# Exploring and visualizing distributional models using graphs

Emiel van Miltenburg
VU University Amsterdam
emiel.van.miltenburg@vu.nl

## 1 Introduction

Visualization of distributional semantic models is often done using dimensionality reduction to a two-dimensional space (e.g. Speelman et al. 2013). While this may be useful in some cases, the results are often hard to interpret and the method itself is not very flexible; there are little to no parameters that one can tweak to produce an aesthetically more pleasing outcome. With this paper, I provide the outline of a graph-based method to visualize distributional models (inspired by Font et al. 2012). An implementation in Python is available online.[1] This makes a useful tool for teaching, as well as data exploration.

## 2 Building a network

A distributional model can be converted into a graph simply by generating edges between all words in the vocabulary, with the similarity between each pair of words as the edge weight.This method ensures that no information is lost in the conversion, but it also produces a very dense graph that is hard to visualize. I propose to only generate edges between lexical items if those lexical items are sufficiently similar. This not only makes the graph more sparse, but it also makes it easier to detect clusters of related words. I will now discuss important parameters that can be tuned to get a different network.

**Edge weights** There are two general weighting approaches: *similarity-based*, where the weight of the edge connecting $w_1$ and $w_2$ is simply the (cosine) similarity between $w_1$ and $w_2$; and *rank-based*, where the weight of the edge connecting $w_1$ and $w_2$ is the average of the relative rank of $w_1$ for $w_2$, and the relative rank of $w_2$ for $w_1$. Font et al. (2012) uses similar measures in his tag recommendation system.

**Edge generation** There are several different similarity criteria that we can use to generate a list of edges. The more strict we make the parameters $n$ and $\theta$ (a smaller value for $n$, or a higher value for $\theta$), the more sparse the network becomes. By default, $n = 5$ and $\theta = 0.5$

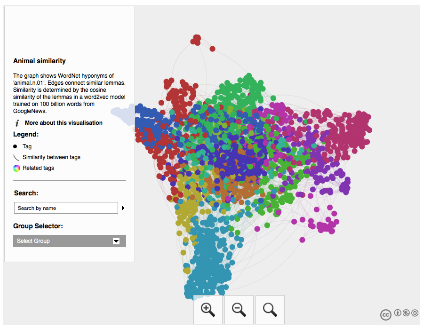
1. For each word $w$, create edges between $w$ and its top-n similar words.
2. Only create an edge between $w_1$ and $w_2$ if $w_1$ is in the top-n of $w_2$ and vice versa.
3. For each word $w$, create edges between $w$ and all words above a similarity threshold $\theta$.
4. For each word $w$, create edges between $w$ and its top-n similar words iff their similarity to $w$ exceeds the similarity threshold $\theta$.

Alternatively, the Pathfinder algorithm (Schvaneveldt 1990; we implemented MST-Pathfinder, Quirin et al. 2008) may also be used to prune the graph (c.f. Cohen 2008), but this might make it *too* sparse; the algorithm produces the union of possible minimal spanning trees. A solution might be to use that as a base, and *add* edges. We do not explore this possibility here.

[1] https://github.com/evanmiltenburg/dm-graphs

## 3 Analyzing the graph

After constructing the graph, it can be partitioned into clusters (roughly corresponding to topics or word senses). Generally speaking, there are two types of clustering: *hard clustering*, where every node can only be part of one cluster, and *soft clustering*, where nodes can be part of multiple clusters (i.e. clusters may overlap). A popular hard clustering algorithm is known as the Louvain method (Blondel et al., 2008), which aims to optimize the modularity of each cluster. Soft clustering methods are still a very active research area, but the clique percolation method (Palla et al., 2005) is a common approach. For an extensive overview of clustering methods, see Fortunato (2010).



**Animal similarity**

The graph shows WordNet hyponyms of 'animal.n.01'. Edges connect similar lemmas. Similarity is determined by the cosine similarity of the lemmas in a word2vec model trained on 100 billion words from GoogleNews.

**Figure 1:** Graph exported as an interactive website. Colors indicate different clusters. Zoom in for more details.

## 4 Visualization

The easiest way to visualize a graph is by using Gephi, an open source program for network analysis and visualization (Bastian et al., 2009). (Their website features a quote from the community, calling it 'Photoshop™ for graphs'.) It comes installed with several different layout algorithms. Gephi also provides tools for post-processing, for example to emphasize particular parts of the graph. Finally, there are many plugins available that expand Gephi's functionality. As an example, I created a graph showing the relatedness of different kinds of animals according to the pre-trained Google News `word2vec` model. The graph is made using the top-n method. After using the Louvain method to cluster the graph, I exported it from Gephi as an interactive website (figure 1) using Scott Hale's Sigma.js plugin. Users can look for a particular lemma, see how it is connected to other lemmas in the graph, and which cluster it belongs to. This provides a new means to explore and evaluate distributional semantic models.

## References

Bastian, M., S. Heymann, M. Jacomy, et al. (2009). Gephi: an open source software for exploring and manipulating networks. *ICWSM 8*, 361–362.

Blondel, V. D., J.-L. Guillaume, R. Lambiotte, and E. Lefebvre (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment 2008*(10), P10008.

Cohen, T. (2008). Exploring medline space with random indexing and pathfinder networks. In *AMIA Annual Symposium Proceedings*, Volume 2008, pp. 126. American Medical Informatics Association.

Font, F., J. Serrà, and X. Serra (2012). Folksonomy-based tag recommendation for online audio clip sharing.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports 486*(3), 75–174.

Palla, G., I. Derényi, I. Farkas, and T. Vicsek (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature 435*(7043), 814–818.

Quirin, A., O. Cordón, V. P. Guerrero-Bote, B. Vargas-Quesada, and F. Moya-Anegón (2008). A quick mst-based algorithm to obtain pathfinder networks (, n- 1). *Journal of the American Society for Information Science and Technology 59*(12), 1912–1924.

Schvaneveldt, R. W. (1990). *Pathfinder associative networks: Studies in knowledge organization.* Ablex Publishing.

Speelman, T., K. Wielfaert, and D. Heylen (2013). Interactive visualizations of semantic vector spaces for lexicological analysis. *TALN-RÉCITAL 2013*, 154.