Exploring and visualizing distributional models using graphs

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

1 Introduction

Visualization of distributional models is often done using dimensionality reduction to a two-dimensional space (e.g. Speelman 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 Many graph clustering and layout algorithms do not rely on edge weights, but only on the fact that two nodes are connected through an edge. To some extent, then, edge weights aren't necessary if the graph only includes edges between similar words. But if one chooses to generate a large amount of edges, weights are the only way to distinguish between related and unrelated nodes. 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. See also Font et al. (2012), who 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 θ (a smaller value for n, or a higher value for θ), the more sparse the network becomes.

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 θ.
4. For each word w, create edges between w and its top-n similar words iff their similarity to w exceeds the similarity threshold θ.

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. Depending on the model the graph is based on, clusters can roughly be interpreted as senses or topics. 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. Put otherwise: hard clustering methods do not allow clusters to overlap, whereas soft clustering does permit this. 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).

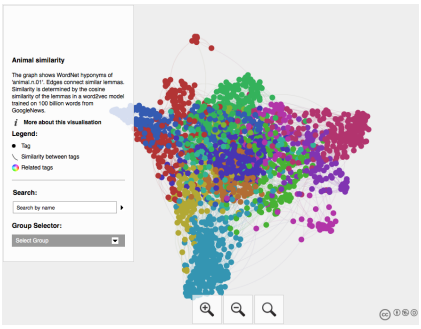

Figure 1: Screenshot of Gephi website export. The graph shows hyponyms (lemmas) of `animal.n.01` in WordNet. Edge weight is determined by cosine similarity on the pre-trained Google News `word2vec` model. The graph is reduced using the top-n method, where $n = 5$. Colors indicate clusters, determined through the Louvain method.

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 some community-provided plugins to export the final product as an interactive website (see figure 1). With the methods outlined in this paper, we have a complete pipeline to go from a model to an insightful demo.

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. W. K. H. D. (2013). Interactive visualizations of semantic vector spaces for lexicological analysis. *TALN-RÉCITAL 2013*, 154.