# Graph Visualization With Latent Variable Models

Juuso Parkkinen, Kristian Nybo, Jaakko Peltonen, and SamuelKaski
Aalto University Schoold of Science and Technology,
Department of Information and Computer Science,
Helsinki Institute for Information Technology HIIT
P.O. Box 15400, FI-00076 Aalto, Finland
firstname.lastname@tkk.fi

## ABSTRACT

Graphs are central representations of information in many domains including biological and social networks. Graph visualization is needed for discovering underlying structures or patterns within the data, for example communities in a social network, or interaction patterns between protein complexes. Existing graph visualization methods, however, often fail to visualize such structures, because they focus on local details rather than global structural properties of graphs. We suggest a novel modeling-driven approach to graph visualization: As usually in modeling, choose the (generative) model such that it captures what is important in the data. Then visualize similarity of the graph nodes with a suitable multidimensional scaling method, with similarity given by the model; we use a multidimensional scaling method optimized for a rigorous visual information retrieval task. We show experimentally that the resulting method outperforms existing graph visualization methods in finding and visualizing global structures in graphs.

## Keywords

Complex networks, graph visualization, latent variable model, nonlinear dimensionality reduction

## 1. INTRODUCTION

Complex networks are actively studied in many fields of science. For example, epidemiologists analyze social networks to understand and predict how epidemics spread, and biologists study protein–protein interaction networks to gain insight into biological functions and diseases. An established way of exploring the structure of any complicated data set is to visualize it. The most common approach to visualizing networks is straight-line graph drawing: each node in the network is drawn as a glyph on the plane, and a link between two nodes is drawn as a straight line between the glyphs. The task of the visualization algorithm is then to arrange the nodes so that a good visualization is produced.


*MLG* '10 Washington DC, USA


Because it is in general impossible to reveal every aspect of a complex and large data set in a single visualization, every visualization is necessarily a compromise. The best we can do is to choose what we consider to be the most important aspect of the graph and then design a visualization that shows that aspect as effectively as possible. We will see in Section 2 that most existing graph drawing methods are not based on an explicit choice of what to visualize. For example, force-directed algorithms, the oldest and most popular class of methods, are formulated purely in terms of local properties of the graph: two nodes connected by an edge should be drawn close to each other, but nodes should not be allowed to overlap. It is hard to say what a visualization based on this local principle will reveal about global structural properties, unless the structure of the graph is very simple and regular, as in the case of a grid or a mesh. Perhaps because of this, authors of graph drawing methods have traditionally tested their methods on graphs with simple and regular structures (see, *e.g.*, [5, 7, 29]). We will see that conventional graph drawing methods do fail to reveal global structure in more complex graphs.

To meet the challenges posed by complex graphs, we propose a new approach to graph visualization. If we were to visualize only the nodes of a graph, a natural principle would be that nodes placed nearby in the layout should be 'similar' in some respect. For graphs where no separate node features are available, the only information about a node is what other nodes it links to; therefore, it is natural to assume that nodes nearby on the display should have similar link distributions.

Directly comparing the observed links from two nodes would yield only a noisy measure of similarity; in many domains the observed links represent stochastic measurements, such as measurements of gene interactions, and an observed graph contains only a sample of possible links. It is then reasonable to assume that the links arise from underlying *link distributions*; more generally, the distributions themselves can arise from several *latent processes* that generate links between nodes. If the activities of such latent processes were known for each node, they could be used to rigorously compute node similarities, which could then be used to optimize a graph layout.

We will estimate the link distributions of the nodes, and the underlying latent processes, by learning a generative latent variable model for the links in a graph. The similarities of nodes can then be compared by using any of several common distance measures between distributions.

Given a good estimate of node similarities, we want to

produce a rigorous visualization that will place nodes with similar link distributions close-by on the display; in addition to yielding an easily interpretable layout of the nodes, a side benefit is that the links from such groups of nodes form intuitive bundles of links that start and end at similar nodes. To produce such a visualization, we will use a novel rigorous framework for visualization which we introduced recently.

As far as we know, this principle for visualizing graphs is new. We will start by discussing earlier graph drawing principles.

# 2. GRAPH DRAWING

In their classic paper [5], Fruchterman and Reingold formulate the problem of graph drawing as producing a drawing "according to some generally accepted aesthetic criteria": distribute the vertices evenly in the image, make edge lengths uniform, reflect inherent symmetry, minimize edge crossings, and conform to the drawing frame. Fruchterman and Reingold note that their algorithm does not explicitly strive for these goals, but that it does well in terms of the three first. They state their drawing method is based on two principles: that *vertices connected by an edge should be drawn close to each other*; and that vertices should not be drawn *too* close to each other.

This is the task definition that is implicitly used in both classical and very recent papers on force-based and spectral graph drawing algorithms. We say 'implicitly' because most papers that we cite in these categories do not explicitly formulate a task, but the task can be inferred from their choices of comparison methods and evaluation criteria.

## 2.1 Force-Based Methods

Force-based algorithms are arguably the most established and wide-spread class of graph drawing algorithms. Simple force-based algorithms are commonly used also in graph visualization papers such as [3, 11] where the graph layout is only a starting point.

The principle can be explained by a physical analogy: imagine each edge is a spring with an equilibrium length $k$, and each node is a steel ring to which its inbound edges are attached. Place the nodes at arbitrary initial positions in the plane, and then release them; they move according to forces exerted on them by the extending and contracting springs, until the system finds an equilibrium, which is the layout. Due to this analogy, force-based methods are sometimes called *spring embedders*.

The classical Fruchterman-Reingold (FR) algorithm follows this analogy, but instead of simulating physical forces, it uses an iterative computation with similar properties. In brief, at each iteration, each node $x$ is displaced by contributions from other nodes: each of the other nodes $z$ repulses $x$ by a displacement $k^2/d(x,z)$ where $d(x,z)$ is the distance between $x$ and $z$; on the other hand, each node $y$ connected to $x$ by an edge attracts $x$ by a displacement $d(x,y)/k$. Similar ideas are used in other classical force-based methods. Fruchterman and Reingold also introduce a faster grid-variant of their algorithm.

The classical force-based algorithms are slow for large graphs [10]: even one of the fastest algorithms [4] has time complexity $\mathcal{O}(n^3)$ for graphs with $n$ nodes. This has led to development of so-called multi-level algorithms, introduced by Walshaw [29] and Hadany and Harel [8]. The core idea of both algorithms is the same; we describe Walshaw's algorithm, which we use as a comparison method in Section 4.

Walshaw's algorithm starts by creating a sequence of increasingly coarse approximations of the graph, call it $G_0$, by finding a maximal independent subset $S$ of edges. A set of edges is independent if no two edges in the set are incident on the same node; it is maximal if no more edges can be added to the set without breaking the independence criterion. Walshaw finds such subsets with an approximate algorithm. Once an (approximate) maximal independent subset of edges $S$ is found, a coarse graph $G_1$ approximating $G_0$ is created, by collapsing the edges in $S$: all edges in $S$ are deleted, and any pair of nodes connected by a deleted edge are replaced by a single node. Next, an even coarser approximation $G_2$ is created by applying the coarsening step to $G_1$, and so on. Once there is an approximation $G_n$ with sufficiently few nodes, we compute a layout for it with the Fruchterman-Reingold grid-variant algorithm. From this layout, an initial layout is interpolated for $G_{n-1}$: each node in $G_n$ that represents a pair of nodes in $G_{n-1}$ is replaced by two nodes connected by an edge. This initial layout is refined by the Fruchterman-Reingold algorithm, and is then used to interpolate a layout for $G_{n-2}$, and so on until we have a layout for the original graph.

Since Walshaw published his algorithm, several other multilevel methods have been developed; see [7] for a comparison. They are all at least an order of magnitude faster than Fruchterman and Reingold's grid-variant algorithm, and generally produce much better layouts [7].

## 2.2 Spectral Methods

Spectral graph drawing algorithms offer another solution to the problem of computational complexity in traditional graph drawing algorithms. Although the spectral approach was introduced as early as 1970 [9], it has only recently become popular.

The term 'spectral method' typically refers to any graph layout algorithm that bases its layout on the eigenvectors (*i.e.*, the spectral decomposition) of some matrix derived from the graph. As an example of this class of algorithms, we describe Hall's original method [9], following Koren's exposition in [12]. A more detailed derivation can be found in [13].

Hall's method computes layouts of a weighted graph $G(V,E)$ with $n$ nodes, into $m < n$ dimensions. Consider first the case $m = 1$. The one-dimensional graph drawing is formulated as finding an $x \in \mathbb{R}^n$, where $x(i)$ is the coordinate of the $i$th node, such that $x$ solves the constrained minimization problem

$$\min_x E(x) = \sum_{(i,j)\in E} w_{i,j}(x(i)-x(j))^2 \qquad \text{given: } Var(x) = 1, \tag{1}$$

where $Var(x)$ is the variance of $x$, that is,
$Var(x) = \frac{1}{n}\sum_{i=1}^{n}(x(i)-\mu)^2$, where $\mu$ is the mean of $x$.

In words, the task is to find a layout $x$ that minimizes the distances between nodes that are connected by an edge; the constraint $Var(x) = 1$ prevents nodes from being mapped too close to each other. Thus we can interpret the minimization problem (1) as a (one-dimensional) mathematical formulation of Fruchterman and Reingold's two graph drawing principles: nodes connected by an edge should be drawn close to each other; and nodes should not be drawn *too* close to each other. To produce a two-dimensional drawing, the

method simply computes another coordinate vector $y$ that satisfies (1), but it is additionally required that there be no correlation between $x$ and $y$. It is easily shown that these vectors $x$ and $y$ are the second and third smallest eigenvectors of the Laplacian matrix of the graph.

Other spectral methods generally follow the same structure; mainly the matrix whose eigenvectors are calculated varies. The classical MDS (CMDS) algorithm by Kruskal and Seery [14], one of our comparison methods, uses the shortest path pairwise distance matrix. Spectral methods are significantly faster than the fastest force-based methods, but they have a tendency to produce layouts with many overlapping nodes [7]. Civril et al., who recently independently rediscovered the algorithm, found that CMDS suffers less from this tendency than other popular spectral methods[2].

## 2.3 Visualizing Clusters: Edge-Repulsion LinLog

Although multi-level force-based algorithms and spectral methods offer great improvements over classical drawing algorithms, these improvements are mainly technical innovations that allow the algorithms to scale to much larger graphs than their predecessors could handle; the graph drawing task is still the same. Noack recently proposed ERLinLog (Edge-Repulsion LinLog) [20], a graph drawing method specifically designed to show graph clusters, a task that Noack shows is not only different from, but actually in conflict with the classical Fruchterman-Reingold task.

Informally, Noack calls a subgraph a cluster if it has many internal edges and few edges to the remaining graph; he introduces various clustering measures with which the notion can be formalized. The goal of ERLinLog is to show clusters in the layout by grouping densely connected nodes (nodes in the same cluster) and separating sparsely connected nodes (nodes not in the same cluster). This is accomplished by minimizing the cost function

$$U_{\mathrm{ERLinLog}} = \sum_{\{u,v\}\in E} \|p(u) - p(v)\| - \sum_{\{u,v\}\in V^{(2)}} \deg u \deg v \log \|p(u) - p(v)\|, \quad (2)$$

where $p(u)$ and $p(v)$ are the coordinates of nodes $u$ and $v$ in the layout. Noack presents a theorem stating that a layout with minimal cost *minimizes the ratio of the mean distance between connected nodes to the mean distance between all nodes.*

There have been some other papers with tasks similar to Noack's. Van Ham and Van Wijk modified an earlier variant of LinLog to create an interactive visualization system for small-world graphs such as social networks [26], Lehmann and Kottler also proposed an algorithm for drawing clustered graphs, but based on the idea of computing a special kind of spanning tree [15].

## 2.4 Summary

We have seen that most papers that introduce a new graph drawing method do not explicitly state what exactly their method tries to visualize. Because it is generally impossible for a single visualization to capture every possible aspect of a graph, however, every graph layout method necessary favors some aspects over others, and hence has at least an implicit objective. We saw that methods without explicitly stated objectives, which includes most force-based and spectral methods, tend to share implicitly the goals of Fruchterman and Reingold [5]: to place nodes connected by an edge close to each other, to keep edge lengths uniform, and to minimize the number of edge crossings. Of all the methods reviewed in this chapter, only Noack's Edge-Repulsion LinLog [20], which visualizes graph clusters, can be said to be based on a clear visualization task.

# 3. MODEL-BASED GRAPH VISUALIZATION

Our proposed graph visualization principle, which was motivated in the previous sections, includes the following steps: (1) Devise a latent variable model for capturing essential structure from a graph; (2) Compute distances between the graph nodes in the latent space; and (3) Apply a non-linear dimensionality reduction to visualize the nodes (and links) in two dimensions. In the following subsections we describe each step.

## 3.1 Generative Model for Graphs

We assume that the links have been generated from a latent variable model, where the latent variables capture what is central in the graph, and the rest is noise. We should, as usual in modeling, build our assumptions about the data into the model, and those assumptions will then determine what kinds of properties of the graphs the visualization will focus on.

A convenient, flexible choice is SSN-LDA (Simple Social Network Latent Dirichlet Allocation) [30], a generative topic-type model for graphs. In SSN-LDA each node is associated with a membership vector over a set of latent components. Each component is in turn associated with a distribution over the nodes in the graph. Edges are generated by first drawing a component for the starting node, and then drawing the receiving node from the component-specific distribution. The assumption behind this generative process is that the graph can be decomposed into overlapping latent components, that is, groups of nodes with similar edge distributions. Hence we assume that the components are more important for graph visualization than are details in link patterns.

For SSN-LDA the latent space takes the form of component probabilities given the node, and thus the distances between link distributions should be evaluated in terms of those probabilities. We will use the quickly computable Hellinger distance which has been proven useful for topic models earlier [1],

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (\sqrt{p_i} - \sqrt{q_i})^2}, \quad (3)$$

where $p$ and $q$ are the probability distributions over the components. The distances could alternatively be computed in the link space (that is, based on the link distributions from each node rather than the higher-level component distributions), using information-geometric formulations.

SSN-LDA has two hyperparameters, $\alpha$ and $\beta$, controlling the node-wise distributions over the components and the component-wise distributions over the nodes, respectively. We learn these parameters from the data, by putting vague gamma priors for both parameters and sampling their values from their posteriors. Our sampling approach for the hyperparameters consists of first finding the maximum of the

posterior with Newton steps, then using Metropolis moves with a Gaussian proposal distribution.

## 3.2 Nonlinear Dimensionality Reduction for Visualization

The final step is to position the nodes on the display so that nearby nodes will have similar link distributions. This is a task of nonlinear dimensionality reduction (multidimensional scaling) for visualization.

We have recently introduced a rigorous formalization of nonlinear dimensionality reduction for *visualization as an information retrieval task* [27, 28], which solves the problem that such visualization has typically had no well-defined goal. Here the goal of visualization is to produce a low-dimensional display from which the analyst can retrieve which points are neighbors. Since a two-dimensional display cannot perfectly represent a high-dimensional data set, all methods make errors: some original neighbors are missed in the visualization, and some points are falsely retrieved as neighbors. The total cost of these errors is equivalent to the tradeoff between *precision* and *recall* of the information retrieval; the tradeoff is determined by the relative cost of a miss versus a false neighbor, as set by the analyst. Both precision and recall are valid goals for visualization.

The information retrieval criterion is a rigorous objective for visualization; we have introduced the Neighbor Retrieval Visualizer (NeRV) method [27, 28] which optimizes this criterion to produce optimal visualizations for information retrieval. In addition to being based on a rigorous formalism, NeRV has performed very well in quantitative comparisons [28], so it is a good choice for visualizing the graphs here. In this paper we use the variant of NeRV introduced in [28] which uses t-distributions to define the neighborhoods in the output space; in [28] this variant was called t-NeRV but here we simply call it NeRV.

In NeRV, the tradeoff between optimizing precision and recall is controlled by a parameter $\lambda \in [0, 1]$, defining the relative cost to the analyst of a miss versus a false neighbor. This parameter is set according to the needs of the analyst; both $\lambda = 0$ (maximizing precision; minimizing false neighbors) and $\lambda = 1$ (maximizing recall; minimizing misses) are useful goals. When the tradeoff parameter is set to $\lambda = 1$ (maximizing recall), NeRV with the t-distributions corresponds to the recent "t-Distributed Stochastic Neighbor Embedding" method (t-SNE; [25]); more generally, NeRV yields a flexible tradeoff between precision and recall. In experiments, we will use both $\lambda = 1$ (maximizing recall) and $\lambda = 0.1$ (maximizing mostly precision).

# 4. EXPERIMENTS

We next show empirically how our proposed graph visualization method can successfully find both assortative and disassortative structures, whereas existing methods fail in the latter task. We also show this quantitatively based on external ground truth.

We compare our method against three existing graph drawing methods discussed in Section 2: Walshaw's multi-level force-based algorithm [29]; Kruskal and Seery's spectral method [14], later independently rediscovered by Civril et al. as SDE [2]; and Noack's Edge-Repulsion Linlog [20]. Walshaw's algorithm is implemented as a Cytoscape [23] plugin [22]. Ali Civril kindly provided his original implementation for SDE. For ERLinLog, we used Noack's publicly available implementation [19].

## 4.1 Datasets

### *The Football graph.*

We first test the methods on a graph representing football teams and their games, with 115 nodes and 613 edges. Each node represents a college football team in the United States, and an edge between two teams implies that the teams played each other during the Division I games of the 2000 season [6]. Each team is known to belong to one of 12 conferences. The conference structure is highly assortative: The teams in each conference played heavily against each other. In addition, there is some structure in games between the conferences which is not as obvious.

### *Word-adjacency graphs.*

To verify the ability of our proposed method to find disassortative structure from graphs we apply it on *word adjacency graphs*. These graphs represent the relationships of words within text: A link is assigned between two directly adjacent words in a text. Word-adjacency graphs have been shown to exhibit disassortative structure [17]: Words from the same word classes (*e.g.*, adjectives, nouns), tend to appear next to words from different classes more often that words from the same class.

As a simple demonstration we use the adjacency network of common adjectives and nouns in the novel *David Copperfield* by Charles Dickens [18], which we call the Adjective-noun graph. It has 112 words as nodes and 424 edges between them.

We also constructed a larger word-adjacency graph based on seven novels by Jane Austen (*Emma*, *Lady Susan*, *Mansfield Park*, *Northanger Abbey*, *Persuasion*, *Pride and Prejudice*, and *Sense and Sensibility*). The novels are obtained from the Project Gutenberg website [21]. The WordNet project [16] offers a set of commonly used adjectives, nouns, and verbs, called *Core WordNet*, which we use here as the nodes of the graph. The graph was constructed by assigning a directed link between two Core WordNet words if they appear next to each other in the text of the Jane Austen novels at least two times. The graph was then binarized (so that multiple links between the same two words are counted only once), self links were removed, and the largest connected component was taken, resulting in a binary directed graph with 879 nodes and 2284 links between them. This graph is later denoted as the Jane Austen graph. The word classes (adjective, noun, or verb) are available for each graph node (Core WordNet word); note that some words may belong to multiple word classes, for example, 'open' can be used both as an adjective and as a verb.

## 4.2 Experiment Setup

We compare our proposed method to three existing graph layout methods, representing the three main approaches to graph visualization: Walshaw, SDE, and LinLog (described in Section 2). All comparison methods were run with their default settings, except that for LinLog the number of iterations was raised to 1000 to assure convergence.

We optimize SSN-LDA with a collapsed Gibbs sampler. We first run 10,000 burn-in iterations and then take 50 samples with 50 iteration interval. We also set the number of components beforehand, estimated based on external data

(and in the small demonstrations simply by setting it manually). In case such external data is not available, there are a couple of other means; a natural choice for a generative model would be to evaluate the predictive likelihood of left-out data for different numbers of components, but in the case of graphs this is not straightforward, as the data points are interdependent. Another possibility is to use a Hierarchical Dirichlet Process (HDP) prior [24] for the number of components, but due to the strick-breaking construction HDP has the unwanted property of producing only few large components and a long tail of tiny ones.

For the Football graph the number of components was simply set to 12, to match to the number of ground truth classes. For the Adjective-noun graph the number of ground truth classes is two, but setting the number of components to two would result in essentially one-dimensional visualization (each node would be positioned between the two components). We thus double the number of components to four.

In the Jane Austen graph we used external data, the known classes. We ran LDA with several numbers of components, ranging from 3 to 30, and evaluated the resulting Hellinger distance matrices by computing the k nearest neighbors classification performance with respect to the ground truth classes, with k set to 5. The best performance was obtained with 5 components.

For the first demonstrations, Adjective-noun and Football graphs, the NeRV parameter $\lambda$ was simply set to 0.1. In the Jane Austen graph we demonstrate the tradeoff between maximizing precision and recall by using both $\lambda = 0.1$ and $\lambda = 1.0$, respectively.

In addition, we also need to set the neighborhood size for NeRV. Here we follow the rule-of-thumb that the number of components times neighborhood size should roughly match the number of nodes in the graph. This way the optimized neighbors would represent the latent components found by the model, and which the user is interested in. Finally, we ran each NeRV ten times and chose the best run according to the NeRV cost function.

## 4.3 Results

*Graph visualizations.* All graph layouts were visualized with Cytoscape [23]. From the Football graph visualizations (Fig. 1) we see that LinLog and our method are able to detect clear clusters, most of which match exactly to the known conferences of the teams. This was expected, as LinLog is designed to find exactly this kind of assortative cluster structure from graphs, and our method is designed to be able to represent both assortative and disassortative structure. We note that some teams do not follow the pattern and are displaced from their conference members. Walshaw and SDE do not show clear clusters, but also in their layouts the nodes with same colors are somewhat grouped together, in Walshaw more clearly than in SDE.

In the case of the Adjective-noun graph (Fig. 2) the ability of our method to detect disassortative structures becomes evident. While the three other methods fail to find any structure and completely mix the adjectives and nouns, our method is able to separate the classes well into four clusters. Similar behavior can be seen in the Jane Austen graphs (Fig. 3, subfigures **A**-**E**): again, our method can separate the word classes (here adjectives, nouns, and verbs) into several clusters while the others do not find the structure. The other methods do show slight separation of verbs from others, but even this faint structure would be practically impossible to see without plotting the ground truth class colors onto the layout. In contrast, our method finds a very clear grouping of the nodes (words) and bundling of the links.

Subfigures **D** and **E** in Figure 3 demonstrate the interesting visual tradeoff between maximizing precision and recall. In both visualizations the word classes are separated nicely, but in the precision end the clusters are more distinct from each other. This highlights the formation of edge bundles between certain node clusters, which allow the analyst to discover the underlying linking patterns in the graph. On the other hand, the other end of the tradeoff (maximizing recall) yields a layout with less separation between nodes having moderate similarity. Both graph layouts are useful but for different needs of the analyst.

*Quantitative validation.* To verify that the visual inspection gave the right impression about how well the word classes are separated, we perform a simple quantitative evaluation of visualization quality, by evaluating class purity on the display with respect to the word classes of the nodes: we perform a leave-one-out classification of the nodes on the display, using $k$-nearest neighbor (KNN) classification with $k = 5$. In case a word (node) to be classified belongs to multiple word classes, a neighbor sharing any of those classes is counted as a positive outcome for the classification.

Subfigure **F** in Figure 3 shows KNN classification results for the Jane Austen graph. It is clear that our proposed methods are able to detect the word classes from the graph significantly better than the other three methods. Note that although the number of components for our method was chosen using the classes, the effect of the number of components on classification performance (about 3%) was small compared to the difference to other methods; thus the advantage of our method did not depend on that choice. The classification accuracy difference between the precision and recall ends of the tradeoff in our method is minimal compared to the difference to other methods, showing that both ends of the tradeoff can capture essential properties of the graph.

# 5. DISCUSSION

We have introduced a new principle for graph visualization, and shown how it can be taken into use in a method which outperforms existing graph visualization algorithms in discovering and visualizing global structures from complex graphs. What is particularly attractive is that the method is model-based. A generative model of the graph is assumed, and the visualization focuses on those properties in the link distribution the generative model models well, that is, considers important. The visualization can be made to focus on different properties of the graph by changing the model. In effect, the principle turns graph visualization, for which only mostly heuristic solutions have existed so far, into a generative modeling problem. With our method it is also possible to control the unavoidable tradeoff between precision and recall in the visualization, depending on the particular needs of the analyst.

The obvious disadvantage is longer running time, but the computation of the graphs in this paper only took some tens of minutes on a standard PC for all of the algorithms.

## 6. ACKNOWLEDGMENTS

The authors belong to AIRC. Ju.P. and K.N. are funded by the HeCSE and FICS graduate schools respectively and Ja.P. is supported by the Academy of Finland, decision number 123983. This work was also supported in part by the PASCAL2 Network of Excellence, ICT 216886.

## 7. REFERENCES

[1] D. Blei and J. Lafferty. A correlated topic model of science. *Annals of Applied Statistics*, 1(1):17–35, 2007.

[2] A. Civril, M. Magdon-ismail, and E. Bocek-rivele. Sde: Graph drawing using spectral distance embedding. In *The Proceedings of the 13th International Symposium on Graph Drawing*, pages 512–513. Springer, 2005.

[3] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1277–1284, 2008.

[4] A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs. In *GD '94: Proceedings of the DIMACS International Workshop on Graph Drawing*, pages 388–403. Springer, 1995.

[5] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software — Practice and Experience*, 21(11):1129–1164, 1991.

[6] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences USA*, 99(12):7821–7826, 2002.

[7] S. Hachul and M. Juenger. Large-graph layout algorithms at work: An experimental study. *Journal of Graph Algorithms and Applications*, 11(2):345–369, 2007.

[8] R. Hadany and D. Harel. A multi-scale algorithm for drawing graphs nicely. In *WG '99: Proceedings of the 25th Int. Workshop on Graph-Theoretic Concepts in Compute Science*, pages 262–277. Springer, 1999.

[9] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, November 1970.

[10] I. Herman, I. C. Society, G. Melancon, and M. S. Marshall. Graph visualization and navigation in information visualization: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 6:24–43, 2000.

[11] Y. Jia, J. Hoberock, M. Garland, and J. Hart. On the visualization of social and other scale-free networks. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1285–1292, 2008.

[12] Y. Koren. On spectral graph drawing. In *Proc. 9th Inter. Computing and Combinatorics Conference (COCOON'03), LNCS 2697*, pages 496–508. Springer-Verlag, 2002.

[13] Y. Koren, L. Carmel, and D. Harel. Ace: A fast multiscale eigenvectors computation for drawing huge graphs. Technical Report MCS01-17, The Weizmann Institute of Science, 2001.

[14] J. B. Kruskal and J. B. Seery. Designing network diagrams. In *Proc. First General Conference on Social Graphics*, pages 22–50, 1980.

[15] K. Lehmann and S. Kottler. Visualizing large and clustered networks. In *GD '07: Proceedings of the 15th International Symposium on Graph Drawing*, pages 240–251, London, UK, 2007. Springer-Verlag.

[16] G. A. Miller. "WordNet - About Us." WordNet. Princeton University (2009). `http://wordnet.princeton.edu`.

[17] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of Evolved and Designed Networks. *Science*, 303(5663):1538–1542, 2004.

[18] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[19] A. Noack. Linloglayout. `http://code.google.com/p/linloglayout/`.

[20] A. Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.

[21] Project Gutenberg. `http://www.gutenberg.org/wiki/Main_Page`.

[22] P. Salmela, O. S. Nevalainen, and T. Aittokallio. A multilevel graph layout algorithm for cytoscape bioinformatics software platform. Technical Report 861, Turku Centre for Computer Science, 2008.

[23] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, November 2003.

[24] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[25] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[26] F. van Ham and J. J. van Wijk. Interactive visualization of small world graphs. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization*, pages 199–206, Washington, DC, USA, 2004. IEEE Computer Society.

[27] J. Venna and S. Kaski. Nonlinear dimensionality reduction as information retrieval. In M. Meila and X. Shen, editors, *Proceedings of AISTATS*07, the 11th International Conference on Artificial Intelligence and Statistics (JMLR Workshop and Conference Proceedings Volume 2)*, pages 572–579, 2007.

[28] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11:451–490, 2010.

[29] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *GD '00: Proceedings of the 8th International Symposium on Graph Drawing*, pages 171–182, London, UK, 2001. Springer-Verlag.

[30] H. Zhang, B. Qiu, C. L. Giles, H. C. Foley, and J. Yen. An LDA-based community structure discovery approach for large-scale social networks. In *Intelligence and Security Informatics (ISI) 2007*, pages 200–207. IEEE, 2007.

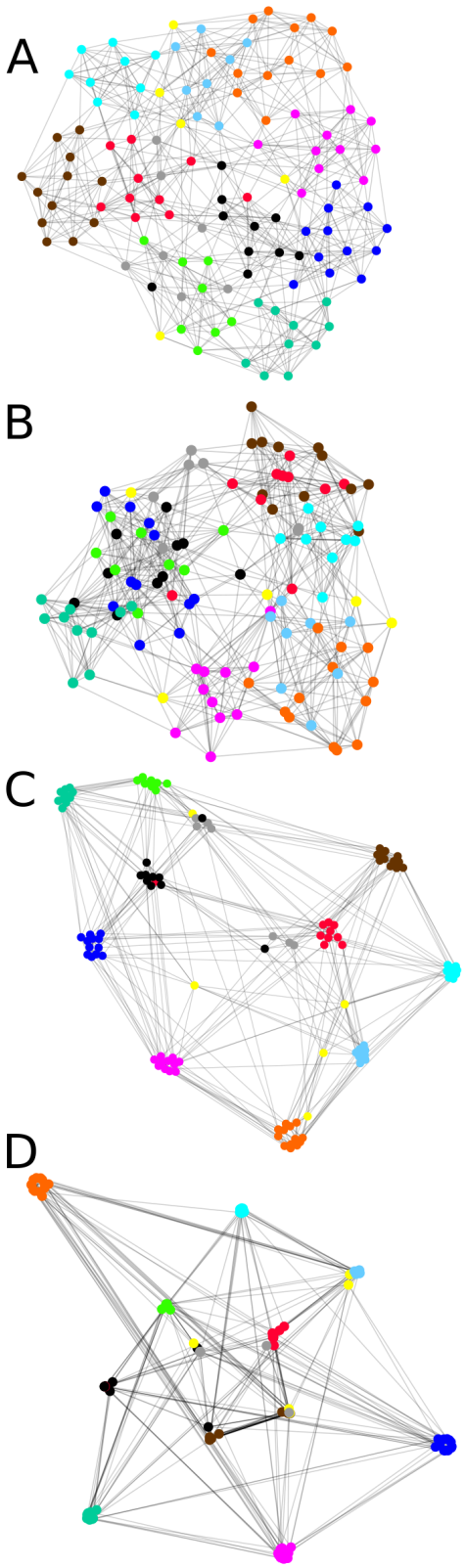



Figure 1: Football graph layouts. (A) Walshaw, (B) SDE, (C) LinLog, (D) Our method. Colors correspond to the 12 football conferences. The graph layouts by LinLog and our method reveal the conference structure the most clearly.

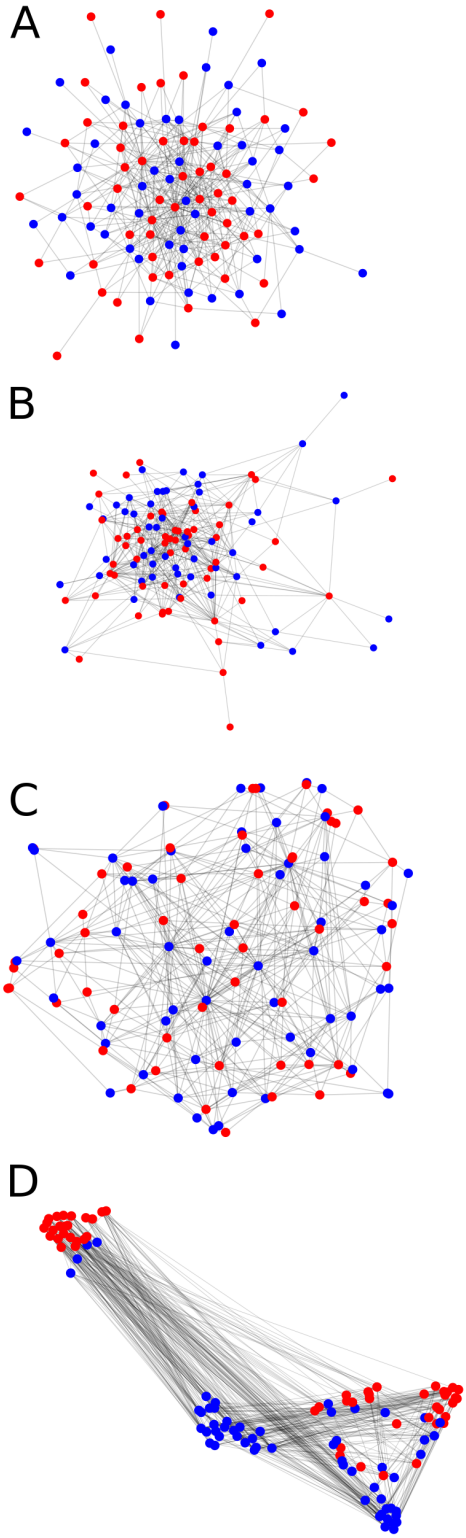
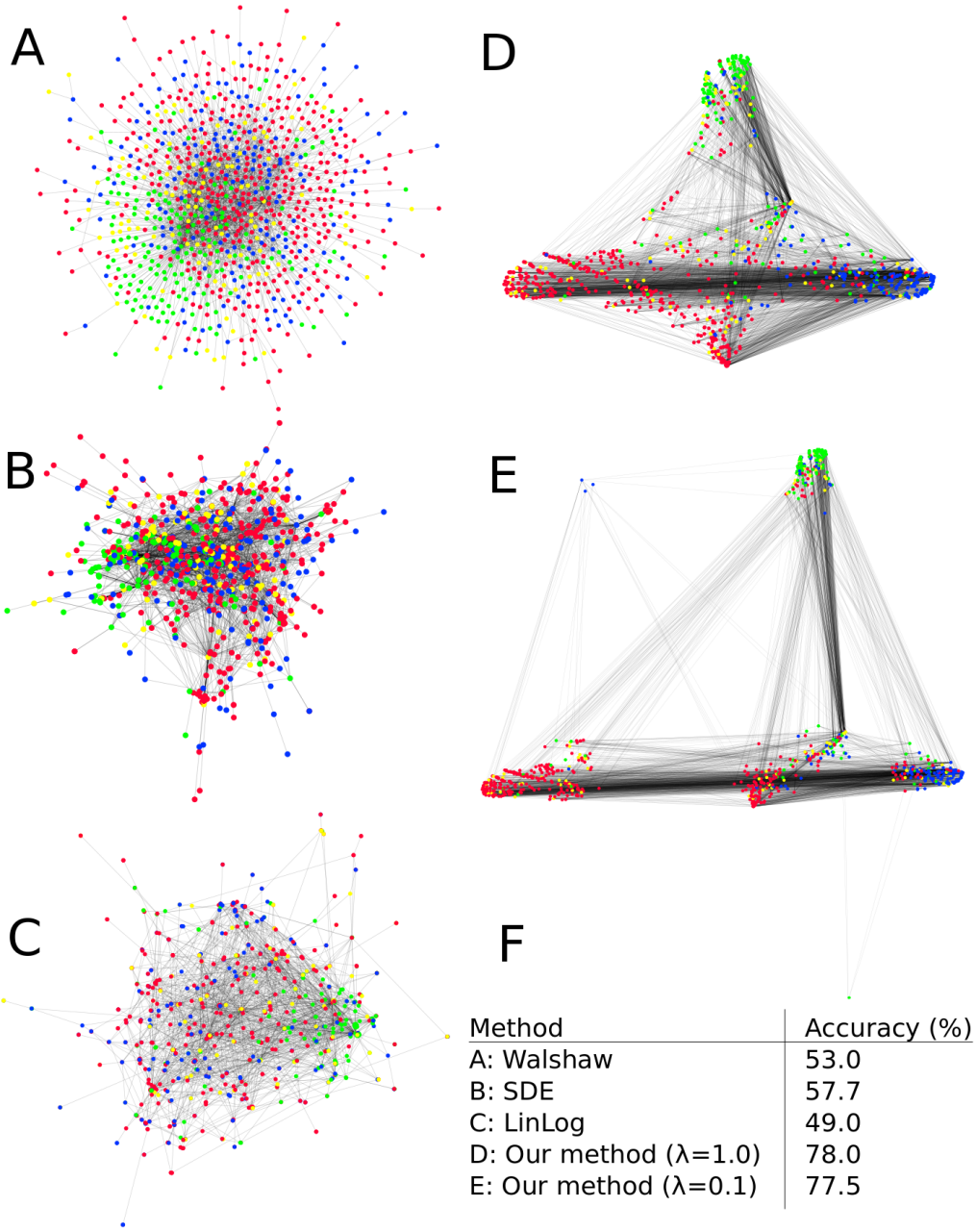



Figure 2: Adjective-noun graph layouts. (A) Walshaw, (B) SDE, (C) LinLog, (D) Our method. Colors: blue nodes are adjectives, red nodes are nouns. Only our method is able to reveal the structure where nouns mostly link to adjectives and vice versa.




| Method | Accuracy (%) |
|---|---|
| A: Walshaw | 53.0 |
| B: SDE | 57.7 |
| C: LinLog | 49.0 |
| D: Our method (λ=1.0) | 78.0 |
| E: Our method (λ=0.1) | 77.5 |

**Figure 3: Jane Austen graph layouts. (A) Walshaw, (B) SDE, (C) LinLog, (D) Our method, $\lambda = 1.0$ (maximizing recall), (E) Our method, $\lambda = 0.1$ (maximizing mostly precision). Blue nodes are adjectives, red nodes are nouns, green nodes are verbs, yellow nodes have multiple word classes. Our method again reveals the linking structure the most clearly. Subfigures D-E demonstrate the precision-recall tradeoff that the analyst can control in our method. Maximizing precision and recall yield different useful visualizations of the same structure: maximizing precision divides data into smaller, tighter groups and edges into tighter bundles, whereas maximizing recall yields less separation between nodes having moderate similarity. (E) Quantitative validation. The table shows the 5 nearest neighbour classification accuracy for each method, with respect to the known word classes. Our method performs best.**