


Making sense of word2vec

 RADIM ŘEHŮŘEK ([HTTPS://RARE-TECHNOLOGIES.COM/AUTHOR/RADIM/](https://rare-technologies.com/author/radim/)). /

 2014-12-23 /

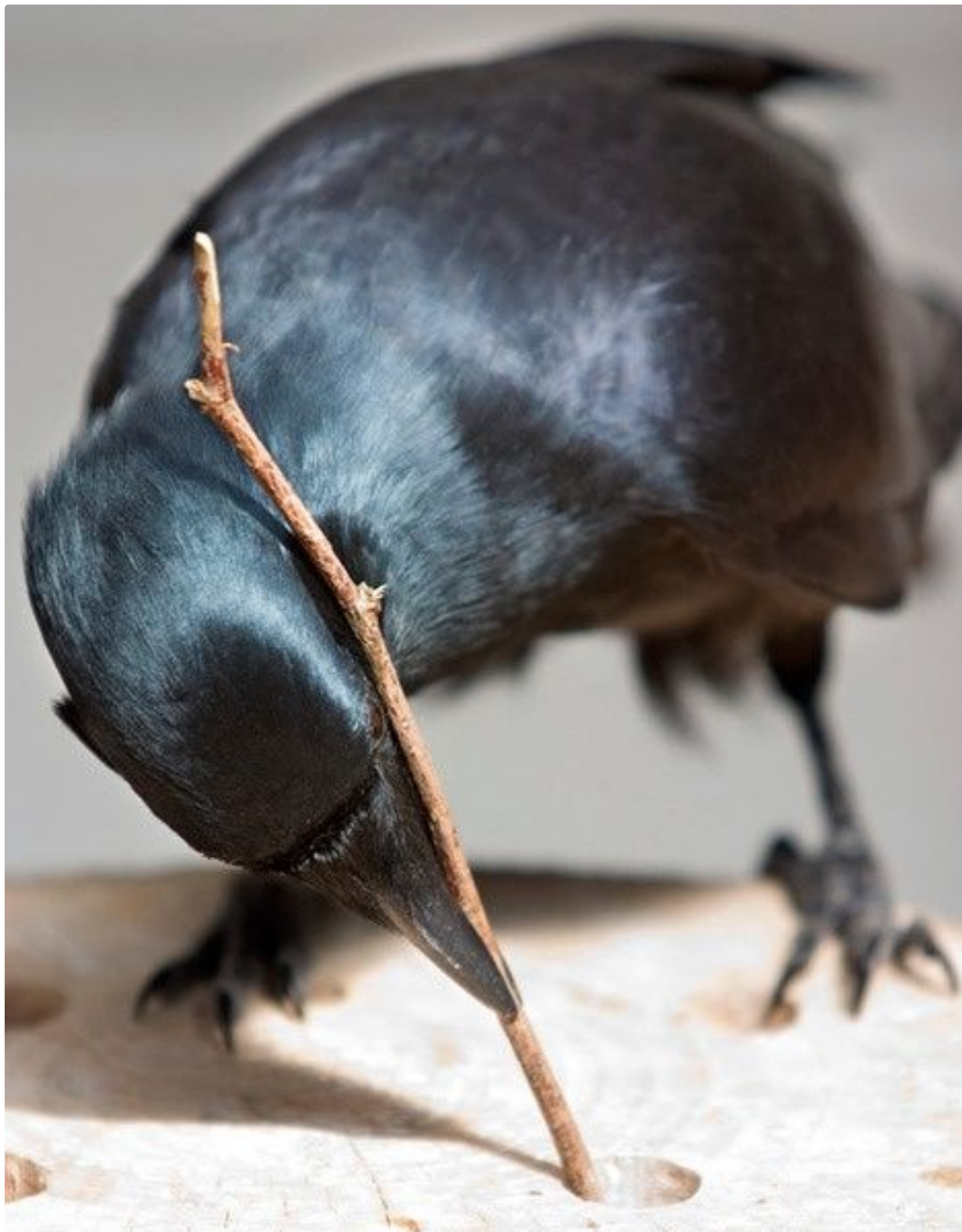
 [GENSIM \(HTTPS://RARE-TECHNOLOGIES.COM/CATEGORY/GENSIM/\)](https://rare-technologies.com/category/gensim/),

 [PROGRAMMING \(HTTPS://RARE-TECHNOLOGIES.COM/CATEGORY/PROGRAMMING/\)](https://rare-technologies.com/category/programming/). /

 50 COMMENTS (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC)

One year ago, Tomáš Mikolov (together with his colleagues at Google) made some ripples by [releasing word2vec \(https://code.google.com/p/word2vec/\)](https://code.google.com/p/word2vec/), an unsupervised algorithm for learning the meaning behind words. In this blog post, I'll evaluate some extensions that have appeared over the past year, including GloVe and matrix factorization via SVD.

In case you missed the buzz, word2vec was widely featured as a member of the “new wave” of machine learning algorithms based on neural networks, commonly referred to as *deep learning* (though word2vec itself is rather shallow). Using large amounts of unannotated plain text, **word2vec learns relationships between words automatically**. The output are vectors, one vector per word, with remarkable linear relationships that allow us to do things like $\text{vec}(\text{“king”}) - \text{vec}(\text{“man”}) + \text{vec}(\text{“woman”}) \approx \text{vec}(\text{“queen”})$, or $\text{vec}(\text{“Montreal Canadiens”}) - \text{vec}(\text{“Montreal”}) + \text{vec}(\text{“Toronto”})$ resembles the vector for “Toronto Maple Leafs”.



Apparently crows are good at that stuff, too: [Crows Can Understand Analogies](http://www.iflscience.com/plants-and-animals/crows-understand-analogies) (<http://www.iflscience.com/plants-and-animals/crows-understand-analogies>).

Check out my [online word2vec demo](http://radimrehurek.com/2014/02/word2vec-tutorial/#app) (<http://radimrehurek.com/2014/02/word2vec-tutorial/#app>) and the [blog series on optimizing word2vec in Python](http://radimrehurek.com/2013/09/deep-learning-with-word2vec-and-gensim/) (<http://radimrehurek.com/2013/09/deep-learning-with-word2vec-and-gensim/>) for more background.

So, what's changed?

For one, Tomáš Mikolov no longer works for Google :-)

More relevantly, there was a lovely piece of research done by the good people at Stanford: Jeffrey Pennington, Richard Socher and Christopher Manning. They explicitly identified the objective that word2vec optimizes through its async stochastic gradient backpropagation algorithm, and neatly connected it to the well-established field of matrix factorizations.

And in case you've never heard of that — in short, word2vec ultimately learns word vectors and word context vectors. These can be viewed as two 2D matrices (of floats), of size $\#words \times \#dim$ each. Their method **GloVe (Global Vectors) identified a matrix which, when factorized using the particular SGD algorithm of word2vec, yields out exactly these two matrices.** So where word2vec was a bit hazy about what's going on underneath, GloVe explicitly names the “objective” matrix, identifies the factorization, and provides some intuitive justification as to why this should give us working similarities.

Very nice and clear paper, [go read it](http://web.stanford.edu/~jpennin/papers/glove.pdf)

(<http://web.stanford.edu/~jpennin/papers/glove.pdf>) if you haven't!

For example, if we have the following [nine preprocessed sentences](http://radimrehurek.com/gensim/tut1.html#from-strings-to-vectors)

(<http://radimrehurek.com/gensim/tut1.html#from-strings-to-vectors>), and set `window=5`, the co-occurrence matrix looks like this:

```
1  # nine input sentences
2  texts = [['human', 'interface', 'computer'],
3           ['survey', 'user', 'computer', 'system', 'response', 'time'],
4           ['eps', 'user', 'interface', 'system'],
5           ['system', 'human', 'system', 'eps'],
6           ['user', 'response', 'time'],
7           ['trees'],
8           ['graph', 'trees'],
9           ['graph', 'minors', 'trees'],
10          ['graph', 'minors', 'survey']]
11
12  # word-word co-occurrence matrix, with context window size of 5
13  [[0 1 1 1 1 1 1 0 0 0 0]
14   [1 0 1 0 0 2 0 0 1 0 0]
15   [1 1 0 0 0 1 0 1 1 0 0]
16   [1 0 0 0 1 1 2 2 0 0 0]
17   [1 0 0 1 0 1 1 1 0 0 1]
18   [1 2 1 1 1 2 1 2 3 0 0]
19   [1 0 0 2 1 1 0 2 0 0 0]
20   [1 0 1 2 1 2 2 0 1 0 0]
21   [0 1 1 0 0 3 0 1 0 0 0]
22   [0 0 0 0 0 0 0 0 0 2 1]
23   [0 0 0 0 1 0 0 0 0 2 0]
24   [0 0 0 0 1 0 0 0 0 1 2]]
25  # (rows/columns represent words:
26  # "computer human interface response survey system time user eps 1
27  # in that order)
```

Note how the matrix is very sparse and symmetrical; the implementation we'll use below takes advantage of both these properties to train GloVe more efficiently.

The GloVe algorithm then transforms such raw integer counts into a matrix where the co-occurrences are weighted based on their distance within the window (word pairs farther apart get less co-occurrence weight):

```
1 # same row/column names as above
2 [[ 0.  0.5  1.  0.5  0.5  1.  0.33  1.  0.  0.  0.  0.  0
3   [ 0.  0.  1.  0.  0.  2.  0.  0.  0.5  0.  0.  0.  0
4   [ 0.  0.  0.  0.  0.  1.  0.  1.  0.5  0.  0.  0.  0
5   [ 0.  0.  0.  0.  0.25  1.  2.  1.33  0.  0.  0.  0.  0
6   [ 0.  0.  0.  0.  0.  0.33  0.2  1.  0.  0.  0.5  1.  0
7   [ 0.  0.  0.  0.  0.  0.  0.5  1.  1.67  0.  0.  0.  0
8   [ 0.  0.  0.  0.  0.  0.  0.  0.75  0.  0.  0.  0.  0
9   [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0
10  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
11  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.5  1.  0
12  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  2.  0
13  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0
```

then takes a log, and factorizes this matrix to produce the final word vectors.

This was really exciting news – **it means the plain softmax word2vec essentially reduces to counting how many times words occur together**, with some scaling thrown in. Technically, this is just a glorified `cooccurrence_counts[word, other_word]++` in a loop, followed by any of the standard matrix factorization algorithms, both of which are well understood processes with efficient implementations.

GloVe vs word2vec

Oddly, the evaluation section of this GloVe paper didn't match the quality of the rest. It had serious flaws in how the experiments compared GloVe to other methods. Several people called the authors out on the weirdness, most lucidly the Levy & Goldberg research duo from Bar Ilan university – check out their [“apples to apples” blog post](#)

(<https://docs.google.com/a/radimrehurek.com/document/d/1ydlujJ7ETSZ688RGfU5IMJJskRl8czSwpti15s/edit#>) for a bit of academic drama. To summarize, when

evaluated properly, paying attention to parameter settings, GloVe doesn't really seem to outperform the original word2vec, let alone by 11% as the GloVe paper claimed.

Luckily, Maciej Kula implemented GloVe in Python, using Cython for performance. Using his [neat implementation](https://github.com/maciejkula/glove-python/) (<https://github.com/maciejkula/glove-python/>), we can try to make sense of the performance and accuracy ourselves.

Code to train GloVe in Python:

```
1 from gensim import utils, corpora, matutils, models
2 import glove
3
4 # Restrict dictionary to the 30k most common words.
5 wiki = models.word2vec.LineSentence('/data/shootout/title_tokens.txt.gz')
6 id2word = corpora.Dictionary(wiki)
7 id2word.filter_extremes(keep_n=30000)
8 word2id = dict((word, id) for id, word in id2word.iteritems())
9
10 # Filter all wiki documents to contain only those 30k words.
11 filter_text = lambda text: [word for word in text if word in word2id]
12 filtered_wiki = lambda: (filter_text(text) for text in wiki) # generator
13
14 # Get the word co-occurrence matrix -- needs lots of RAM!!
15 cooccur = glove.Corpora()
16 cooccur.fit(filtered_wiki(), window=10)
17
18 # and train GloVe model itself, using 10 epochs
19 model_glove = glove.Glove(no_components=600, learning_rate=0.05)
20 model_glove.fit(cooccur.matrix, epochs=10)
```

And similarly for training word2vec:

```
1 model_word2vec = models.Word2Vec(size=600, window=10)
2 model_word2vec.build_vocab(filtered_wiki())
3 model_word2vec.train(filtered_wiki())
```

The reason why we restricted the vocabulary to only 30,000 words is that Maciej's implementation of **GloVe requires memory quadratic in the number of words**: it keeps that sparse matrix of all word x word co-occurrences in RAM. In contrast, the gensim word2vec implementation is happy with linear memory, so millions of words are not a problem there. This is not an intrinsic limitation of GloVe though; with a different implementation, the co-occurrence matrix could be assembled out-of-core (Map/Reduce seems ideal for the job), and the factorization could just stream over it with constant memory too, in a more gensim-like fashion.

Results for 600 dims, context window of 10, 1.9B words of EN Wikipedia.

Algorithm	Accuracy on the word analogy task	Wallclock time
I/O only = iterating over wiki with $\text{sum}(\text{len}(\text{text}) \text{ for } \text{text} \text{ in } \text{filtered_wiki}())$	N/A	3m
GloVe, 10 epochs, learning rate 0.05	67.1%	4h12m
GloVe, 100 epochs, learning rate 0.05	67.3%	18h39m
word2vec, hierarchical skipgram, 1 epoch	57.4%	3h10m
word2vec, negative sampling with 10 samples, 1 epoch	68.3%	8h38m
word2vec, <u>pre-trained GoogleNews model released by Tomáš Mikolov</u> (https://code.google.com/p/word2vec/#Pre-trained_word_and_phrase_vectors), 300 dims, 3,000,000 vocabulary	55.3%	?

Basically, where GloVe precomputes the large word x word co-occurrence matrix in memory and then quickly factorizes it, word2vec sweeps through the sentences in an online fashion, handling each co-occurrence separately. So, there is a **tradeoff between taking more memory (GloVe) vs. taking longer to train (word2vec)**. Also, once computed, GloVe can re-use the co-occurrence matrix to quickly factorize with any dimensionality, whereas word2vec has to be trained from scratch after changing its embedding dimensionality.

Note that both implementations are fairly optimized, running on 8 threads (on an 8 core machine), using the exact same input corpus, text preprocessing, vocabulary and evaluation code, so that the numbers are directly comparable. [Code here](https://github.com/piskvorky/word_embeddings) (https://github.com/piskvorky/word_embeddings).

SPPMI and SVD

In a manner analogous to GloVe, Levy and Goldberg (the same researchers mentioned above) [analyzed the objective function of word2vec with negative sampling](https://levyomer.wordpress.com/2014/09/10/neural-word-embeddings-as-implicit-matrix-factorization/) (<https://levyomer.wordpress.com/2014/09/10/neural-word-embeddings-as-implicit-matrix-factorization/>). That's the one that performed best in the table above, so I decided to check it out too.

Again, they manage to derive a beautifully simple connection to matrix factorization. This time, the word x context objective “source” matrix is computed differently to GloVe. Each matrix cell, corresponding to word w and context word c , is computed as $\max(0.0, PMI(w, c) - \log(k))$, where k is the number of negative samples in word2vec (for example, $k = 10$). PMI is the standard [pointwise mutual information](https://en.wikipedia.org/wiki/Pointwise_mutual_information) (https://en.wikipedia.org/wiki/Pointwise_mutual_information) – if we use the notation that word w and context c occurred together wc times in the training corpus, then

$$PMI(w, c) = \log \frac{wc}{\sum_w wc \sum_c wc} \quad (\text{no smoothing}).$$

The funky “SPPMI” name simply reflects that we’re subtracting $\log(k)$ from PMI (“shifting”) and that we’re taking the $\max(0.0, SPMI)$ (“positive”; should be non-negative, really). So, Shifted Positive Pointwise Mutual Information.

For example, for the same nine texts we used above and $k = 1$, the SPPMI matrix looks like this:

1	[0.	0.83	0.83	0.49	0.49	0.	0.49	0.13	0.	0.	0.	0
2	[0.83	0.	1.16	0.	0.	0.83	0.	0.	0.98	0.	0.	0
3	[0.83	1.16	0.	0.	0.	0.13	0.	0.47	0.98	0.	0.	0
4	[0.49	0.	0.	0.	0.49	0.	1.18	0.83	0.	0.	0.	0
5	[0.49	0.	0.	0.49	0.	0.	0.49	0.13	0.	0.	0.83	1
6	[0.	0.83	0.13	0.	0.	0.	0.	0.13	1.05	0.	0.	0
7	[0.49	0.	0.	1.18	0.49	0.	0.	0.83	0.	0.	0.	0
8	[0.13	0.	0.47	0.83	0.13	0.13	0.83	0.	0.29	0.	0.	0
9	[0.	0.98	0.98	0.	0.	1.05	0.	0.29	0.	0.	0.	0
10	[0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	2.37	1
11	[0.	0.	0.	0.	0.83	0.	0.	0.	0.	2.37	0.	2
12	[0.	0.	0.	0.	1.05	0.	0.	0.	0.	1.9	2.08	0

No neural network training, no parameter tuning, we can directly take rows of this SPPMI matrix to be the word vectors. Very fast and simple. How does raw SPPMI compare to word2vec's and GloVe's factorizations though?

Comparison on 600 dims, context window 10, 1.9B words of EN Wikipedia.

Algorithm	Accuracy on the analogy task	Wallclock time	Peak RAM [MB]
word2vec, negative sampling k=10, 1 epoch	68.3%	8h38m	628
GloVe, learning rate 0.05, 10 epochs	67.1%	4h12m	9,414
SPPMI, k=1	48.7%	50m	3,433
SPPMI, k=10	30.3%	50m	3,429
SPPMI-SVD, k=1	39.4%	1h23m	3,426
SPPMI-SVD, k=10	3.8%	1h23m	3,444

The SPPMI-SVD method simply factorizes the sparse SPPMI matrix using Singular Value Decomposition (SVD) (<http://radimrehurek.com/gensim/models/lsmodel.html>), rather than the gradient descent methods of word2vec/GloVe, and uses the (dense) left singular vectors as the final word embeddings. SVD is a fast, scalable method with straightforward geometric interpretation, and it performed very well in the NIPS experiments of Levy & Goldberg (<https://levyomer.wordpress.com/2014/09/10/neural-word-embeddings-as-implicit-matrix-factorization/>), who suggested SSPMI-SVD.

In the table above, the **quality of both SPPMI and SPPMI-SVD models is atrocious**, especially for higher values of k (more “shift”). I’m not sure why this is; I’ll try to get the original implementation of Levy & Goldberg to compare.

Also, I originally tried to get this table on 1,000 dims, rather than 600. But the GloVe implementation started failing, producing word vectors with NaNs in them, and weird <1% accuracies when I tried to combat that by decreasing its learning rate. Maciej is still working on that one, so if you’re thinking of using GloVe in production, beware. **EDIT: Successfully resolved** (<https://github.com/maciejkula/glove-python/pull/9#issuecomment-68058795>), Maciej’s GloVe handles that fine now.

To make experiments easier, I wrote and published a script that takes the parsed English Wikipedia (<http://radimrehurek.com/2013/12/performance-shootout-of-nearest-neighbours-contestants/>) and computes accuracy on the analogy task, using each of these different algorithms in turn. **You can get it from GitHub** (https://github.com/piskvorky/word_embeddings) and **experiment with the various methods yourself, trying them on your own data / application.**

What does that all mean?

Playing with Wikipedia is fun, but usually clients require more concrete insights from us.

How do we tweak word2vec to better model what we want?

How to tune word2vec model quality on a specific task (which is, in all likelihood, *not* “word analogies”)?

I’ll postpone that until the next post. Suffice to say that the performance depends on tuning the methods’ internal parameters, in non-obvious ways. The Bar Ilan powerhouse of Levy, Goldberg & Dagan wrote a full paper on that, exploring the various parameter combinations (dynamic vs. fixed context window, subsampling, negative distribution smoothing, taking context vectors into account as well...). Their paper is under review now – I’ll post a link here as soon as it becomes public. **EDIT: ACL paper [here](http://www.aclweb.org/anthology/Q15-1016) (<http://www.aclweb.org/anthology/Q15-1016>).**

In the meanwhile, there has been some tentative research into the area of word2vec error analysis and tuning. Check out [this web demo](http://irsv2.cs.biu.ac.il:9998/?word=machine) (<http://irsv2.cs.biu.ac.il:9998/?word=machine>) (by Levy & Goldberg again, from [this paper](http://www.cs.bgu.ac.il/~yoavg/publications/acl2014syntemb.pdf) (<http://www.cs.bgu.ac.il/~yoavg/publications/acl2014syntemb.pdf>)), for investigating which contexts get activated for different words. This can lead to visual insights into which co-occurrences are responsible for a particular class of errors.

TL;DR: the word2vec implementation is still fine and state-of-the-art, you can continue using it :-)

Note from Radim: Get my latest machine learning tips & articles delivered straight to your inbox (it's free).

Unsubscribe anytime, no spamming. Max 2 posts per month, if lucky.

[Subscribe Now](#)

If you like such machine learning shenanigans, sign up for my newsletter above, get in touch for [commercial projects](https://rare-technologies.com/contact) (<https://rare-technologies.com/contact>), or check out my older posts: [efficient nearest neighbour similarity search](http://radimrehurek.com/2013/11/performance-shootout-of-nearest-neighbours-intro/) (<http://radimrehurek.com/2013/11/performance-shootout-of-nearest-neighbours-intro/>), [optimizing word2vec](http://radimrehurek.com/2013/09/deep-learning-with-word2vec-and-gensim/) (<http://radimrehurek.com/2013/09/deep-learning-with-word2vec-and-gensim/>), its [doc2vec extension](http://radimrehurek.com/2014/12/doc2vec-tutorial/) (<http://radimrehurek.com/2014/12/doc2vec-tutorial/>).

[DEEP LEARNING \(HTTPS://RARE-TECHNOLOGIES.COM/TAG/DEEP-LEARNING/\)](https://rare-technologies.com/tag/deep-learning/)

[GENSIM \(HTTPS://RARE-TECHNOLOGIES.COM/TAG/GENSIM/\)](https://rare-technologies.com/tag/gensim/)

[WORD2VEC \(HTTPS://RARE-TECHNOLOGIES.COM/TAG/WORD2VEC/\)](https://rare-technologies.com/tag/word2vec/)

50 COMMENTS



MR. POWERHOUSE

[2014-12-24 AT 6:56 AM \(HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2507\)](https://rare-technologies.com/making-sense-of-word2vec/#comment-2507)

Nice work, Radim! Few comments:

- in your first table, you report 55.3% accuracy for the pretrained vectors from word2vec page – I'm getting >70%
- the training time & accuracy you obtained with your Gensim implementation of word2vec seems to be worse than the C version (when I used script demo-train-big-model-v1.sh from word2vec, the vectors themselves are trained in about three hours using 8 billion words, and the accuracy is 10% higher than yours – big difference!) – maybe you use wrong hyper-parameters? or the C version is faster and more accurate?
- finally, Omer and Yoav report much better results with SPPMI – any idea why your results are worse?

Thanks!

[Reply ↩ \(HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/?replytocom=2507#respond\)](https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2507#respond)



[RADIM \(HTTP://RADIMREHUREK.COM\)](http://radimrehurek.com)

[2014-12-25 AT 12:37 PM \(HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2508\)](https://rare-technologies.com/making-sense-of-word2vec/#comment-2508)

post
author

Hi Mr. Powerhouse 😊

1. All experiments were done using the same 30k vocabulary, so that the numbers can be compared directly. The accuracy there is ~55% using the GoogleNews model, not >70%. The model performed very well on the syntactic tasks, and poorly on the semantic ones.

2. No, the two are pretty much identical. Check out the [word2vec porting series](http://radimrehurek.com/2013/09/deep-learning-with-word2vec-and-gensim/optimizing-word2vec) (<http://radimrehurek.com/2013/09/deep-learning-with-word2vec-and-gensim/optimizing-word2vec>). Your results are most likely due to different training corpus / hardware / level of parallelization. “Word2vec” are actually many algorithms, depending on the parameter settings. You’d have to control for all of those, to get meaningful apples-to-apples comparison.

3. I’m in touch with them, looking for an answer. Will update when we’ve found out the reason. It’s probably not the SPPMI code itself, which is very straightforward (see the [github repo](https://github.com/piskvorky/word_embeddings/blob/master/run_embed.py#L177) (https://github.com/piskvorky/word_embeddings/blob/master/run_embed.py#L177)).

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2508#respond>)



SEBASTIEN-J

2014-12-29 AT 9:55 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2510](https://rare-technologies.com/making-sense-of-word2vec/#comment-2510))

The accuracy function in gensim is unfair to the GoogleNews model, which was trained on non-lowercased data. For a better evaluation, the first letter of countries, cities, etc. should not be lowercased. Arguably, words that differ from “a”, “b” and “c” by letter case only should also be rejected, while answers that are identical to the expected word, except for case, should be accepted.

Using the full vocabulary, the accuracy with these modifications is 73.8% (14,222/19,544), while the current gensim implementation gives 55.3% (7581/13705), with many questions rejected.

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2510#respond>)



SEBASTIEN-J

2014-12-29 AT 11:05 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2511](https://rare-technologies.com/making-sense-of-word2vec/#comment-2511))

typo: 14,222 -> 14,422

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2511#respond>)



RADIM ([HTTP://RADIMREHUREK.COM](http://radimrehurek.com))

2015-01-01 AT 10:54 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2513](https://rare-technologies.com/making-sense-of-word2vec/#comment-2513))

post
author

No, the evaluation procedure is the same for both the C and gensim version. Can you post your code? I’m not sure what you’re talking about.

Both evaluations treat words as case insensitive, meaning if the model suggests “Montreal”, and the expected answer is “montreal”, it’s recorded as a success = correct answer.

The only difference is the Python version can handle unicode, whereas the C version is ASCII-only. (So if a word contained “Č” or “Á”, the C version wouldn’t be able to handle the lower case correctly. No words in the eval set do, though, so it makes no difference here.)

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2513#respond>)



SEBASTIEN-J

2015-01-02 AT 9:32 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2514](https://rare-technologies.com/making-sense-of-word2vec/#comment-2514))

<https://gist.github.com/anonymous/06e060cdf53257fde1ef>
(<https://gist.github.com/anonymous/06e060cdf53257fde1ef>)

For the GoogleNews model, `lowercase=False` should be used in the function above.

If a word is represented by many vectors, which one we choose when asking the analogy questions matters. In general, using the vector corresponding to the most common surface form will lead to higher accuracy. The vector for “Montreal” (common) would be `better` than the one for “montreal” (rare).

For example, the GoogleNews model gives the wrong answer to “athens:greece::beijing:?”. However, if we ask “Athens:Greece::Beijing:?”, then the prediction is correct.



RADIM ([HTTP://RADIMREHUREK.COM](http://radimrehurek.com))

2015-01-02 AT 5:38 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2515](https://rare-technologies.com/making-sense-of-word2vec/#comment-2515))

post
author

I think I see what you mean. You’re proposing a different evaluation technique, one where letter case matters.

That makes sense, and you’re right in that such evaluation may give different results for models trained with case sensitivity. However, neither the C tool nor gensim implement such evaluation. So if they’re “unfair” in their evaluation, they are both equally unfair.

Looking at the C source, it will ASCII-**uppercase** all eval words and all model vocabulary; gensim will unicode-**lowercase** all eval words and leave vocabulary intact.



RADIM ([HTTP://RADIMREHUREK.COM](http://radimrehurek.com))

2015-01-03 AT 6:27 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2516](https://rare-technologies.com/making-sense-of-word2vec/#comment-2516))

post
author

You left me wondering how the C tools handles conflicts when uppercasing its model vocabulary: “Montreal” and “montreal” will both map to “MONTREAL”, so which word’s embedding is actually chosen during the accuracy evaluation?

It seems that in this case, the C word match during eval is set up (maybe accidentally, maybe by design, there are no comments) so that the most frequent surface form is used. Which is probably what we want. Or maybe being explicit, raising an error, is preferable when there’s a casing conflict? Or your approach, where there’s a bool flag that says “either ignore case, or use my case exactly”? I’m not sure.

I think I’ll change gensim to use one of these options & make the docs clear about what’s going on. This only affects models with case-sensitive vocabulary, of course.

Thanks for bringing this up Sebastien.



STEFAN

2016-04-09 AT 2:30 PM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2543)

The link is dead unfortunately:

<http://web.stanford.edu/~jpennin/papers/glove.pdf>
(<http://web.stanford.edu/~jpennin/papers/glove.pdf>).

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2543#respond)



JACK

2016-08-30 AT 6:03 PM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2558)

Why do we have to remove words that appear only once when building word2vec?

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2558#respond)



LEONID BOYTSOV (HTTP://SEARCHIVARIUS.ORG/ABOUT)

2014-12-25 AT 8:28 PM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2509)

A very interesting post, thank you.

I haven't seen the original papers. However, off the top of my head, it seems very strange to me that you can use SPPMI matrix directly without any dimensionality reduction. The number of columns (each one corresponds to a context word) should be huge. Did I miss something?

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2509#respond)



post
author

RADIM (HTTP://RADIMREHUREK.COM)

2015-01-01 AT 10:49 AM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2512)

Hi Leonid — you missed nothing. The SPPMI matrix is huge.

It's basically #words x #words, like I described in the post. In theory, it's also very sparse. In this particular case though, after using only the most frequent 30k words, the sparsity is pretty low = the SPPMI matrix is almost dense.

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2512#respond)



MARCEL

2015-02-16 AT 7:50 AM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2519)

Hi Radim,
thank you for your great post!
Do you have an update on this for us?
Eg. you write
“How do we tweak word2vec to better model what we want? How to tune word2vec model quality on a specific task (which is, in all likelihood, not “word analogies”)? I’ll postpone that until the next post.”
and
“Levy & Goldberg wrote a full paper on that, exploring the various parameter combinations (dynamic vs. fixed context window, subsampling, negative distribution smoothing, taking context vectors into account as well....). Their paper is under review now – I’ll post a link here as soon as it becomes public.”
Thank you for your great work!
Marcel

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2519#respond)



post
author

RADIM (HTTP://RADIMREHUREK.COM)

2015-02-18 AT 8:54 AM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2522)

Thanks Marcel!
Too much “real” work, I know I’ve been neglecting the blog a bit... I’ll try to post again soon.
Levy & Goldberg: I don’t know, I’ll check for you.

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2522#respond)



post
author

RADIM (HTTP://RADIMREHUREK.COM)

2015-02-18 AT 9:30 AM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2523)

Checked: no, not out yet.
But try contacting Omer Levy directly, they say they’ll be happy to share privately!

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2523#respond)



2015-02-17 AT 5:58 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2520](https://rare-technologies.com/making-sense-of-word2vec/#comment-2520))

Have you considered the approach of Hellinger PCA, which applies SVD to the co-occurrence matrix and is also cited in the GloVe paper?

Lebret, Rémi, and Ronan Collobert. "Word Embeddings through Hellinger PCA." EACL 2014 (2014): 482.

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2520#respond>)



post
author

[RADIM \(HTTP://RADIMREHUREK.COM\)](http://radimrehurek.com)

2015-02-18 AT 8:52 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2521](https://rare-technologies.com/making-sense-of-word2vec/#comment-2521))

No we haven't, I haven't seen that paper yet. Do you want to add the method & report back?

Our evaluation here is open source, and gensim supports large scale SVD, so I'm thinking the implementation may be straightforward.

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2521#respond>)



ANATOLY

2015-04-17 AT 8:59 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2524](https://rare-technologies.com/making-sense-of-word2vec/#comment-2524))

"Levy & Goldberg wrote a full paper on that, exploring the various parameter combinations (dynamic vs. fixed context window, subsampling, negative distribution smoothing, taking context vectors into account as well...)"

Is it following paper?

<https://levyomer.files.wordpress.com/2015/03/improving-distributional-similarity-tacl-2015.pdf> (<https://levyomer.files.wordpress.com/2015/03/improving-distributional-similarity-tacl-2015.pdf>)

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2524#respond>)



post
author

[RADIM \(HTTP://RADIMREHUREK.COM\)](http://radimrehurek.com)

2015-04-18 AT 2:34 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2525](https://rare-technologies.com/making-sense-of-word2vec/#comment-2525))

Yes. I tweet it out some time ago, but forgot to update this article 😊

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2525#respond>)

Pingback: [GloVe: Global Vectors for Word Representations « Building Babylon](http://building-babylon.net/2015/07/29/glove-global-vectors-for-word-representations/) (<http://building-babylon.net/2015/07/29/glove-global-vectors-for-word-representations/>).

Pingback: [CodieNerd \(2\) – Toying with Word2Vec | Everything about Data Analytics](https://datawarrior.wordpress.com/2015/10/25/codienerd-2-toying-with-word2vec/) (<https://datawarrior.wordpress.com/2015/10/25/codienerd-2-toying-with-word2vec/>).

Pingback: [» Compressed Representation: A Really Good Idea Jo's Blog](http://www.josephcatrambone.com/?p=806) (<http://www.josephcatrambone.com/?p=806>).



DMITRIY SELIVANOV ([HTTP://DSNOTES.COM](http://dsnotes.com))

2015-11-09 AT 10:12 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2529](https://rare-technologies.com/making-sense-of-word2vec/#comment-2529))

Hi, Radim. Do you have exactly the same data, on which you perform benchmarks? I implemented GloVe in text2vec R package algorithm (from scratch) and on latest 2.1B wikipedia dump it produce much better results, then you reported here – 75% on word analogy task (top 30k words, dim = 600, 10 sgd epochs, 12182 questions).

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2529#respond>)



post
author

RADIM ([HTTP://RADIMREHUREK.COM](http://radimrehurek.com))

2015-11-09 AT 11:57 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2530](https://rare-technologies.com/making-sense-of-word2vec/#comment-2530))

Hello Dmitriy!

The script used to preprocess the data (xml.bz2 Wikipedia dump) is linked to from this post. It comes from the “nearest neighbour shootout” blog series and is public.

The raw data itself was taken as the latest Wikipedia dump at the time of writing that “shootout” series. Also public.

The GloVe implementation used was from Maciej Kule, also linked to in this post. This implementation is from the time of writing this blog post obviously (it may have changed since).

If you get much better results on the same inputs + parameters, it may be worth checking back with Maciej. I'm sure he'd be interested in hearing comparisons + improving / fixing his implementation.

EDIT: here's some file stats I found on disk:

```
$ ls -l title_tokens.txt
-rw-r--r-- 1 radim develop 11566440066 Apr 15 2015 title_tokens.txt
```

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2530#respond>)



DMITRIY SELIVANOV ([HTTP://DSNOTES.COM](http://dsnotes.com))

2015-11-09 AT 12:14 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2531](https://rare-technologies.com/making-sense-of-word2vec/#comment-2531))

Yes, I used your scripts for preparation, but I can't find wiki dump for 2014-12. The oldest one is for 2015-02: <http://dumps.wikimedia.org/enwiki/> (<https://dumps.wikimedia.org/enwiki/>).

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2531#respond>)



DMITRIY SELIVANOV ([HTTP://DSNOTES.COM](http://dsnotes.com))

2015-11-10 AT 11:33 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2532](https://rare-technologies.com/making-sense-of-word2vec/#comment-2532))

If you are still have interest, the problem is here:

<https://github.com/maciejkula/glove-python/issues/22>
(<https://github.com/maciejkula/glove-python/issues/22>)

He use only upper triangular co-occurrence matrix, and build only single word-vector matrix. This reduce memory and training time by factor of 2, but far less accurate.

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2532#respond>)



RADIM ([HTTP://RADIMREHUREK.COM](http://radimrehurek.com))

2015-11-10 AT 11:44 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2533](https://rare-technologies.com/making-sense-of-word2vec/#comment-2533))

post
author

I don't see how that would affect accuracy. Or in your package, in your evaluation, do you combine the word and context vectors together?

Anyway, I think your analysis and findings are very interesting! Please keep me posted if you write it up in a blog post (and I'll ping Maciej as well). There's such a lack of practical tools in this space (as opposed to media hype).

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2533#respond>)



DMITRIY SELIVANOV ([HTTP://DSNOTES.COM](http://dsnotes.com))

2015-11-10 AT 11:49 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2534](https://rare-technologies.com/making-sense-of-word2vec/#comment-2534))

Yes, I combine two matrices (just add), as it done in original paper and implementation. I'll write a post in next few weeks and ping you back.



DMITRIY SELIVANOV ([HTTP://DSNOTES.COM](http://dsnotes.com))

2015-12-01 AT 11:28 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2536](https://rare-technologies.com/making-sense-of-word2vec/#comment-2536))

Hi! see my post <http://dsnotes.com/blog/text2vec/2015/12/01/glove-enwiki/>
(<http://dsnotes.com/blog/text2vec/2015/12/01/glove-enwiki/>).

If comparison looks not fair, let me know, I'll happy to correct. Thanks for all your work, it

helps me a lot in development of text2vec.



post
author

RADIM ([HTTP://RADIMREHUREK.COM](http://radimrehurek.com))

2015-12-02 AT 6:23 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2537](https://rare-technologies.com/making-sense-of-word2vec/#comment-2537))

Great work Dmitriy!

It's good to see people running realistic comparison on real, replicable datasets. I'll tweet your results.

Pingback: [گوریتیم | مهندسی داده Word2Vec آشنایی با الگوریتم](http://www.bigdata.ir/2015/10/%d8%a2%d8%b4%d9%86%d8%a7%db%8c%db%8c-%d8%a8%d8%a7-%d8%a7%d9%84%da%af%d9%88%d8%b1%db%8c%d8%aa%d9%85-word2vec-%da%af%d9%88%da%af%d9%84/)

(<http://www.bigdata.ir/2015/10/%d8%a2%d8%b4%d9%86%d8%a7%db%8c%db%8c-%d8%a8%d8%a7-%d8%a7%d9%84%da%af%d9%88%d8%b1%db%8c%d8%aa%d9%85-word2vec-%da%af%d9%88%da%af%d9%84/>)

Pingback: [inter-word2vec | Rob Myers](http://robmyers.org/2015/07/21/inter-word2vec/) (<http://robmyers.org/2015/07/21/inter-word2vec/>)

Pingback: [Deep Learning-ML Tutorials | Tony Deep Techs](https://tonydeep.wordpress.com/2015/12/14/deep-learning-ml-tutorials/) (<https://tonydeep.wordpress.com/2015/12/14/deep-learning-ml-tutorials/>)

Pingback: [自然语言处理相关深度学习资源 |](http://hanshitou.com/archives/674/) (<http://hanshitou.com/archives/674/>)



SANDER

2016-03-02 AT 4:55 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2541](https://rare-technologies.com/making-sense-of-word2vec/#comment-2541))

sorry, but what is it "accuracy on the word analogy task"?

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2541#respond>)



post
author

RADIM REHUREK ([HTTP://RADIMREHUREK.COM](http://radimrehurek.com))

2016-03-03 AT 3:08 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2542](https://rare-technologies.com/making-sense-of-word2vec/#comment-2542))

See <https://code.google.com/archive/p/word2vec/> (<https://code.google.com/archive/p/word2vec/>) (it's a set of word 4-tuples, loaded from word2vec's questions-words.txt file).

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2542#respond>)



LILING

2016-04-15 AT 5:49 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2544](https://rare-technologies.com/making-sense-of-word2vec/#comment-2544))

@Radim, just wondering what happens when the #dimensions in the GloVe output is more than the vocab size, would GloVe break? Or would GloVe just throw 0s in the additional dimensions?

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2544#respond>)

Pingback: » "Numberless degrees of similitude": A response to Ryan Heuser's 'Word Vectors in the Eighteenth Century, Part 1' Gabriel Recchia (<http://www.twonewthings.com/gabrielrecchia/2016/06/11/numberless-degrees-of-similitude-word-vectors/>)



JACK

2016-08-30 AT 6:04 PM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2559](https://rare-technologies.com/making-sense-of-word2vec/#comment-2559))

why do we have to remove the words that appear only once?

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2559#respond>)

Pingback: Short Text Categorization using Deep Neural Networks and Word-Embedding Models – Everything about Data Analytics (<https://datawarrior.wordpress.com/2016/10/12/short-text-categorization-using-deep-neural-networks-and-word-embedding-models/>)

Pingback: word2vec - deep learning on NLP - SolutionHacker.com (<http://solutionhacker.com/word2vec-deep-learning-nlp/>)



HOBSON LANE ([HTTP://TOTALGOOD.COM](http://totalgood.com))

2017-04-23 AT 6:10 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2633](https://rare-technologies.com/making-sense-of-word2vec/#comment-2633))

The Pennington GloVe paper has moved. It's now here:
<https://nlp.stanford.edu/pubs/glove.pdf> (<https://nlp.stanford.edu/pubs/glove.pdf>)

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2633#respond>)



RADIM ŘEHŮŘEK ([HTTP://RADIMREHUREK.COM](http://radimrehurek.com))

2017-04-23 AT 6:58 AM ([HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2634](https://rare-technologies.com/making-sense-of-word2vec/#comment-2634))

post
author

Many thanks! Link updated.

Reply ↩ (<https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2634#respond>)

Pingback: [Sentence Based similarity – Research & Experimental Blog](https://rebcs.wordpress.com/2017/06/05/sentence-based-similarity/)
(<https://rebcs.wordpress.com/2017/06/05/sentence-based-similarity/>)



HOWARD

2017-11-06 AT 4:51 AM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2668)

Hi, Radim,

Thanks for this post. Just curious any update on SVD regarding “I’m not sure why this is; I’ll try to get the original implementation of Levy & Goldberg to compare.” Thanks,

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2668#respond)

Pingback: [Episode 3: Word Embeddings – New Paradigm](https://mungingdata.wordpress.com/2018/01/15/episode-3-word-embeddings/)
(<https://mungingdata.wordpress.com/2018/01/15/episode-3-word-embeddings/>)



SUMEET KUMAR

2018-09-16 AT 9:52 AM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2770)

Hi,

I am using word2vec from gensim library for one of the project. I am successfully able to get the word embedding vector of size = 16. Now I want to evaluate the result of that vector whether the received vector is referring to the correct word in the dataset or not. How do i do that?

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2770#respond)



BOB

2018-09-21 AT 9:49 PM (HTTPS://RARE-TECHNOLOGIES.COM/MAKING-SENSE-OF-WORD2VEC/#COMMENT-2771)

Hi Radim, thank you for this very clear explanation, I need to create a words co-occurrence matrix which need a lot of memory as you said, can you please give me some details about the way you’re mentioning here to do the task : “This is not an intrinsic limitation of GloVe though; with a different implementation, the co-occurrence matrix could be assembled out-of-core (Map/Reduce seems ideal for the job)”

Thank you

Reply ↩ (https://rare-technologies.com/making-sense-of-word2vec/?replytocom=2771#respond)

Leave a comment

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Submit

Author of Post



Radim Řehůřek's bio:

Founder at RARE Technologies, creator of Gensim. SW engineer since 2004, PhD in AI in 2011. Lover of geology, history and beginnings in general. Occasional travel blogger.

Need Expert Consulting in ML and NLP?

Your Name

Email

Project Details

Send

Categories

Select Category



Archives

Select Month



Recent Posts

Export PII drill-down reports (<https://rare-technologies.com/personal-data-reports/>)

Personal Data Analytics (https://rare-technologies.com/pii_analytics/)

Scanning Office 365 for sensitive PII information (<https://rare-technologies.com/pii-scan-o365-connector/>)

Pivoted document length normalisation (<https://rare-technologies.com/pivoted-document-length-normalisation/>)

STAY AHEAD OF THE CURVE

Get our latest tutorials, updates and insights delivered straight to your inbox.

Your first name

Your email address

Subscribe

1-2 times a month, if lucky. Your information will not be shared.



[SERVICES \(HTTPS://RARE-TECHNOLOGIES.COM/SERVICES/\)](https://rare-technologies.com/services/)

[OUR TEAM \(HTTPS://RARE-TECHNOLOGIES.COM/OUR-TEAM/\)](https://rare-technologies.com/our-team/)

[BLOG \(HTTPS://RARE-TECHNOLOGIES.COM/BLOG/\)](https://rare-technologies.com/blog/)

[CONTACT \(HTTPS://RARE-TECHNOLOGIES.COM/CONTACT/\)](https://rare-technologies.com/contact/)

[SITE MAP \(/SITEMAP\)](/sitemap)

[CAREERS \(HTTPS://RARE-](https://rare-technologies.com/careers/)

[TECHNOLOGIES.COM/CAREERS/\)](https://rare-technologies.com/careers/)

[CORPORATE TRAINING \(HTTPS://RARE-TECHNOLOGIES.COM/CORPORATE-TRAINING/\)](https://rare-technologies.com/corporate-training/)

[INCUBATOR \(HTTPS://RARE-TECHNOLOGIES.COM/INCUBATOR/\)](https://rare-technologies.com/incubator/)

[COMPETITIONS \(HTTPS://RARE-TECHNOLOGIES.COM/COMPETITIONS/\)](https://rare-technologies.com/competitions/)

RaRe Technologies

✉ info@rare-technologies.com (<mailto:info@rare-technologies.com>)



[.https://www.facebook.com/RaReTechnologies](https://www.facebook.com/RaReTechnologies)



[.https://twitter.com/RaReTechTeam](https://twitter.com/RaReTechTeam)



[.https://www.linkedin.com/company/6457766](https://www.linkedin.com/company/6457766)



[.https://github.com/piskvorky/](https://github.com/piskvorky/)



[.https://rare-technologies.com/feed/](https://rare-technologies.com/feed/)