The Art of Reproducible Machine Learning
A Survey of Methodology in Word Vector Experiments

Vít Novotný

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
witiko@mail.muni.cz
https://mir.fi.muni.cz/

Abstract. Since the seminal work of Mikolov et al. (2013), word vectors of log-bilinear SVMs have found their way into many NLP applications as an unsupervised measure of word relatedness. Due to the rapid pace of research and the publish-or-perish mantra of academic publishing, word vector experiments contain undisclosed parameters, which make them difficult to reproduce. In our work, we introduce the experiments and their parameters, compare the published experimental results with our own, and suggest default parameter settings and ways to make previous and future experiments easier to reproduce. We show that the lack of variable control can cause up to 24% difference in accuracy on the word analogy tasks.

Keywords: Machine learning, word vectors, word2vec, fastText, word analogy, reproducibility

1 Introduction

After a long reign of topic modeling, [9] log-bilinear SVMs have emerged as a faster[1] method for learning word representations, [18] which can also infer representations of unseen words using a subword model [2]. Word vectors produced by log-bilinear SVMs have found their way into many NLP applications, such as word similarity, word analogy, and language modeling [2] as well as dependency parsing [8, Section 5], word sense disambiguation [6], text classification [14], semantic text similarity [5], and information retrieval [22, Section 4].

Although the usefulness of word vectors is rarely disputed, their theoretical foundations were only later addressed by Levy and Goldberg (2014) [16]. In their later work, Levy at al. (2015) [17] have shown that several pre-processing steps and fixed parameters can have a significant impact on experimental results. In this work, we describe six new undisclosed parameters that make word vector experiments difficult to reproduce. We compare the published experimental results with our own and suggest improvements to reproducibility.

[1] The time complexity of streaming approximations of the sparse SVD method used in topic modeling scales quadratically with the word vector size, [31, Section 2.3] whereas log-bilinear SVMs are linear in both the word vector size and the corpus size.

Table 1. The accuracies (%) of word vectors on the word analogy tasks for various languages using only the n most frequent words as the word analogy candidates for different values of n. Three languages most affected by the variations of n are *highlighted*.

	Cs	De	Es	*Fi*	Fr	*Hi*	It	*Pl*	Pt	Zh
Grave et al. [10], $n = 2 \cdot 10^5$	69.9	72.9	65.4	*70.3*	73.6	*32.1*	69.8	*67.9*	66.7	78.4
Our results, $n = 2 \cdot 10^5$	70.7	73.4	65.6	*71.2*	73.7	*32.2*	73.0	*68.5*	67.0	78.5
Our results, $n = 3 \cdot 10^5$	68.9	73.3	65.6	*68.9*	73.2	*26.4*	72.0	*66.3*	65.7	78.5
Our results, $n = 1 \cdot 10^6$	65.4	70.4	63.4	*60.9*	71.9	*16.0*	68.3	*61.1*	61.3	78.3

2 Word Analogy Tasks

In their seminal work, Mikolov et al. (2013) [18, Section 4] introduced the English word analogy task, which measures the ability of word vectors to answer the question "Which word b' is to a' as a is to b?". In the following years, the English word analogy task has been translated to many languages, including Czech [28], German [15], Spanish [4], Finnish [30], Italian [1], Portuguese [12], Turkish [27,11] and simplified Chinese [7] as well as French, Hindi, and Polish [2].

Rogers et al. (2017) [26] discuss the many problems with the word analogy task, including the selection of word pairs, the significant impact of including/excluding the words a, b, and a' in the candidates for b', and the underlying assumption that word relations are unique or even symmetric. In this section, we discuss two undisclosed parameters of the word analogy task.

2.1 Using only the most frequent words

To make the evaluation less susceptible to rare words, Mikolov et al. (2013) [18] only considered the n most frequent words as the candidates for b'. However, the value of n changes between experiments and is usually undocumented: Mikolov et al. (2013) [18] used $n = 1 \cdot 10^6$, the reference implementation[2] defaults to $n = 3 \cdot 10^5$, and Grave et al. (2018) [10] used $n = 2 \cdot 10^5$. Mikolov et al. (2013) [20], Bojanowski et al. (2017) [2], and Mikolov et al. (2018) [19] did not disclose the value n they used, which makes their results difficult to reproduce.

To show how significant the value of n is, we reproduce[3] the results of Grave et al. (2018) [10, Table 4] with $n \in \{2 \cdot 10^5, 3 \cdot 10^5, 1 \cdot 10^6\}$. Table 1 shows that up to 16% of word analogy accuracy can depend on the value of n. The most affected languages are Hindi, French, and Polish, indicating small/noisy training data.

To make results reproducible, we suggest that all papers should report the value of n they used in their evaluation on the word analogy tasks. If unreported, the value should be assumed to be $3 \cdot 10^5$, which is the default in the reference implementation and in a popular implementation from the Gensim library[4] [25].

[2] https://github.com/tmikolov/word2vec (file compute-accuracy.c)
[3] https://github.com/mir-mu/reproducible-ml (file word-analogy.ipynb)
[4] https://github.com/rare-technologies/gensim (file gensim/models/keyedvectors.py, method evaluate_word_analogies, also discussed in issue #2999)

Table 2. The accuracies (%) of word vectors on the word analogy tasks for various languages and case transformations (upper-casing and lower-casing) with either the U.S. English locale or the corresponding locales for the word analogy task languages. Three languages most affected by the variations are *highlighted*.

	Cs	De	Es	*Fi*	*Fr*	Hi	It	Pl	Pt	*Tr*	Zh
Grave et al. [10]	69.9	72.9	65.4	*70.3*	*73.6*	32.1	69.8	67.9	66.7		78.4
Our res., no case tran.	69.9	74.9	63.9	*53.3*	*76.7*	32.2	71.9	71.4	67.5	*58.2*	78.5
Our res., u.c., U.S. En.	70.7	73.4	65.6	*71.2*	*73.7*	32.2	73.0	68.5	67.0	*57.0*	78.5
Our res., u.c., corres.	70.7	73.4	65.6	*71.2*	*73.7*	32.2	73.0	68.5	67.0	*61.0*	78.5
Our res., l.c., U.S. En.	70.7	73.4	65.6	*71.2*	*73.7*	32.2	73.0	68.5	67.0	*56.9*	78.5
Our res., l.c., corres.	70.7	73.4	65.6	*71.2*	*73.7*	32.2	73.0	68.5	67.0	*61.0*	78.5

2.2 Caseless matching

To make the evaluation less susceptible to case, the words a, b, a', and b' are all lower-cased in the experiments of Bojanowski et al. (2017) [2]. This is never mentioned in the published papers, only in the code of the reference implementation.[5][6] Another problem is that in Unicode, lower-casing is locale-sensitive:

1. Lower-casing maps I to ı in Turkish and Azari, and to i in other locales.

A popular implementation in Gensim[7] [25] uses upper-casing instead of lower-casing. However, Unicode case is neither bijective nor transitive:

2. Upper-casing maps ß to SS, and lower-casing maps SS to ss (not ß).

This introduces several uncontrolled variables to the evaluation, most importantly the locale and the case transformation used for caseless matching.

To show how significant the locale and the case transformation are, we reproduce[8] the results of Grave et al. (2018) [10, Table 4] with various case transformations, using either the U.S. English locale (`en_US.UTF-8`), or the corresponding locales of the word analogy tasks. Table 2 shows that up to 18% of word analogy accuracy can depend on the case transformations and the locale. The most affected languages are Finnish, Turkish, and French.

To make results reproducible, we suggest that all papers should report the case transformations and locales they used in their evaluation on the word analogy tasks. If unreported, the locale of the word analogy task should be assumed. For case transformation, we suggest using the locale-independent Unicode case-folding algorithm [29, Section 3.13] instead of lower- or upper-casing:

3. Case-folding maps I to i in all locales, although implementations such as ICU can map[9] I to ı for Turkish and Azari. Case-folding maps ß, SS, and ss to ss.

[5] https://github.com/facebookresearch/fastText (file get-wikimedia.sh)
[6] https://github.com/facebookresearch/fastText/blob/master/python/doc/examples/compute_accuracy.py (function process_question)
[7] https://github.com/rare-technologies/gensim (file gensim/models/keyedvectors.py, method evaluate_word_analogies, also discussed in issue #2999)
[8] https://github.com/mir-mu/reproducible-ml (file word-analogy.ipynb)
[9] https://github.com/unicode-org/icu (file icu4c/source/common/unistr_case.cpp, method UnicodeString::foldCase)

3 Multi-Word Expressions

In their work, Mikolov et al. (2013) [20, Section 4] introduced a phrasing algorithm for merging commonly co-occuring words into multi-word expressions. The algorithm forms phrases using unigram and bigram counts, using the following scoring formula, which is proportional to the non-normalized pointwise mutual information (NPMI) [3]:

$$\mathrm{score}(w_i, w_j) = \frac{\mathrm{count}(w_i w_j)}{\mathrm{count}(w_i) \cdot \mathrm{count}(w_j)}. \tag{1}$$

Mikolov et al. (2013) merged candidate bigrams $w_i w_j$ with $\mathrm{score}(w_i, w_j)$ above a threshold δ into phrases. Mikolov et al. (2018) [19, Section 2.3] further improved the algorithm by randomly merging only 50% of the candidate bigrams, and reached SOTA performance on the English word analogy task.

In this section, we discuss three undisclosed parameters of the phrasing algorithm and the differences between the reference implementation[10] and a popular implementation in Gensim[11] [25].

3.1 Thresholding the bigram scores

Neither Mikolov et al. (2013) nor Mikolov et al. (2018) disclosed the threshold δ they used for merging candidate bigrams into phrases, which makes their results difficult to reproduce. The reference implementation uses $\delta_{\text{reference}} = 100$ and a different formula:

$$\mathrm{score}_{\text{reference}}(w_i, w_j) = \frac{\mathrm{count}(w_i w_j) \cdot \mathrm{corpusSize}}{\mathrm{count}(w_i) \cdot \mathrm{count}(w_j)} \tag{2}$$

The implementation in Gensim uses $\delta_{\text{Gensim}} = 10$ and also a different formula:

$$\mathrm{score}_{\text{Gensim}}(w_i, w_j) = \frac{\mathrm{count}(w_i w_j) \cdot \mathrm{dictionarySize}}{\mathrm{count}(w_i) \cdot \mathrm{count}(w_j)} \tag{3}$$

Apparently, $\mathrm{score} \neq \mathrm{score}_{\text{reference}} \neq \mathrm{score}_{\text{Gensim}}$. Due to the Heaps' law [13], $\mathrm{dictionarySize} \approx \sqrt{\mathrm{corpusSize}}$, so we might assume $\mathrm{score}_{\text{Gensim}} \approx \sqrt{\mathrm{score}_{\text{ref.}}}$. Since $\delta_{\text{Gensim}} = \sqrt{\delta_{\text{reference}}}$, we might then conclude $\mathrm{score}_{\text{Gensim}} > \delta_{\text{Gensim}} \iff \sqrt{\mathrm{score}_{\text{reference}}} > \sqrt{\delta_{\text{reference}}}$. However, the assumption does not actually hold:

$$\mathrm{score}_{\text{Gensim}} \approx \frac{\mathrm{count}(w_i w_j) \cdot \sqrt{\mathrm{corpusSize}}}{\mathrm{count}(w_i) \cdot \mathrm{count}(w_j)} \neq \sqrt{\frac{\mathrm{count}(w_i w_j) \cdot \mathrm{corpusSize}}{\mathrm{count}(w_i) \cdot \mathrm{count}(w_j)}}. \tag{4}$$

To make results reproducible, we suggest that all papers should report the scoring formula and the value of δ they used for phrasing. If unreported, the $\mathrm{score}_{\text{reference}}$ scoring formula and the $\delta_{\text{reference}} = 100$ value should be assumed.

[10] https://github.com/tmikolov/word2vec (file word2phrase.c)
[11] https://github.com/rare-technologies/gensim (file gensim/models/phrases.py, class Phrases and function bigram_scorer)

3.2 Incremental threshold decay

Mikolov et al. (2013) [20, Section 4] and Mikolov et al. (2018) [19, Section 2.2] apply the phrasing algorithm iteratively to form longer multi-word expressions. Mikolov et al. (2013) use 2–4 iterations, whereas Mikolov et al. (2018) use 5–6 iterations. Mikolov et al. (2013) also reports using a decaying threshold to make it easier for longer phrases to form. However, neither Mikolov et al. (2013) nor Mikolov et al. (2018) disclosed the threshold decay function they used, which makes their results difficult to reproduce.

To make results reproducible, we suggest that all papers should report the exact number of iterations and the threshold decay function they used.

3.3 Maximum dictionary size

To make the phrasing algorithm less susceptible to rare words, Mikolov et al. (2013) [20] and Mikolov et al. (2018) [19] have only considered the n most frequent words for the candidate bigrams. This is mentioned only in the code of the reference implementation. Additionally, the reference implementation uses $n = 5 \cdot 10^8$, whereas a popular implementation in Gensim uses $n = 4 \cdot 10^7$.

To make results reproducible, we suggest that all papers should report the value of n they used for phrasing. If unreported, the value should be assumed to be $n = 5 \cdot 10^8$, which is the default in the reference implementation.

4 Positional Weighting

The cbow model of Mikolov et al. (2013) [18] is trained to predict a masked word in a context window P from the average $\mathbf{v}_C$ of the context word vectors $\mathbf{d}_p$:

$$\mathbf{v}_C = \frac{1}{|P|} \sum_{p \in P} \mathbf{d}_p. \tag{5}$$

In many sentences, the position of the context words is important for predicting the masked word. Consider the following two sentences, which produce an identical context vector $\mathbf{v}_C$, although the masked words are significantly different:

1. Unlike dogs, cats are ⟨*mask*⟩.
2. Unlike cats, dogs are ⟨*mask*⟩.

If the context window P is large, distant context words will also be unimportant for predicting the masked word.

To better model these scenarios, Mikolov et al. (2018) [19, Section 2.2] adopted the positional weighting of Mnih and Kavukcuoglu (2013) [21, Section 3], and reached sota performance on the English word analogy task. Positional weighting makes the average $\mathbf{v}_C$ into a weighted average $\mathbf{w}_C$, where the weight of a context word at a position p is the positional vector $\mathbf{u}_p$, and the weighting is carried out using the pointwise (Hadamard) vector product $\odot$:

$$\mathbf{w}_C = \frac{1}{|P|} \sum_{p \in P} \mathbf{d}_p \odot \mathbf{u}_p, \tag{6}$$

In this section, we discuss an undisclosed parameter of positional weighting.

4.1 Positional weight initialization

In the CBOW model of Mikolov et al. (2013) [18], the word vectors $\mathbf{d}_p$ are initialized to a random sample of the continuous uniform distribution $\mathcal{U}(\pm\frac{1}{2D})$, where D is the dimensionality of the word vectors. Word vector initialization is an important parameter that affects the gradient size and therefore the effective learning rate. However, it is never mentioned in the published papers, only in the code of the reference implementation.[12] In the subword CBOW model of Bojanowski et al. (2017) [2], the initialization changes to $\mathbf{d}_p \sim \mathcal{U}(\pm\frac{1}{D})$ in the code of the reference implementation.[13] Mikolov et al. (2018) [19] do not describe the initialization of the word vectors $\mathbf{d}_p$ or the positional vectors $\mathbf{u}_p$. Since no reference implementation exists either, their results are difficult to reproduce.

To show how significant initialization is, we describe several initialization options for the word vectors $\mathbf{d}_p$ and the positional vectors $\mathbf{u}_p$. We then use the initializations to reproduce[14] the results of Mikolov et al. (2018) [19, Table 2] using the subword CBOW model of Bojanowski et al. (2017) [2] and the 2017 English Wikipedia[15] training corpus (4% of the Common Crawl dataset used by Mikolov et al., 2018) without phrasing. We report English word analogy scores using the $n = 2 \cdot 10^5$ most frequent words, and the case-folding case transformation.

Same as vanilla word vectors The simplest option is to use the initialization of the word vectors $\mathbf{d}_p$ also for the positional vectors $\mathbf{u}_p$: $\mathbf{d}_p \sim \mathbf{u}_p \sim \mathcal{U}(\pm\frac{1}{D})$. In practice, this causes $\mathbf{v}_C \gg \mathbf{w}_C$, decreasing the learning rate (see Figure 1).

Identity positional vectors To ensure $\mathbf{v}_C \sim \mathbf{w}_C$, the simplest option is to initialize the word vectors $\mathbf{d}_p$ to $\mathcal{U}(\pm\frac{1}{D})$ and the positional vectors $\mathbf{u}_p$ to $\mathbf{1}$. Intuitively, the training starts with no positional weighting and positional vectors are learnt later. In practice, $\mathbf{d}_p \ll \mathbf{u}_p$, causing the gradient updates of $\mathbf{d}_p$ to explode for dimensionality $D > 600$. This leads to instability, causing $\mathbf{d}_p = \mathbf{u}_p = \text{NaN}$.

Same as word vectors To ensure $\mathbf{v}_C \sim \mathbf{w}_C$ and $\mathbf{d}_p \sim \mathbf{u}_p$, we require a square distribution $\mathcal{U}^{0.5}(\pm\frac{1}{D})$ such that for i.i.d. $\mathbf{d}_p, \mathbf{u}_p : \mathbf{d}_p \sim \mathbf{u}_p \sim \mathcal{U}^{0.5}(\pm\frac{1}{D})$, we get $\mathbf{d}_p \odot \mathbf{u}_p \sim \mathcal{U}(\pm\frac{1}{D})$. Although an empirical approximate of $\mathcal{U}^{0.5}(0,1)$ using the β-distribution is known [24] (see Figure 2), this does not help with $\mathcal{U}^{0.5}(\pm\frac{1}{D})$, so we need a different approach: If we assume that the context window P is sufficiently large, then $\mathbf{v}_C \sim \mathrm{N}(\mu, \frac{\sigma^2}{|P|})$ by the CLT, where $\mu = \mathrm{E}[\mathcal{U}(\pm\frac{1}{D})] = 0$ and $\sigma^2 = \mathrm{Var}[\mathcal{U}(\pm\frac{1}{D})] = \frac{1}{6D^2}$. For $\mathbf{v}_C \sim \mathbf{w}_C$, we need a distribution X such that $\mathbf{d}_p \sim \mathbf{u}_p \sim X, \mathrm{E}[X^2] = \mu, \mathrm{Var}[X^2] = \sigma^2$. $\mathrm{E}[X^2] = \mu = 0$ leads to $\mathrm{E}[X] = 0$ and $\mathrm{Var}[X^2] = \mathrm{Var}[X]^2$, leading to $\mathrm{Var}[X] = \sigma$. We tested two such X: the uniform $\mathcal{U}\left(\pm\frac{\sqrt[4]{3}}{\sqrt{D}}\right)$ (see Figure 3) and the square-normal $\mathrm{N}^{0.5}\left(0, \frac{1}{\sqrt{6}D}\right)$ [23] (see Figure 4).

[12] https://github.com/tmikolov/word2vec (file word2vec.c, function InitNet)
[13] https://github.com/facebookresearch/fastText (file src/fasttext.cc)
[14] https://github.com/mir-mu/reproducible-ml (file positional-weighting.ipynb)
[15] https://github.com/rare-technologies/gensim-data (release wiki-english-20171001)

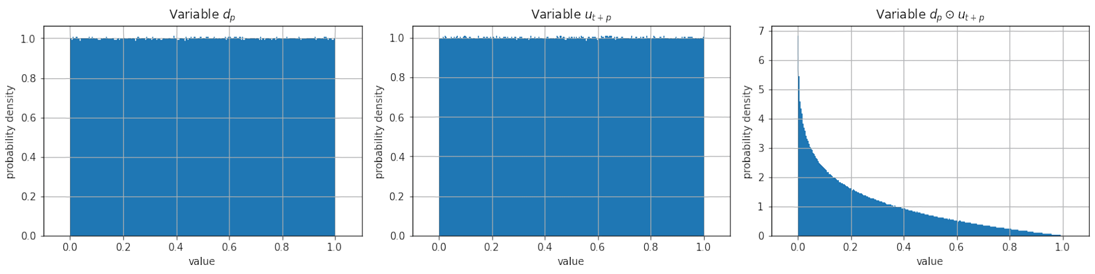

Fig. 1. Probability density functions of values in word vectors $\mathbf{d}_p$ (left), positional vectors $\mathbf{u}_p$ (middle), and their Hadamard products $\mathbf{d}_p \odot \mathbf{u}_p$ (right) with the *same as vanilla word vectors* initialization to $\mathcal{U}(0, 1)$. Since $\mathbf{v}_C$ is the average $\mathbf{d}_p$, and $\mathbf{w}_C$ is the average $\mathbf{d}_p \odot \mathbf{u}_p$, we conclude that $\mathbf{v}_C \gg \mathbf{w}_C$, decreasing the learning rate of positional weighting.

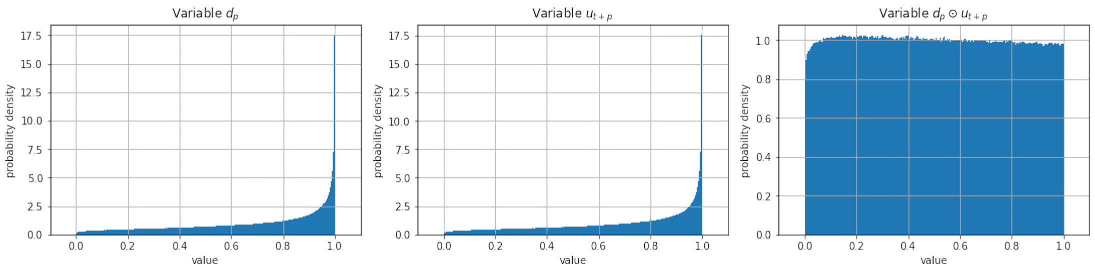

Fig. 2. Probability density functions of values in word vectors $\mathbf{d}_p$ (left), positional vectors $\mathbf{u}_p$ (middle), and their Hadamard products $\mathbf{d}_p \odot \mathbf{u}_p$ (right) with the unused initialization to an empirical approximation of $\mathcal{U}^{0.5}(0, 1)$. We need $\mathcal{U}^{0.5}(\pm 1)$ instead of $\mathcal{U}^{0.5}(0, 1)$.

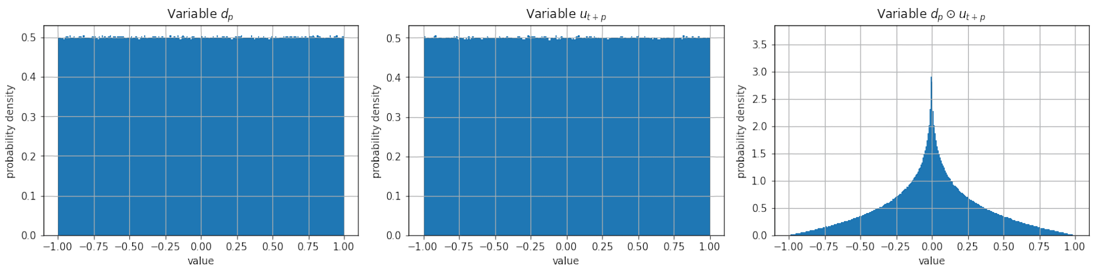

Fig. 3. Probability density functions of values in word vectors $\mathbf{d}_p$ (left), positional vectors $\mathbf{u}_p$ (middle), and their Hadamard products $\mathbf{d}_p \odot \mathbf{u}_p$ (right) with the *same as vanilla word vectors (uniform)* initialization to $\mathcal{U}(\pm 1)$.

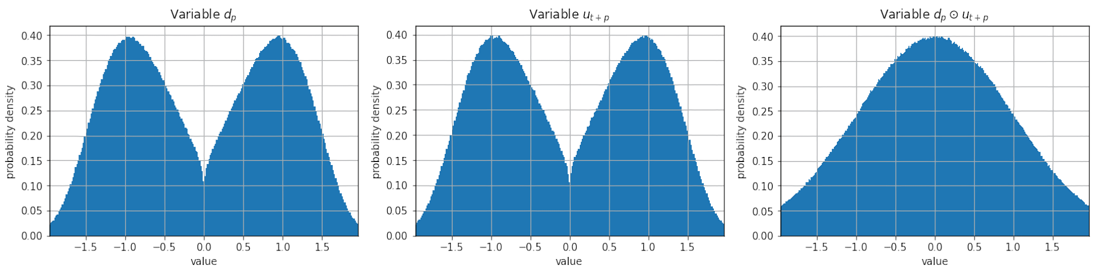

Fig. 4. Probability density functions of values in word vectors $\mathbf{d}_p$ (left), positional vectors $\mathbf{u}_p$ (middle), and their Hadamard products $\mathbf{d}_p \odot \mathbf{u}_p$ (right) with the *same as vanilla word vectors (square-normal)* initialization to a finite-sum approximation of $\mathrm{N}^{0.5}(0, 1)$.

Table 3. English word analogy task accuracies and training times of word vectors without positional weighting and with different initializations for positional weighting. The *identity positional vectors* initialization is unstable for dimensionality $D > 600$.

	Accuracy	Training time
No positional weighting	65.52%	2h 06m 33s
No positional weighting (three epochs)	70.94%	4h 41m 17s
Positional weighting, same as vanilla word vectors	50.96%	5h 01m 16s
Positional weighting, identity positional vectors*	75.02%	4h 59m 27s
Positional weighting, same as word vectors (uniform)	74.31%	4h 57m 25s
Positional weighting, same as word vectors (sq.-normal)	74.95%	5h 01m 11s

Table 3 shows that up to 24% of word analogy accuracy can depend on the initialization. The simplest *same as vanilla word vectors* initialization decreases the effective learning rate of positional weighing, leading to a 15% decrease in word analogy accuracy compared to no positional weighting. The second most obvious *identity positional vectors* initialization leads to a 9% increase in word analogy accuracy, but it is numerically unstable for word vector dimensionality $D > 600$. The least obvious *same as word vectors* initializations also achieve a 9% increase in word analogy accuracy, but they are stable for any word vector dimensionality D. Although positional weighting is three times slower, training with no positional weighting for three epochs only leads to a 5% increase in word analogy accuracy, which shows the practical usefulness of positional weighting.

To make results reproducible, we suggest that all papers should report the initialization of weights in their neural networks.

5 Conclusion

With the rapid pace of research in machine learning, the publish-or-perish mantra of academic publishing, and the ever-increasing complexity of language models, maintaining a controlled experimental environment is more difficult than ever. However, identifying and disclosing all confounding variables is important, since it allows us to reproduce and meaningfully compare results.

Our study shows that even simple log-bilinear SVMs contain parameters that are frequently neglected in experiments, although their impact on the results is significant. We believe that more complex machine learning models such as Transformers contain dozens of baked-in parameters and implicit weight initializations that might well be the tipping point towards the singularity.

We hope that our study will make it easier to reproduce both previous and future word vector experiments, and will serve as an inspiration for upholding the principles of reproducibility in future research of machine learning.

Acknowledgments. First author's work was funded by the South Moravian Centre for International Mobility as a part of the Brno Ph.D. Talent project.

References

1. Berardi, G., Esuli, A., Marcheggiani, D.: Word Embeddings Go to Italy: A Comparison of Models and Training Datasets. In: Boldi, P., Perego, R., Sebastiani, F. (eds.) Italian Information Retrieval Workshop (IIR 2015). Cagliari, Italy (2015), http://ceur-ws.org/Vol-1404/paper_11.pdf
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
3. Bouma, G.: Normalized (pointwise) mutual information in collocation extraction. Proceedings of GSCL pp. 31–40 (2009)
4. Cardellino, C.: Spanish billion word corpus and embeddings (2016), https://crscardellino.github.io/SBWCE/
5. Charlet, D., Damnati, G.: Simbow at SemEval-2017 task 3: Soft-cosine semantic similarity between questions for community question answering. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). pp. 315–319 (2017)
6. Chen, X., Liu, Z., Sun, M.: A unified model for word sense representation and disambiguation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1025–1035 (2014)
7. Chen, X., Xu, L., Liu, Z., Sun, M., Luan, H.: Joint learning of character and word embeddings. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
8. Clark, K., Khandelwal, U., Levy, O., Manning, C.D.: What does BERT look at? An analysis of BERT's attention. arXiv preprint arXiv:1906.04341 (2019)
9. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American society for information science **41**(6), 391–407 (1990)
10. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. arXiv preprint arXiv:1802.06893 (2018)
11. Güngör, O., Yıldız, E.: Linguistic features in turkish word representations. In: 2017 25th Signal Processing and Communications Applications Conference (SIU). pp. 1–4. IEEE (2017)
12. Hartmann, N., Fonseca, E., Shulby, C., Treviso, M., Rodrigues, J., Aluisio, S.: Portuguese word embeddings: Evaluating on word analogies and natural language tasks. arXiv preprint arXiv:1708.06025 (2017)
13. Heaps, H.S.: Information retrieval, computational and theoretical aspects. Academic Press (1978)
14. Kusner, M., Sun, Y., Kolkin, N., Weinberger, K.: From word embeddings to document distances. In: International conference on machine learning. pp. 957–966 (2015)
15. Köper, M., Scheible, C., im Walde, S.S.: Multilingual reliability and "semantic" structure of continuous word spaces. In: Proceedings of the 11th international conference on computational semantics. pp. 40–45 (2015)
16. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Advances in neural information processing systems. pp. 2177–2185 (2014)
17. Levy, O., Goldberg, Y., Dagan, I.: Improving distributional similarity with lessons learned from word embeddings. Transactions of the Association for Computational Linguistics **3**, 211–225 (2015)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

19. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A.: Advances in pre-training distributed word representations. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018) (2018)
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
21. Mnih, A., Kavukcuoglu, K.: Learning word embeddings efficiently with noise-contrastive estimation. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems. vol. 26, pp. 2265–2273. Curran Associates, Inc. (2013), https://proceedings.neurips.cc/paper/2013/file/db2b4182156b2f1f817860ac9f409ad7-Paper.pdf
22. Novotný, V., Sojka, P., Štefánik, M., Lupták, D.: Three is better than one. In: CEUR Workshop Proceedings. Thessaloniki, Greece (2020), http://ceur-ws.org/Vol-2696/paper_235.pdf
23. Pinelis, I.: The exp-normal distribution is infinitely divisible. arXiv preprint arXiv:1803.09838 (2018)
24. Ravshan, S.K.: Factor analysis and uniform distributions (2018), https://ravshansk.com/articles/uniform-distribution.html
25. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), https://is.muni.cz/publication/884893/en
26. Rogers, A., Drozd, A., Li, B.: The (too many) problems of analogical reasoning with word vectors. In: Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017). pp. 135–148 (2017)
27. Sen, M.U., Erdogan, H.: Learning word representations for turkish. In: 2014 22nd Signal Processing and Communications Applications Conference (SIU). pp. 1742–1745. IEEE (2014)
28. Svoboda, L., Brychcin, T.: New word analogy corpus for exploring embeddings of czech words. In: International Conference on Intelligent Text Processing and Computational Linguistics. pp. 103–114. Springer (2016)
29. Unicode Consortium: The Unicode® standard. Mountain view, CA (2020)
30. Venekoski, V., Vankka, J.: Finnish resources for evaluating language model semantics. In: Proceedings of the 21st Nordic Conference on Computational Linguistics. pp. 231–236 (2017)
31. Řehůřek, R.: Subspace tracking for latent semantic analysis. In: European Conference on Information Retrieval. pp. 289–300. Springer (2011)