# Word2vec

**Word2vec** is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space.[1]

Word2vec was created and published in 2013 by a team of researchers led by Tomas Mikolov at Google and patented.[2] The algorithm has been subsequently analysed and explained by other researchers.[3][4] Embedding vectors created using the Word2vec algorithm have many advantages compared to earlier algorithms[1] such as latent semantic analysis.

## Contents



## CBOW and skip grams

Word2vec can utilize either of two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or continuous skip-gram. In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption). In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words.[1][5] According to the authors' note,[6] CBOW is faster while skip-gram is slower but does a better job for infrequent words.

# Parametrization

Results of word2vec training can be sensitive to parametrization. The following are some important parameters in word2vec training.

## Training algorithm

A Word2vec model can be trained with hierarchical softmax and/or negative sampling. To approximate the conditional log-likelihood a model seeks to maximize, the hierarchical softmax method uses a Huffman tree to reduce calculation. The negative sampling method, on the other hand, approaches the maximization problem by minimizing the log-likelihood of sampled negative instances. According to the authors, hierarchical softmax works better for infrequent words while negative sampling works better for frequent words and better with low dimensional vectors.[6] As training epochs increase, hierarchical softmax stops being useful.[7]

## Sub-sampling

High frequency words often provide little information. Words with frequency above a certain threshold may be subsampled to increase training speed.[8]

## Dimensionality

Quality of word embedding increases with higher dimensionality. But after reaching some point, marginal gain will diminish.[1] Typically, the dimensionality of the vectors is set to be between 100 and 1,000.

## Context window

The size of the context window determines how many words before and after a given word would be included as context words of the given word. According to the authors' note, the recommended value is 10 for skip-gram and 5 for CBOW.[6]

# Extensions

An extension of word2vec to construct embeddings from entire documents (rather than the individual words) has been proposed.[9] This extension is called paragraph2vec or doc2vec and has been implemented in the C, Python[10][11] and Java/Scala[12] tools (see below), with the Java and Python versions also supporting inference of document embeddings on new, unseen documents.

# Word vectors for bioinformatics: BioVectors

An extension of word vectors for n-grams in biological sequences (e.g. DNA, RNA, and Proteins) for bioinformatics applications have been proposed by Asgari and Mofrad.[13] Named bio-vectors (BioVec) to refer to biological sequences in general with protein-vectors (ProtVec) for proteins (amino-acid sequences) and gene-vectors (GeneVec) for gene sequences, this representation can be widely used in applications of machine learning in proteomics and genomics. The results suggest that BioVectors can characterize

biological sequences in terms of biochemical and biophysical interpretations of the underlying patterns.[13] A similar variant, dna2vec, has shown that there is correlation between Needleman-Wunsch similarity score and cosine similarity of dna2vec word vectors.[14]

## Word vectors for Radiology: Intelligent Word Embedding (IWE)

An extension of word vectors for creating a dense vector representation of unstructured radiology reports has been proposed by Banerjee et al.[15] One of the biggest challenges with Word2Vec is how to handle unknown or out-of-vocabulary (OOV) words and morphologically similar words. This can particularly be an issue in domains like medicine where synonyms and related words can be used depending on the preferred style of radiologist, and words may have been used infrequently in a large corpus. If the word2vec model has not encountered a particular word before, it will be forced to use a random vector, which is generally far from its ideal representation.

IWE combines Word2vec with a semantic dictionary mapping technique to tackle the major challenges of information extraction from clinical texts, which include ambiguity of free text narrative style, lexical variations, use of ungrammatical and telegraphic phases, arbitrary ordering of words, and frequent appearance of abbreviations and acronyms. Of particular interest, the IWE model (trained on the one institutional dataset) successfully translated to a different institutional dataset which demonstrates good generalizability of the approach across institutions.

## Analysis

The reasons for successful word embedding learning in the word2vec framework are poorly understood. Goldberg and Levy point out that the word2vec objective function causes words that occur in similar contexts to have similar embeddings (as measured by cosine similarity) and note that this is in line with J. R. Firth's distributional hypothesis. However, they note that this explanation is "very hand-wavy" and argue that a more formal explanation would be preferable.[3]

Levy et al. (2015)[16] show that much of the superior performance of word2vec or similar embeddings in downstream tasks is not a result of the models per se, but of the choice of specific hyperparameters. Transferring these hyperparameters to more 'traditional' approaches yields similar performances in downstream tasks. Arora et al. (2016)[17] explain word2vec and related algorithms as performing inference for a simple generative model for text, which involves a random walk generation process based upon loglinear topic model. They use this to explain some properties of word embeddings, including their use to solve analogies.

## Preservation of semantic and syntactic relationships

The word embedding approach is able to capture multiple different degrees of similarity between words. Mikolov et al. (2013)[18] found that semantic and syntactic patterns can be reproduced using vector arithmetic. Patterns such as “Man is to Woman as Brother is to Sister” can be generated through algebraic operations on the vector representations of these words such that the vector representation of “Brother” - ”Man” + ”Woman” produces a result which is closest to the vector representation of “Sister” in the model. Such relationships can be generated for a range of semantic relations (such as Country–Capital) as well as syntactic relations (e.g. present tense–past tense)

## Assessing the quality of a model

Mikolov et al. (2013)[1] develop an approach to assessing the quality of a word2vec model which draws on the semantic and syntactic patterns discussed above. They developed a set of 8,869 semantic relations and 10,675 syntactic relations which they use as a benchmark to test the accuracy of a model. When assessing the quality of a vector model, a user may draw on this accuracy test which is implemented in word2vec,[19] or develop their own test set which is meaningful to the corpora which make up the model. This approach offers a more challenging test than simply arguing that the words most similar to a given test word are intuitively plausible.[1]

### Parameters and model quality

The use of different model parameters and different corpus sizes can greatly affect the quality of a word2vec model. Accuracy can be improved in a number of ways, including the choice of model architecture (CBOW or Skip-Gram), increasing the training data set, increasing the number of vector dimensions, and increasing the window size of words considered by the algorithm. Each of these improvements comes with the cost of increased computational complexity and therefore increased model generation time.[1]

In models using large corpora and a high number of dimensions, the skip-gram model yields the highest overall accuracy, and consistently produces the highest accuracy on semantic relationships, as well as yielding the highest syntactic accuracy in most cases. However, the CBOW is less computationally expensive and yields similar accuracy results.[1]

Accuracy increases overall as the number of words used increases, and as the number of dimensions increases. Mikolov et al.[1] report that doubling the amount of training data results in an increase in computational complexity equivalent to doubling the number of vector dimensions.

Altszyler and coauthors (2017) studied Word2vec performance in two semantic tests for different corpus size.[20] They found that Word2vec has a steep learning curve, outperforming another word-embedding technique (LSA) when it is trained with medium to large corpus size (more than 10 million words). However, with a small training corpus LSA showed better performance. Additionally they show that the best parameter setting depends on the task and the training corpus. Nevertheless, for skip-gram models trained in medium size corpora, with 50 dimensions, a window size of 15 and 10 negative samples seems to be a good parameter setting.

## Implementations

- C (https://github.com/tmikolov/word2vec)
- C# (https://github.com/eabdullin/Word2Vec.Net)
- Python (TensorFlow) (https://github.com/tensorflow/tensorflow/blob/r1.1/tensorflow/examples/tutorials/word2vec/word2vec_basic.py)
- Python (Gensim) (http://radimrehurek.com/gensim/models/word2vec.html)
- Java/Scala (https://github.com/deeplearning4j/deeplearning4j)
- Wikipedia2Vec (https://wikipedia2vec.github.io/demo/)[2] (http://arxiv.org/abs/1812.06280) (introduction (https://wikipedia2vec.github.io/wikipedia2vec/))

## See also

- Autoencoder
- Document-term matrix
- Feature extraction
- Feature learning
- Neural network language models
- Vector space model
- Thought vector
- fastText

- GloVe
- Normalized compression distance

# References


1. Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781 (https://arxiv.org/abs/1301.3781) [cs.CL (https://arxiv.org/archive/cs.CL)].
2. [1] (https://patents.google.com/patent/US9037464B1/en), "Computing numeric representations of words in a high-dimensional space"
3. Goldberg, Yoav; Levy, Omer (2014). "word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method". arXiv:1402.3722 (https://arxiv.org/abs/1402.3722) [cs.CL (https://arxiv.org/archive/cs.CL)].
4. Řehůřek, Radim. *Word2vec and friends* (https://www.youtube.com/watch?v=wTp3P2UnTfQ) (Youtube video). Retrieved 14 August 2015.
5. Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S.; Dean, Jeff (2013). *Distributed representations of words and phrases and their compositionality*. Advances in Neural Information Processing Systems. arXiv:1310.4546 (https://arxiv.org/abs/1310.4546). Bibcode:2013arXiv1310.4546M (https://ui.adsabs.harvard.edu/abs/2013arXiv1310.4546M).
6. "Google Code Archive - Long-term storage for Google Code Project Hosting" (https://code.google.com/archive/p/word2vec/). *code.google.com*. Retrieved 13 June 2016.
7. "Parameter (hs & negative)" (https://groups.google.com/forum/#!msg/word2vec-toolkit/WUWad9fL0jU/LdbWy1jQjUIJ). *Google Groups*. Retrieved 13 June 2016.
8. "Visualizing Data using t-SNE" (http://jmlr.csail.mit.edu/papers/volume9/vandermaaten08a/vandermaaten08a.pdf) (PDF). *Journal of Machine Learning Research, 2008. Vol. 9, pg. 2595*. Retrieved 18 March 2017.
9. Le, Quoc; et al. (2014). "Distributed Representations of Sentences and Documents". arXiv:1405.4053 (https://arxiv.org/abs/1405.4053) [cs.CL (https://arxiv.org/archive/cs.CL)].
10. "Doc2Vec tutorial using Gensim" (https://medium.com/@klintcho/doc2vec-tutorial-using-gensim-ab3ac03d3a1). Retrieved 2 August 2015.
11. "Doc2vec for IMDB sentiment analysis" (https://github.com/piskvorky/gensim/blob/develop/docs/notebooks/doc2vec-IMDB.ipynb). Retrieved 18 February 2016.
12. "Doc2Vec and Paragraph Vectors for Classification" (http://deeplearning4j.org/doc2vec.html). Retrieved 13 January 2016.
13. Asgari, Ehsaneddin; Mofrad, Mohammad R.K. (2015). "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4640716). *PLOS One*. **10** (11): e0141287. arXiv:1503.05140 (https://arxiv.org/abs/1503.05140). Bibcode:2015PLoSO..1041287A (https://ui.adsabs.harvard.edu/abs/2015PLoSO..1041287A). doi:10.1371/journal.pone.0141287 (https://doi.org/10.1371%2Fjournal.pone.0141287). PMC 4640716 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4640716). PMID 26555596 (https://pubmed.ncbi.nlm.nih.gov/26555596).
14. Ng, Patrick (2017). "dna2vec: Consistent vector representations of variable-length k-mers". arXiv:1701.06279 (https://arxiv.org/abs/1701.06279) [q-bio.QM (https://arxiv.org/archive/q-bio.QM)].
15. Banerjee, Imon; Chen, Matthew C.; Lungren, Matthew P.; Rubin, Daniel L. (2018). "Radiology report annotation using intelligent word embeddings: Applied to multi-institutional chest CT cohort" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5771955). *Journal of Biomedical Informatics*. **77**: 11–20. doi:10.1016/j.jbi.2017.11.012 (https://doi.org/10.1016%2Fj.jbi.2017.11.012). PMC 5771955 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5771955). PMID 29175548 (https://pubmed.ncbi.nlm.nih.gov/29175548).

16. Levy, Omer; Goldberg, Yoav; Dagan, Ido (2015). "Improving Distributional Similarity with Lessons Learned from Word Embeddings" (http://www.aclweb.org/anthology/Q15-1016). *Transactions of the Association for Computational Linguistics*. Transactions of the Association for Computational Linguistics. **3**: 211–225. doi:10.1162/tacl_a_00134 (https://doi.org/10.1162%2Ftacl_a_00134).
17. Arora, S; et al. (Summer 2016). "A Latent Variable Model Approach to PMI-based Word Embeddings" (http://aclweb.org/anthology/Q16-1028). *Transactions of Assoc. Of Comp. Linguistics*. **4**: 385–399. doi:10.1162/tacl_a_00106 (https://doi.org/10.1162%2Ftacl_a_00106) – via ACLWEB.
18. Mikolov, Tomas; Yih, Wen-tau; Zweig, Geoffrey (2013). "Linguistic Regularities in Continuous Space Word Representations". *HLT-Naacl*: 746–751.
19. "Gensim - Deep learning with word2vec" (https://radimrehurek.com/gensim/models/word2vec.html). Retrieved 10 June 2016.
20. Altszyler, E.; Ribeiro, S.; Sigman, M.; Fernández Slezak, D. (2017). "The interpretation of dream meaning: Resolving ambiguity using Latent Semantic Analysis in a small corpus of text". *Consciousness and Cognition*. **56**: 178–187. arXiv:1610.01520 (https://arxiv.org/abs/1610.01520). doi:10.1016/j.concog.2017.09.004 (https://doi.org/10.1016%2Fj.concog.2017.09.004). PMID 28943127 (https://pubmed.ncbi.nlm.nih.gov/28943127).

---