

[\(https://www.gavagai.io/\)](https://www.gavagai.io/)

A Brief History of Word Embeddings

One of the strongest trends in Natural Language Processing (NLP) at the moment is the use of **word embeddings**, which are vectors whose relative similarities correlate with semantic similarity. Such vectors are used both as an end in itself (for computing similarities between terms), and as a representational basis for downstream NLP tasks like text classification, document clustering, part of speech tagging, named entity recognition, sentiment analysis, and so on. That this is a trend is obvious when looking at the proceedings from the recent large conferences in NLP, e.g. *ACL* (<https://aclanthology.info/>) or *EMNLP* (<https://aclweb.org/anthology/D/D15/>). For the first time (ever), semantics was the dominating subject at EMNLP ("Empirical Methods in NLP") this year. In fact, some people even suggested the conference be renamed "Embedding Methods in NLP", due to the large amount of papers covering various types of methods, applications and evaluations for word embeddings.

This is a positive trend (if you like semantics and embeddings), and there is a lot of progress being made in NLP currently. However, many recent publications (and talks) on word embeddings are surprisingly oblivious of the large body of previous work in fields like computer science, cognitive science, and computational linguistics. Apart from this being bad academic manners, it also risks delaying progress by reinventing the wheel rather than building on existing knowledge.

The point of this post is to provide a brief history of word embeddings, and to summarize the current state of the art. The usual caveat applies: this is by no means meant as a complete list of all relevant research on the subject; on the contrary, it is intended as a reference and starting point for those interested in exploring the field further.

First, a note on terminology. **Word embedding** seems to be the dominating term at the moment, no doubt because of the current popularity of methods coming from the *deep learning* (https://en.wikipedia.org/wiki/Deep_learning) community. In computational linguistics, we often prefer the term **distributional semantic**



model (since the underlying semantic theory is called *distributional semantics* (https://en.wikipedia.org/wiki/Distributional_semantics)). There are also many other alternative terms in use, from the very general **distributed representation** to the more specific **semantic vector space** or simply **word space**. For consistency, I will adhere to current practice and use the term word embedding in this post.

Word embeddings are based on the idea that contextual information alone constitutes a viable representation of linguistic items, in stark contrast to formal linguistics and the Chomsky tradition. This idea has its theoretical roots in structuralist linguistics and ordinary language philosophy, and in particular in the works of *Zellig Harris* (https://en.wikipedia.org/wiki/Zellig_Harris), *John Firth* (https://en.wikipedia.org/wiki/John_Rupert_Firth), and *Ludwig Wittgenstein* (https://en.wikipedia.org/wiki/Ludwig_Wittgenstein), all publishing important works in the 1950s (in the case of Wittgenstein, posthumously). The earliest attempts at using feature representations to quantify (semantic) similarity used hand-crafted features. Charles Osgood's *semantic differentials* (https://en.wikipedia.org/wiki/Semantic_differential) in the 1960s is a good example, and similar representations were also used in early works on connectionism and artificial intelligence in the 1980s.

Methods for using automatically generated contextual features were developed more or less simultaneously around 1990 in several different research areas. One of the most influential early models was *Latent Semantic Analysis/Indexing* (https://en.wikipedia.org/wiki/Latent_semantic_analysis) (LSA/LSI), developed in the context of information retrieval, and the precursor of today's **topic models**. At roughly the same time, there were several different models developed in research on artificial neural networks that used contextual representations. The most well-known of these are probably *Self Organizing Maps* (https://en.wikipedia.org/wiki/Self-organizing_map) (SOM) and *Simple Recurrent Networks* (https://en.wikipedia.org/wiki/Recurrent_neural_network) (SRN), of which the latter is the precursor to today's **neural language models**. In computational linguistics, Hinrich Schütze developed models that were based on word co-occurrences, which was also used in *Hyperspace Analogue to Language* (https://en.wikipedia.org/wiki/Hyperspace_Analogue_to_Language) (HAL) that was developed as a model of semantic memory in cognitive science.

Later developments are basically only refinements of these early models. **Topic models** are refinements of LSA, and include methods like *probabilistic LSA* (https://en.wikipedia.org/wiki/Probabilistic_latent_semantic_analysis) (PLSA) and *Latent Dirichlet Allocation* (https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation) (LDA).

(LDA). **Neural language models** are based on the same application of neural networks as SRN and include architectures like Convolutional Neural Networks (https://en.wikipedia.org/wiki/Convolutional_neural_network) (CNN) and Autoencoders (<https://en.wikipedia.org/wiki/Autoencoder>). **Distributional semantic models** are often based on the same type of representation as HAL, and includes models like Random Indexing (https://en.wikipedia.org/wiki/Random_indexing) and BEAGLE (<http://www.indiana.edu/~clcl/BEAGLE/>).

The main difference between these various models is the type of contextual information they use. LSA and topic models use *documents* as contexts, which is a legacy from their roots in information retrieval. Neural language models and distributional semantic models instead use *words* as contexts, which is arguably more natural from a linguistic and cognitive perspective. These different contextual representations capture different types of semantic similarity; the document-based models capture **semantic relatedness** (e.g. "boat" – "water") while the word-based models capture **semantic similarity** (e.g. "boat" – "ship"). This very basic difference is too often misunderstood.

Speaking of common misunderstandings, there are two other myths that need debunking:

- There is no need for *deep* neural networks in order to build good word embeddings. In fact, two of the most successful and acknowledged recent models – the Skipgram and CBoW models included in the word2vec (<https://code.google.com/p/word2vec/>) library – are shallow neural networks of the same flavor as the original SRN.

- There is no qualitative difference between (current) *predictive* neural network models and *count-based* distributional semantics models. Rather, they are different computational means to arrive at the same type of semantic model; several recent papers have demonstrated both theoretically and empirically the correspondence between these different types of models [Levy and Goldberg (2014) (<https://levyomer.files.wordpress.com/2014/09/neural-word-embeddings-as-implicit-matrix-factorization.pdf>), Pennington et al. (2014) (<http://www.nlp.stanford.edu/pubs/glove.pdf>), Österlund et al. (2015) (<http://aclweb.org/anthology/D/D15/D15-1024.pdf>)].

So that's a very brief history and a couple of clarifications. What about the current state of the art? The boring answer is that it depends on what task you want ' solve, and how much effort you want to spend on optimizing the model. Th somewhat more enjoyable answer is that you will probably do fine whichever

current method you choose, since they are more or less equivalent. A good bet is to use a factorized model ~~(<https://www.gavagai.io>)~~ or using explicit factorization of a distributional semantic model (available in e.g. the [PyDSM](https://github.com/jimmycallin/pydsm) (<https://github.com/jimmycallin/pydsm>) python library or the [GloVe](http://nlp.stanford.edu/projects/glove/) (<http://nlp.stanford.edu/projects/glove/>) implementation), or using a neural network model like those implemented in [word2vec](https://code.google.com/p/word2vec/) (<https://code.google.com/p/word2vec/>) – since they produce state of the art results ([Österlund et al., 2015](https://www.aclweb.org/anthology/D15-1024) (<https://www.aclweb.org/anthology/D15-1024>)) and are robust across a wide range of semantic tasks ([Schnabel et al., 2015](https://www.aclweb.org/anthology/D15-1036) (<https://www.aclweb.org/anthology/D15-1036>)).

For those interested in playing around with word embeddings, I recommend the following libraries:

Standalone implementations

- [word2vec](https://code.google.com/p/word2vec/) (<https://code.google.com/p/word2vec/>) (also available in e.g. [Spark MLlib](http://spark.apache.org/mllib/) (<http://spark.apache.org/mllib/>) and [DL4J](http://deeplearning4j.org/) (<http://deeplearning4j.org/>))
- [GloVe](http://nlp.stanford.edu/projects/glove/) (<http://nlp.stanford.edu/projects/glove/>)

Frameworks

- [S-Space](https://github.com/fozziethebeat/S-Space) (<https://github.com/fozziethebeat/S-Space>) (Java)
- [SemanticVectors](https://github.com/semanticvectors/semanticvectors) (<https://github.com/semanticvectors/semanticvectors>) (Java)
- [Gensim](https://radimrehurek.com/gensim/) (<https://radimrehurek.com/gensim/>) (Python)
- [PyDSM](https://github.com/jimmycallin/pydsm) (<https://github.com/jimmycallin/pydsm>) (Python)
- [DISSECT](http://clic.cimec.unitn.it/composes/toolkit/) (<http://clic.cimec.unitn.it/composes/toolkit/>) (Python)



Get Started for Free

[\(https://www.gavagai.io/\)](https://www.gavagai.io/)

Learn about our AI-powered text analysis tool in a **Personal Demo**.

Then get a **Free Trial** to test-run Gavagai Explorer using your own data.



Thousands of texts
in any language
analyzed in minutes

[Get a demo \(https://www.gavagai.io/l/free-demo/\)](https://www.gavagai.io/l/free-demo/)

Other tools

[Gavagai Monitor \(https://monitor.gavagai.se/\)](https://monitor.gavagai.se/)

[Gavagai Explorer API \(https://api.gavagai.se/explorer/v1/docs/\)](https://api.gavagai.se/explorer/v1/docs/)

[Gavagai Simple API \(https://developer.gavagai.se/\)](https://developer.gavagai.se/)

[Gavagai Lexicon \(https://lexicon.gavagai.se/\)](https://lexicon.gavagai.se/)

[Gavagai Labs \(https://www.gavagai.io/labs/\)](https://www.gavagai.io/labs/)

[Integrations \(https://www.gavagai.io/integrations/\)](https://www.gavagai.io/integrations/)

[Plugins & Addons \(https://www.gavagai.io/plugins-addons/\)](https://www.gavagai.io/plugins-addons/)

Information

[About \(https://www.gavagai.io/about/\)](https://www.gavagai.io/about/)

[What's different \(https://www.gavagai.io/whats-different/\)](https://www.gavagai.io/whats-different/)

[Documentation \(https://docs.gavagai.io/\)](https://docs.gavagai.io/)

[Contact \(https://www.gavagai.io/contact/\)](https://www.gavagai.io/contact/)

[Management team \(https://www.gavagai.io/meet-the-team/\)](https://www.gavagai.io/meet-the-team/)

[Careers \(https://www.gavagai.io/career/\)](https://www.gavagai.io/career/)

[Status \(http://status.gavagai.io/\)](http://status.gavagai.io/)

[Legal Information \(https://www.gavagai.io/legal/\)](https://www.gavagai.io/legal/)

Customer experience dictionary

[Text Analytics Dictionary \(https://www.gavagai.io/text-analytics/\)](https://www.gavagai.io/text-analytics/)



[Customer Feedback \(https://www.gavagai.io/text-analytics/customer-feedback/\)](https://www.gavagai.io/text-analytics/customer-feedback/)

[Employee Insight \(https://www.gavagai.io/text-analytics/employee-insight/\)](https://www.gavagai.io/text-analytics/employee-insight/)

[Natural Language Processing \(https://www.gavagai.io/text-analytics/natural-language-processing/\)](https://www.gavagai.io/text-analytics/natural-language-processing/)

[Natural Language Understanding \(https://www.gavagai.io/text-analytics/natural-language-understanding/\)](https://www.gavagai.io/text-analytics/natural-language-understanding/)

[Net Promoter Score \(https://www.gavagai.io/text-analytics/net-promoter-score/\)](https://www.gavagai.io/text-analytics/net-promoter-score/)

[Predictive Analytics \(https://www.gavagai.io/text-analytics/predictive-analytics/\)](https://www.gavagai.io/text-analytics/predictive-analytics/)

[Sentiment Analysis and Opinion Mining \(https://www.gavagai.io/text-analytics/sentiment-analysis-opinion-mining/\)](https://www.gavagai.io/text-analytics/sentiment-analysis-opinion-mining/)

[Social Media Analytics \(https://www.gavagai.io/text-analytics/social-media-analytics/\)](https://www.gavagai.io/text-analytics/social-media-analytics/)

[Speech Analytics \(Speech to Text\) \(https://www.gavagai.io/text-analytics/speech-analytics/\)](https://www.gavagai.io/text-analytics/speech-analytics/)

[Text Mining \(https://www.gavagai.io/text-analytics/text-mining/\)](https://www.gavagai.io/text-analytics/text-mining/)

[Topic Modelling \(https://www.gavagai.io/text-analytics/topic-modelling/\)](https://www.gavagai.io/text-analytics/topic-modelling/)

[Voice of the Customer \(https://www.gavagai.io/text-analytics/voice-of-the-customer/\)](https://www.gavagai.io/text-analytics/voice-of-the-customer/)

[Word Embeddings \(https://www.gavagai.io/text-analytics/word-embeddings/\)](https://www.gavagai.io/text-analytics/word-embeddings/)

Contact

Visitors: Kungsgatan 60 11122 Stockholm

Postal address: Gavagai AB. Kivra: 556745-2924. 106 31 Stockholm, Sweden

LEI: 98450073053A9FP76Y37

https://twitter.com/gavagai_corp https://www.youtube.com/channel/UCDN5ry8_ET4cQ

© Copyright 2020. Gavagai AB. All rights reserved.

