

Matrix Factorization using Window Sampling and Negative Sampling for Improved Word Representations

Alexandre Salle¹ Marco Idiart² Aline Villavicencio¹

¹ Institute of Informatics

² Physics Department

Universidade Federal do Rio Grande do Sul

Porto Alegre, Brazil

{atsalle, avillavicencio}@inf.ufrgs.br, idiart@if.ufrgs.br

Abstract

In this paper, we propose LexVec, a new method for generating distributed word representations that uses low-rank, weighted factorization of the Positive Point-wise Mutual Information matrix via stochastic gradient descent, employing a weighting scheme that assigns heavier penalties for errors on frequent co-occurrences while still accounting for negative co-occurrence. Evaluation on word similarity and analogy tasks shows that LexVec matches and often outperforms state-of-the-art methods on many of these tasks.

1 Introduction

Distributed word representations, or word embeddings, have been successfully used in many NLP applications (Turian et al., 2010; Collobert et al., 2011; Socher et al., 2013). Traditionally, word representations have been obtained using *count-based* methods (Baroni et al., 2014), where the co-occurrence matrix is derived directly from corpus counts (Lin, 1998) or using association measures like Point-wise Mutual Information (PMI) (Church and Hanks, 1990) and Positive PMI (PPMI) (Bullinaria and Levy, 2007; Levy et al., 2014).

Techniques for generating lower-rank representations have also been employed, such as PPMI-SVD (Levy et al., 2015) and GloVe (Pennington et al., 2014), both achieving state-of-the-art performance on a variety of tasks.

Alternatively, vector-space models can be generated with *predictive* methods, which generally outperform the count-based methods (Baroni et al., 2014), the most notable of which is Skip-gram with Negative Sampling (SGNS, Mikolov

et al. (2013b)), which uses a neural network to generate embeddings. It implicitly factorizes a shifted PMI matrix, and its performance has been linked to the weighting of positive and negative co-occurrences (Levy and Goldberg, 2014).

In this paper, we present Lexical Vectors (LexVec), a method for factorizing PPMI matrices that combines characteristics of all these methods. On the one hand, it uses SGNS window sampling, negative sampling, and stochastic gradient descent (SGD) to minimize a loss function that weights frequent co-occurrences heavily but also takes into account negative co-occurrence. However, since PPMI generally outperforms PMI on semantic similarity tasks (Bullinaria and Levy, 2007), rather than implicitly factorize a shifted PMI matrix (like SGNS), LexVec explicitly factorizes the PPMI matrix.

This paper is organized as follows: First, we describe PPMI-SVD, GloVe, and SGNS (§2) before introducing the proposed method, LexVec (§3), and evaluating it on word similarity and analogy tasks (§4). We conclude with an analysis of results and discussion of future work.

We provide source code for the model at <https://github.com/alexandres/lexvec>.

2 Related Work

2.1 PPMI-SVD

Given a word w and a symmetric window of win context words to the left and win to the right, the co-occurrence matrix of elements M_{wc} is defined as the number of times a target word w and the context word c co-occurred in the corpus within the window. The PMI matrix is defined as

$$PMI_{wc} = \log \frac{M_{wc} M_{**}}{M_{w*} M_{*c}} \quad (1)$$

where ‘*’ represents the summation of the corresponding index. As this matrix is unbounded in the inferior limit, in most applications it is replaced by its positive definite version, PPMI, where negative values are set to zero. The performance of the PPMI matrix on word similarity tasks can be further improved by using *context-distribution smoothing* (Levy et al., 2015) and *sub-sampling* the corpus (Mikolov et al., 2013b). As word embeddings with lower dimensionality may improve efficiency and generalization (Levy et al., 2015), the improved PPMI* matrix can be factorized as a product of two lower rank matrices.

$$PPMI_{wc}^* \simeq W_w \tilde{W}_c^\top \quad (2)$$

where W_w and \tilde{W}_c are d -dimensional row vectors corresponding to vector embeddings for the target and context words. Using the truncated SVD of size d yields the factorization $U\Sigma T^\top$ with the lowest possible L_2 error (Eckert and Young, 1936).

Levy et al. (2015) recommend using $W = U\Sigma^p$ as the word representations, as suggested by Bullinaria and Levy (2012), who borrowed the idea of weighting singular values from the work of Caron (2001) on Latent Semantic Analysis. Although the optimal value of p is highly task-dependent (Österlund et al., 2015), we set $p = 0.5$ as it has been shown to perform well on the word similarity and analogy tasks we use in our experiments (Levy et al., 2015).

2.2 GloVe

GloVe (Pennington et al., 2014) factors the logarithm of the co-occurrence matrix \hat{M} that considers the position of the context words in the window. The loss function for factorization is

$$L_{wc}^{GloVe} = \frac{1}{2} f(\hat{M}_{wc})(W_w \tilde{W}_c^\top + b_w + \tilde{b}_c - \log \hat{M}_{wc})^2 \quad (3)$$

where b_w and \tilde{b}_c are bias terms, and f is a weighting function defined as

$$f(x) = \begin{cases} (x/x_{max})^\beta & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

W and \tilde{W} are obtained by iterating over all non-zero (w, c) cells in the co-occurrence matrix and minimizing eq. (3) through SGD.

The weighting function (in eq. (3)) penalizes more heavily reconstruction error of frequent co-occurrences, improving on PPMI-SVD’s L_2 loss,

which weights all reconstruction errors equally. However, as it does not penalize reconstruction errors for pairs with zero counts in the co-occurrence matrix, no effort is made to scatter the vectors for these pairs.

2.3 Skip-gram with Negative Sampling (SGNS)

SGNS (Mikolov et al., 2013b) trains a neural network to predict the probability of observing a context word c given a target word w , sliding a symmetric window over a subsampled training corpus with the window size being sampled uniformly from the range $[1, win]$. Each observed (w, c) pair is combined with k randomly sampled noise pairs (w, w_i) and used to calculate the loss function

$$L_{wc}^{SGNS} = \log \sigma(W_w \tilde{W}_c^\top) + \sum_{i=1}^k \mathbf{E}_{w_i \sim P_n(w)} \log \sigma(-W_w \tilde{W}_{w_i}^\top) \quad (5)$$

where $P_n(w)$ is the distribution from which noise words w_i are sampled.¹ We refer to this routine which SGNS uses for selecting (w, c) pairs by sliding a context window over the corpus for loss calculation and SGD as *window sampling*.

SGNS is implicitly performing the weighted factorization of a shifted PMI matrix (Levy and Goldberg, 2014). Window sampling ensures the factorization weights frequent co-occurrences heavily, but also takes into account negative co-occurrences, thanks to negative sampling.

3 LexVec

LexVec is based on the idea of factorizing the PPMI matrix using a reconstruction loss function that does not weight all errors equally, unlike SVD, but instead penalizes errors of frequent co-occurrences more heavily, while still treating negative co-occurrences, unlike GloVe. Moreover, given that using PPMI results in better performance than PMI on semantic tasks, we propose keeping the SGNS weighting scheme by using window sampling and negative sampling, but explicitly factorizing the PPMI matrix rather than implicitly factorizing the shifted PMI matrix. The LexVec loss function has two terms

¹Following Mikolov et al. (2013b) it is the unigram distribution raised to the 3/4 power.

$$L_{wc}^{LexVec} = \frac{1}{2} (W_w \tilde{W}_c^\top - PPMI_{wc}^*)^2 \quad (6)$$

$$L_w^{LexVec} = \frac{1}{2} \sum_{i=1}^k \mathbf{E}_{w_i \sim P_n(w)} (W_w \tilde{W}_{w_i}^\top - PPMI_{ww_i}^*)^2 \quad (7)$$

We minimize eqs. (6) and (7) using two alternative approaches:

Mini-Batch (MB): This variant executes gradient descent in exactly the same way as SGNS. Every time a pair (w, c) is observed by window sampling and pairs $(w, w_{1\dots k})$ drawn by negative sampling, W_w , \tilde{W}_c , and $\tilde{W}_{w_{1\dots k}}$ are updated by gradient descent on the sum of eq.(6) and eq.(7). The global loss for this approach is

$$L^{LexVec} = \sum_{(w,c)} \#(w, c) (L_{wc}^{LexVec} + L_w^{LexVec}) \quad (8)$$

where $\#(w, c)$ is the number of times (w, c) is observed in the subsampled corpus.

Stochastic (St): Every context window is extended with k negative samples $w_{1\dots k}$. Iterative gradient descent of eq. (6) is then run on pairs (w, c_j) , for $j = 1, \dots, 2*win$ and (w, c_i) , $j = 1, \dots, k$ for each window. The global loss for this approach is

$$L^{LexVec'} = \sum_{(w,c)} \#(w, c) L_{wc}^{LexVec} + \sum_w \#(w) L_w^{LexVec} \quad (9)$$

where $\#(w)$ is the number of times w is observed in the subsampled corpus.

If a pair (w, c) co-occurs frequently, $\#(w, c)$ will weigh heavily in both eqs. (8) and (9), giving the desired weighting for frequent co-occurrences. The noise term, on the other hand, has corrections proportional to $\#(w)$ and $\#(w_i)$, for each pair (w, w_i) . It produces corrections in pairs that due to frequency should be in the corpus but are not observed, therefore accounting automatically for negative co-occurrences.

4 Materials

All models were trained on a dump of Wikipedia from June 2015, split into sentences, with punctuation removed, numbers converted to words, and lower-cased. Words with less than 100 counts were removed, resulting in a vocabulary of

302,203 words. All models generate embeddings of 300 dimensions.

The PPMI* matrix used by both PPMI-SVD and LexVec was constructed using smoothing of $\alpha = 3/4$ suggested in (Levy et al., 2015) and an unweighted window of size 2. A dirty subsampling of the corpus is adopted for PPMI* and SGNS with threshold of $t = 10^{-5}$ (Mikolov et al., 2013b).² Additionally, SGNS uses 5 negative samples (Mikolov et al., 2013b), a window of size 10 (Levy et al., 2015), for 5 iterations with initial learning rate set to the default 0.025. GloVe is run with a window of size 10, $x_{max} = 100$, $\beta = 3/4$, for 50 iterations and initial learning rate of 0.05 (Pennington et al., 2014).

In LexVec two window sampling alternatives are compared: WS_{PPMI} , which keeps the same fixed size $win = 2$ as used to create the $PPMI^*$ matrix; or WS_{SGNS} , which adopts identical SGNS settings ($win = 10$ with size randomization). We run LexVec for 5 iterations over the training corpus.

All methods generate both word and context matrices (W and \tilde{W}): W is used for SGNS, PPMI-SVD and $W + \tilde{W}$ for GloVe (following Levy et al. (2015), and W and $W + \tilde{W}$ for LexVec.

For evaluation, we use standard word similarity and analogy tasks (Mikolov et al., 2013b; Levy et al., 2014; Pennington et al., 2014; Levy et al., 2015). We examine, in particular, if LexVec weighted PPMI* factorization outperforms SVD, GloVe (weighted factorization of $\log \hat{M}$) and Skip-gram (implicit factorization of the shifted PMI matrix), and compare the stochastic and mini-batch approaches.

Word similarity tasks are:³ WS-353 Similarity (WSim) and Relatedness (WRel) (Finkelstein et al., 2001), MEN (Bruni et al., 2012), MTurk (Radinsky et al., 2011), RW (Luong et al., 2013), SimLex-999 (Hill et al., 2015), MC (Miller and Charles, 1991), RG (Rubenstein and Goodenough, 1965), and SCWS (Huang et al., 2012), calculated using cosine. Word analogy tasks are: Google semantic (GSem) and syntactic (GSyn) (Mikolov et al., 2013a) and MSR syntactic analogy dataset (Mikolov et al., 2013c), using $3CosAdd$ and $3CosMul$ (Levy et al., 2014).

²Words with unigram relative frequency $f > t$ are discarded from the training corpus with probability $p_w = 1 - \sqrt{t/f}$.

³<http://www.cs.cmu.edu/mfaruqui/suite.html>

Method	WSim	WRel	MEN	MTurk	RW	SimLex-999	MC	RG	SCWS
PPMI-SVD	.731	.617	.731	.627	.427	.303	.770	.756	.615
GloVe	.719	.607	.736	.643	.400	.338	.725	.774	.573
SGNS	.770	.670	.763	.675	.465	.339	.823	.793	.643
LexVec + MB + $WS_{PPMI} + (W + \tilde{W})$.770	.671	.755	.650	.455	.322	.824	.830	.623
LexVec + St. + $WS_{PPMI} + (W + \tilde{W})$.763	.671	.760	.655	.458	.336	.816	.827	.630
LexVec + MB + $WS_{PPMI} + W$.748	.635	.741	.636	.456	.320	.827	.820	.632
LexVec + St. + $WS_{PPMI} + W$.741	.622	.733	.628	.457	.338	.820	.808	.638
LexVec + MB + $WS_{SGNS} + (W + \tilde{W})$.768	.675	.755	.654	.448	.312	.824	.827	.626
LexVec + St. + $WS_{SGNS} + (W + \tilde{W})$.775	.673	.762	.654	.468	.339	.838	.848	.628
LexVec + MB + $WS_{SGNS} + W$.745	.640	.734	.645	.447	.311	.814	.802	.624
LexVec + St. + $WS_{SGNS} + W$.740	.628	.728	.640	.459	.339	.821	.818	.638

Table 1: Spearman rank correlation on word similarity tasks.

Method	GSem		GSyn		MSR	
	3CosAdd	3CosMul	3CosAdd	3CosMul	3CosAdd	3CosMul
PPMI-SVD	.460	.498	.445	.455	.303	.313
GloVe	.818	.813	.630	.626	.539	.547
SGNS	.773	.777	.642	.644	.481	.505
LexVec + MB + $WS_{PPMI} + (W + \tilde{W})$.775	.792	.520	.539	.371	.413
LexVec + St. + $WS_{PPMI} + (W + \tilde{W})$.794	.807	.543	.555	.378	.408
LexVec + MB + $WS_{PPMI} + W$.800	.805	.584	.597	.421	.457
LexVec + St. + $WS_{PPMI} + W$.787	.782	.597	.613	.445	.475
LexVec + MB + $WS_{SGNS} + (W + \tilde{W})$.762	.785	.520	.534	.349	.386
LexVec + St. + $WS_{SGNS} + (W + \tilde{W})$.792	.809	.536	.553	.362	.396
LexVec + MB + $WS_{SGNS} + W$.798	.807	.573	.580	.399	.435
LexVec + St. + $WS_{SGNS} + W$.779	.778	.600	.614	.434	.463

Table 2: Results on word analogy tasks, given as percent accuracy.

5 Results

Results for word similarity and for the analogy tasks are in tables 1 and 2, respectively. Compared with PPMI-SVD, LexVec performs better in all tasks. As they factorize the same $PPMI^*$ matrix, it is the loss weighting from window sampling that is an improvement over L_2 loss. As expected, due to PPMI, LexVec performs better than SGNS in several word similarity tasks, but in addition it also does so on the semantic analogy task, nearly approaching GloVe. LexVec generally outperforms GloVe on word similarity tasks, possibly due to the factorization of the PPMI matrix and to window sampling’s weighting of negative co-occurrences.

We believe LexVec fares well on semantic analogies because its vector-space does a good job of preserving semantics, as evidenced by its performance on word similarity tasks. We believe the poor syntactic performance is a result of the PPMI measure. PPMI-SVD also struggled with syntactic analogies more than any other task. Levy et al. (2015) obtained similar results, and suggest that using positional contexts as done by Levy et al. (2014) might help in recovering syntactic analogies.

In terms of configurations, WS_{SGNS} performed marginally better than WS_{PPMI} . We hypothesize it is simply because of the additional computation. While W and $(W + \tilde{W})$ are roughly equivalent on word similarity tasks, W is better for analogies. This is inline with results for PPMI-SVD and SGNS models (Levy et al., 2015). Both mini-batch and stochastic approaches result in similar scores for all tasks. For the same parameter k of negative samples, the mini-batch approach uses $2 * win_{WS_{PPMI}}$ times more negative samples than stochastic when using WS_{PPMI} , and $win_{WS_{SGNS}}$ times more samples when using WS_{SGNS} . Therefore, the stochastic approach is more computationally efficient while delivering similar performance.

6 Conclusion and Future Work

In this paper, we introduced LexVec, a method for low-rank, weighted factorization of the PPMI matrix that generates distributed word representations, favoring low reconstruction error on frequent co-occurrences, whilst accounting for negative co-occurrences as well. This is in contrast with PPMI-SVD, which does no weighting, and GloVe, which only considers positive co-

occurrences. Finally, its PPMI factorization seems to better capture semantics when compared to the shifted PMI factorization of SGNS. As a result, it outperforms PPMI-SVD and SGNS in a variety of word similarity and semantic analogy tasks, and generally outperforms GloVe on similarity tasks.

Future work will examine the use of positional contexts for improving performance on syntactic analogy tasks. Moreover, we will explore further the hyper-parameter space to find globally optimal values for LexVec, and will experiment with the factorization of other matrices for developing alternative word representations.

Acknowledgments

This work has been partly funded by CAPES and by projects AIM-WEST (FAPERGS-INRIA 1706- 2551/13-7), CNPq 482520/2012-4, 312114/2015-0, “Simplificação Textual de Expressões Complexas”, sponsored by Samsung Eletrônica da Amazônia Ltda. under the terms of Brazilian federal law No. 8.248/91.

References

- Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don’t count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. volume 1, pages 238–247.
- Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 136–145.
- John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods* 39(3):510–526.
- John A Bullinaria and Joseph P Levy. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *Behavior research methods* 44(3):890–907.
- John Caron. 2001. Experiments with lsa scoring: Optimal rank and basis. In *Proceedings of the SIAM Computational Information Retrieval Workshop*. pages 157–169.
- Kenneth W. Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics* 16(1):22–29.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.
- C. Eckert and G. Young. 1936. The approximation of one matrix by another of lower rank. *Psych.* 1:211–218.
- Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppín. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*. ACM, pages 406–414.
- Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* .
- Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 873–882.
- Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*. pages 2177–2185.
- Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* pages 211–225.
- Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. *CoNLL-2014* page 171.
- Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *of the 36th and 17th , Volume 2*. Montreal, Quebec, Canada, pages 768–774.

- Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. *CoNLL-2013* 104.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. pages 3111–3119.
- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *HLT-NAACL*. pages 746–751.
- George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes* 6(1):1–28.
- Arvid Österlund, David Ödling, and Magnus Sahlgren. 2015. Factorization of latent variables in distributional semantic models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 227–231.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)* 12.
- Kira Radinsky, Eugene Agichtein, Evgeniy Gabilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*. ACM, pages 337–346.
- Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM* 8(10):627–633.
- Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *ACL (1)*. pages 455–465.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, pages 384–394.