{P(w,c)}{P(w)P(c)} \tag{1}$$

Good collocation pairs have high PMI because the probability of co-occurrence is only slightly lower than the probabilities of occurrence of each word. Conversely, a pair of words whose probabilities of occurrence are considerably higher than their probability of co-occurrence gets a small PMI score.

$PMI\ (w,c) = -\infty$ if the number of co-occurrence of w and c equals 0. In order to address this, positive PMI (PPMI) is used, in which all negative values are replaced by 0.

It is well-known that PMI (and PPMI) suffers from its bias towards infrequent events [Turney & Pantel, 2010]. A rare context c that co-occurred with a word w even once will often lead to relatively high PMI score because (c), which is in PMI's denominator, is very small. This causes a situation in which the most associated contexts with w are often very rare words.

This problem can be addressed by smoothing variation of PMI. Smoothing context distribution increases the probability of sampling a rare context ($P\ (c) > P\ (c)$ when c is rare), which reduces the PMI of (w,c) for any w co-occurring with the rare context c. The smoothed PMI is very effective and consistently improves performance across different tasks and methods [Levy, Goldberg, & Dagan, 2015].

$$PMI_\alpha(w,c) = \log \frac{P(w,c)}{P(w)P_\alpha(c)} \tag{2}$$

$$P_\alpha(c) = \frac{\#(c)^\alpha}{\sum_c \#(c)^\alpha} \tag{3}$$

3.2. Singular Value Decomposition (SVD)

The dense low-dimensional vectors can be obtained by applying truncated Singular Value Decomposition (SVD) [Eckart & Young, 1936]. Formally, SVD of matrix M is factorization of the form $U \cdot \Sigma \cdot V^T$, where U and V are orthonormal and Σ is a diagonal matrix of eigenvalues in decreasing order. We obtain $M_d = U_d \cdot \Sigma_d \cdot V_d^T$ by keeping only top d elements of Σ. The high dimensional spare word-vector representations of matrix M can be substituted by low dimensional dense vectors of W_{SVD} and C_{SVD} that represent words and contexts respectively.

$$W_{SVD} = U_d \cdot \Sigma_d,\ C_{SVD} = V_d \tag{4}$$

However, word-vector representations of W_{SVD} are not necessary the optimal for semantic tasks. It was shown that weighting the eigenvalues matrix Σ_d can have a significant effect on the performance [Levy, Goldberg, & Dagan, 2015].

$$W_{SVD}^p = U_d \cdot \Sigma_d^p \tag{5}$$

4. Word Normalization

Identifying the original forms of words is important for natural language processing applications. The goal of normalization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form, for instance

$$am,\ are,\ is \rightarrow be$$
$$car,\ cars,\ car's \rightarrow car$$

Normalization can involve either lemmatization or stemming.

Stemming usually refers to a crude heuristic process that chops off the ends of words, and removal of derivational affixes. As result of the process we get a stem that does not have to be a proper word. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. Therefore stemmers are much simpler, smaller and usually faster than lemmatizers, and for many applications their results are good enough.

Usually low resource languages lack NLP tools like stemmer/lemmatizer. There is no normalizer for Buryat language, however several techniques were proposed for Mongolian language [Fujii & Chimeddorj, 2012]. It is extremely important to develop normalizer for low resource language in order to alleviate language variability.

4.1. Yet Another Suffix Striper (YASS)

YASS is a statistical corpus-based stemmer that does not rely on linguistic expertise. It stems by clustering a lexicon without any linguistic input. Its performance is comparable to that obtained using standard rule-based stemmers such as Porter's. Information retrieval experiments done on English, French and Bengali datasets found YASS very effective [Majumder, et al., 2007].

The clusters are created using hierarchical approach and distance measures. Four distance functions D_1, D_2, D_3, D_4 were proposed. The main intuition behind defining these distances was to reward long matching prefixes, and to penalize an early mismatch. The D_3 distance function was found to be the most effective, so we focused on D_3 solely [Majumder, et al., 2007].

If the strings X and Y are of unequal lengths we pad the shorter string with null characters to make the strings lengths equal. The distance D_3 between two strings $X = x_0 x_1 \dots x_n$ and $Y = y_0 y_1 \dots y_n$ is as following

$$D_3(X,Y) = \begin{cases} \dfrac{n-m+1}{m} \displaystyle\sum_{i=m}^{n} \frac{1}{2^{i-m}} \ if\ m > 0 \\ \infty\ otherwise \end{cases} \tag{6}$$

where m denotes the position of the first mismatch between X and Y.

The distance function defined above is used to compose a distance matrix. Then the distance matrix is used to cluster words. Each cluster is expected to represent morphological variants of a single root word. The words within a cluster are stemmed to the "central" word in that cluster.

Three variants of hierarchical clustering algorithms were tested, namely, single-linkage, average-linkage and complete-linkage. In single-linkage clustering the similarity between two clusters is the maximal similarity between any two members of the groups. Complete-linkage clustering is similar to single-linkage, but instead of maximal similarity, it considers the minimal similarity between any two members as a clusters similarity. In average-linkage clustering the similarity between two clusters is the mean similarity between members of different clusters [Jain, Murty, & Flynn, 1999].

5. Experimental Setup

Section 5.1 describes the Buryat Wikipedia corpus. Section 5.2 specifies the methods that were used to learn the word-vector representations. Finally, the evaluation scheme is defined in Section 5.3.

5.1. Corpus

The models were trained on the Buryat Wikipedia, which consist of 1381 articles (each one is more than 50 words long). The articles were lower-cased and non-textual were removed. In addition, we excluded all words that contain non-Buryat characters. As result the corpus contains 406715 words (64403 unique words).

5.2. Training Embeddings

The models were derived using windows of 2, 5, 10 tokens to each side of the focus word. For every window size we calculated PMI[4] word representations and we learned a 50, 100, 500-dimensional representations with SVD, SGNS, CBOW and GloVe methods.

5.3. Evaluation Datasets

Several datasets have been used for evaluating word-vector representations. Among them RG [Rubenstein & Goodenough, 1965], WordSim-353 [Finkelstein, et al., 2001], WS-Sim [Agirre, et al., 2009] and MEN [Bruni, Boleda, Baroni, & Tran, 2012]. Each of these datasets consists of word pairs with corresponding similarity scores assigned by human annotators. A model is evaluated by assigning a similarity score to each pair and calculating the correlation (Spearman's ρ) with the human ranking.

However, these datasets suffer from some common shortcomings they have: associations of dissimilar words, low inter-rater agreement over the annotators [Hill, Reichart, & Korhonen, 2016]. In addition, more fundamental problems were pointed out. In some cases the use of rating scales might lead to a variety of annotations biases. In addition, different relations were rated by the same scale and different target-words were rated on the same scale, e.g.: (cat, pet) vs. (winter, season). The mentioned problems were addressed by the method proposed in [Avraham & Goldberg, 2016], however this method requires extensive human annotations.

[4] To calculate PMI matrices we used COMPOSES by [Baroni, Bernardi, & Zamparelli, 2014]

We proposed a simple evaluation scheme that was inspired by [Avraham & Goldberg, 2016], however it does not require extensive human annotations. In addition, our evaluation method can be easily adapted to any language.

In order to evaluate the word-vector representations we picked 32 nouns (hypernyms) with corresponding hyponyms (from two to five for every hypernym). In hypernym-hyponym pairs, the target word (hypernym) with corresponding hyponyms were used to measure the positive pairs of the preferred relations, to measure the negative pairs we used the target words with hyponyms from the different hypernym. To simplify the process, we did not use human annotators to assign similarity scores, the similarity between positive pairs was set to 1 and the similarity between negative pairs was set to 0.

Finally, as a result we calculated Spearman correlation between gold standard 0–1 vector and the vector of cosine similarities calculated in accordance to the tested model.

6. Results

We begin by identifying the best possible settings for stemmer including clustering algorithm and a threshold (Section 6.1). Section 6.2 compares the methods for learning word-vector representations.

6.1. Word Normalization

To find the best performing combination of clustering method and the threshold Θ we ran number of experiments[5]. The preliminary results show that complete-linkage and average-linkage approaches highly outperformed the single-average clustering, so we omit results for the single-average clustering.

According to the evaluation scheme there are *88* positive and *82* negative pairs for hypernym-hyponym relation.

As a baseline approach for comparison we used PMI. The PMI performance score without the stemming was *0.515* for hypernym-hyponym relation. The average-linkage clustering outperformed the complete-linkage clustering. The average-linkage clustering achieves its best results at $\Theta = 1.5$ for hypernym-hyponym relation.

[5] The clustering was performed by fastcluster 1.1.24 [Mullner, 2013]

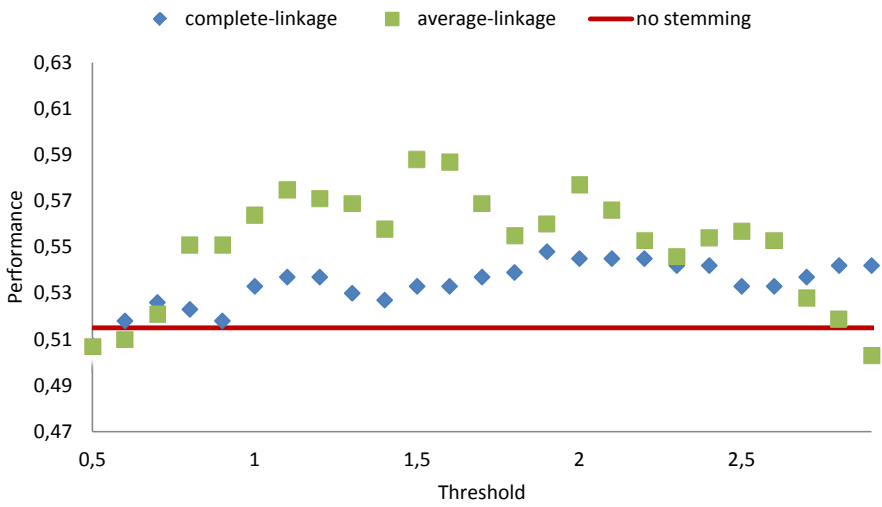

Fig. 1. Evaluation results for various clustering methods and thresholds for hypernym-hyponym relation

6.2. Word-Vector Representations

The performance of the different word embedding methods is in Table 1. Almost in all cases bigger window size leads to better results. Stemming (based on average-linkage with fine-tuned threshold θ) considerably improves word embedding performance.

Table 1. Performance of each method for different settings

Method	win dim	hypernym-hyponym		
		2	5	10
PMI no stemming	—	.517	.510	.528
PMI	—	.585	.588	.611
PMI smoothed	—	.555	.571	.599
SVD	50	.638	.663	.690
	100	.632	.641	**.722**
	500	.612	.662	.691
W2V CBOW	50	.022	−.006	.042
	100	−.003	−.015	.038
	500	−.024	−.033	.011
W2V SGNS	50	.064	.146	.293
	100	.043	.136	.290
	500	.061	.150	.280
GloVe	50	.115	.262	.363
	100	.124	.267	.363
	500	.127	.267	.390

Our findings confirm that SGNS outperforms CBOW on small datasets [Mikolov, Le, & Sutskever, 2013]. In addition, it justifies that Skip-Grams approach works much better on the semantic tasks [Mikolov, Chen, Corrado, & Dean, 2013].

Surprisingly, smoothed variation of PMI (with $\alpha = 0.75$) that was shown to outperform traditional PMI on English Wikipedia corpus [Levy, Goldberg, & Dagan, 2015], lost to traditional PMI when small corpus was used.

To reduce dimensionality we used SVD factorization on PMI matrices after stemming. As expected, SVD factorization outperformed PMI matrices performance in all modes. However, weighted SVD ($d = 0; 0.5$) did not improve the performance further.

Both fails of the count-based methods' enhancements (smoothed PMI and weighted SVD) can be caused by the small size of the dataset.

In addition, traditional count-based methods notably outperformed neural based methods in all settings, which contradicts with the results obtained on big datasets [Baroni, Dinu, & Kruszewski, 2014].

7. Conclusion and Future Work

In this paper we compared the capabilities of traditional count-based methods and neural embeddings methods to learn accurate word-vector representations in small text corpora (on the case of the Buryat Wikipedia). We found that traditional count-based methods outperform neural-based methods when the models are trained on small dataset. We believe that word2vec performance decrease in small corpora was caused by the fact that neural-based models need a lot of training data in order to fit their high number of parameters.

We found that the tweaks (smoothed PMI and weighted SVD) that were found to improve performance on big text corpora [Levy, Goldberg, & Dagan, 2015] did not outperform the traditional PMI and SVD in our case. These fails can be caused by a small size of the dataset. Therefore, future work should carefully explore the influence of the hyperparameters on the quality of the word-vector representations.

We found that language independent stemming approach (with tuned hyperparameters) can considerably improve word embeddings quality.

In addition, we proposed a coarse but easily reproducible word embedding evaluation scheme.

To promote further research, we made our code freely available[6].

8. Acknowledgements

We thank Anton Alexeev, Anna Potapenko, Andrey Kutuzov and Dmitry Ustalov for their assistance and contribution.

[6] https://github.com/vaskonov/burvec

References

1. *Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pacsca, M., Soroa, A..., et al.* (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 19–27.
2. *Altszyler, E., Sigman, M., Ribeiro, S., & Slezak, D. F.* (2016). Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. arXiv preprint arXiv:1610.01520.
3. *Avraham, O., & Goldberg, Y.* (2016). Improving reliability of word similarity evaluation by redesigning annotation task and performance measure. arXiv preprint arXiv:1611.03641.
4. *Badagarov, J., Trosterud, T., & Tyers, F.* (2016). Language Documentation and Language Technologies for Circumpolar Region.
5. *Badmaeva, E., & Francis, T.* (2017). A Dependency Treebank for Buryat. TLT, 1–17.
6. *Baroni, M., Bernardi, R., & Zamparelli, R.* (2014). Frege in space: A program of compositional distributional semantics. Linguistic Issues in Language Technology.
7. *Baroni, M., Dinu, G., & Kruszewski, G.* (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 238–247.
8. *Bruni, E., Boleda, G., Baroni, M., & Tran, N.-K.* (2012). Distributional semantics in technicolor. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, 136–145.
9. *Chen, L.-Y., & Chen, S.-M.* (2007). A new approach for automatic thesaurus construction and query expansion for document retrieval. International Journal of Information and Management Sciences.
10. *Church, K. W., & Hanks, P.* (1990). Word association norms, mutual information, and lexicography. Computational linguistics, 22–29.
11. *Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R.* (1990). Indexing by latent semantic analysis. Journal of the American society for information science.
12. *Di Marco, A., & Navigli, R.* (2013). Clustering and diversifying web search results with graph-based word sense induction. Computational Linguistics, 709–754.
13. *Eckart, C., & Young, G.* (1936). The approximation of one matrix by another of lower rank. Psychometrika, 211–218.
14. *Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., et al.* (2001). Placing search in context: The concept revisited. Proceedings of the 10th international conference on World Wide Web, 406–414.
15. *Fujii, O., & Chimeddorj, A.* (2012). Enhancing Lemmatization for Mongolian and its Application to Statistical Machine Translation. 24th International Conference on Computational Linguistics, 115.
16. *Goldberg, Y.* (2016). A Primer on Neural Network Models for Natural Language Processing. Journal of Artificial Intelligence Research, 345–420.
17. *Harris, Z.* (1954). Distributional structure. Word, 146–162.

18. *Hill, F., Reichart, R., & Korhonen, A.* (2016). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. Computational Linguistics.
19. *Jain, A., Murty, N., & Flynn, P.* (1999). Data clustering: a review. ACM computing surveys, 264–323.
20. *Janhunen, J.* (2006). The Mongolic Languages.
21. *Jivani, A. G.* (2011). A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl, 1930–1938.
22. *Khaltar, B.-O., & Fujii, A.* (2008). A Lemmatization Method for Modern Mongolian and its Application to Information Retrieval. IJCNLP, 1–8.
23. *language, B.* (2017). Buryat language—Wikipedia, The Free Encyclopedia. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Buryat_language
24. *Levy, O., & Goldberg, Y.* (2014). Dependency-Based Word Embeddings. ACL, 302–308.
25. *Levy, O., Goldberg, Y., & Dagan, I.* (2015). Improving distributional similarity with lessons learned from word embeddings. Transactions of the Association for Computational Linguistics, 211–225.
26. *Majumder, P., Mitra, M., Parui, S., Kole, G., Mitra, P., Datta, K..., et al.* (2007). YASS: Yet another suffix stripper. ACM transactions on information systems (TOIS), 18.
27. *Marco, B., Georgiana, D., & German, K.* (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. ACL.
28. *Melamud, O., McClosky, D., Patwardhan, S., & Bansal, M.* (2016). The role of context types and dimensionality in learning word embeddings. arXiv preprint arXiv:1601.00893.
29. *Mikolov, T., Chen, K., Corrado, G., & Dean, J.* (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
30. *Mikolov, T., Le, Q., & Sutskever, I.* (2013). Exploiting similarities among languages for machine translation. arXiv preprint arXiv:1309.4168.
31. *Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J.* (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 3111--3119.
32. *Mullner, D.* (2013). fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. Journal of Statistical Software, 1–18.
33. *Pennington, J., Socher, R., & Manning, C.* (2014). Glove: Global vectors for word representation. Proceedings of the 2014 conference on empirical methods in natural language processing, 1532–1543.
34. *Porter, M.* (1980). An algorithm for suffix stripping. Program, 130–137.
35. *Rubenstein, H., & Goodenough, J.* (1965). Contextual correlates of synonymy. Communications of the ACM, 627–633.
36. *Rumelhart, D., Hinton, G., & Williams, R.* (1986). Learning representations by back-propagating errors. Nature, 60–88.
37. *Turney, P., & Pantel, P.* (2010). From frequency to meaning: Vector space models of semantics. Journal of artificial intelligence research, 141–188.
38. *Utsumi, A.* (2014). A semantic space approach to the computational semantics of noun compounds. Natural Language Engineering, 185–234.
39. *Zhu, Z., Li, M., Chen, L., & Yang, Z.* (2013). Building Comparable Corpora Based on Bilingual LDA Model. ACL, 278–282.