Fast Overlapping Clustering of Networks Using Sampled Spectral Distance Embedding and GMMs

Malik Magdon-Ismail
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
magdon@cs.rpi.edu

Jonathan Purnell*
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
purnej@cs.rpi.edu

ABSTRACT

Clustering social networks is vital to understanding online interactions and influence. This task becomes more difficult when communities overlap, and when the social networks become extremely large. We present an efficient algorithm for constructing overlapping clusters, roughly linear in the size of the network. The algorithm first embeds the graph and then performs a metric clustering using a Gaussian Mixture Model (GMM). We evaluate the algorithm on the DBLP paper-paper network which consists of about 1 million nodes and over 30 million edges; we can cluster this network in under 20 minutes on a modest single CPU machine.

Keywords

Social Networks, Clustering, GMM

1. INTRODUCTION

Social, collaboration and interaction networks (Twitter, Facebook, DBLP, Citeseer, Amazon, etc.) are a source of vast, multimodal information. Many communities have taken advantage of such networks in order to communicate ideas broadly and quickly. The same technology has also allowed researchers to observe the dynamic nature of communities and how individuals work within them. One of the fundamental tasks is to identify the *overlapping* communities in a network, represented by its interactions – nodes abstractly represent the entities of interest (books, people, etc) and edges represent the interactions between nodes (e.g. an edge between two people if they purchased the same book). Clusters represent the collections of nodes which are "similar" and the same node may be in more than one cluster (a person may have interests in science fiction and philosophy).

As networks increase in size, clustering algorithms need to be scalable. Even quadratic time algorithms will soon be unacceptable. For our experiments, we use the DBLP paper-paper network, with about 1 million papers, with over 30 million edges. Since there is no accepted definition of what a cluster in a social network is, the typical approach has been to "define" a cluster by constructing an intuitive algorithm the result of which is a cluster. We take this approach, in that we present an intuitive algorithm with a focus on efficiency. The result of the algorithm is "our definition of a cluster" and the validation is whether, in real social networks, the algorithm can produce good results.

Overview of SSDE-Cluster The input is a (typically sparse) social network graph $G = (V, E)$ on n nodes (eg. books as nodes and (positive) edge similarities between books being the probability that a user would purchase both books conditioned on purchasing one). Typically, the definition of the node-similarities is application-specific, and is a determining factor for the quality of the results.

Our approach has two phases. The graph distances define a finite metric on n points. The first phase efficiently embeds this metric in $\mathbb{R}^d$, so that Euclidean distances reasonably approximate the graph metric. According to the Johnson-Lindenstrauss theorem, $d = O(\log n)$ dimensions essentially suffices for this (practically, a constant number of dimensions (say $d = 100$)). In our experiments, we find that 5-10 dimensions suffices. The challenge is to construct this embedding without visiting every entry in the distance matrix (that would make the algorithm quadratic); we do this by carefully sampling d rows of the distance matrix, which can be done in $O(nd)$, because the graph is sparse.

The advantage of first creating an approximate embedding is that now techniques can be used for clustering in metric spaces, for example k-means. We choose to use a Gaussian Mixture Model (GMM) because it can readily be adapted to give overlapping clusters (k-means delivers a partitioning). We choose the number of clusters by comparing to a random data set, and we determine the extent to which the clusters overlap by selecting a threshold to maximize the quality of the clusters.

1.1 Related Work

For text, a common approach is Latent Dirichlet Allocation (LDA)[4]; though DBLP is text data on which LDA could be applied, our method is generally applicable to any weighted graph. We use the DBLP because the relations between authors and papers have many of the properties found in social networks. Nevertheless, we use text-based metrics to concretely validate our results.

Finding communities in social networks has been rapidly researched during the past decade [13, 15, 14, 21]. Early on researchers worked

*www.cs.rpi.edu/~purnej/

with the 'small world', power-law, and network transitivity properties. These properties characterized communities of isolated groups where a small number of objects in a community were connected to objects in other communities. A more recently proposed property is the connectedness of communities, where objects within the community have shorter path lengths than to objects outside the community [11]; building on such definitions, one should expect communities to be densely connected subgroups of the graph [19, 20].

A drawback of the aforementioned approaches is that they are graph partitioning methods – the communities cannot overlap. In reality, communities often overlap and sometimes in non-trivial ways. To address efficiency and memory footprint issues, work has progressed by extending algorithms to work within subspaces[25], identifying overlapping groups using local optimality [2, 3], and removing the constraints on the number of communities[1]. Methods for finding overlapping communities by searching for clique-like structures also exist [23, 30, 29].

In this paper we propose an algorithm that can handle large datasets, particularly those with a high number of objects; we also allow overlap. The work in [22, 28] is similar to ours in that the first step in our algorithm is spectral, similar to multi-dimensional scaling (MDS); their algorithm is based on spectral properties of the Laplacian of a graph which they construct from a metric embedding; in some sense, they do the reverse of what we do. However, construction of the Laplacian is $\Omega(n^2)$. While we use GMMs as our tool for "soft" metric clustering, any method for metric clustering which readily gives overlapping clusters could be applied. GMMs happen to be an area of very active research (see for example [8, 9, 24, 27]). We use a fast approximation to MDS [7] based on an online sampling of the squared distance matrix. The Nyström method is similar to MDS, and has been used for spectral segmentation (eg. [**?**]). Sparse approximations to matrices in machine learning, typically to approximate the Gram matrix in kernel methods have been considered ([26, 10]); in all cases, the methods sample the whole matrix ($\Omega(n^2)$). When it comes to graph partitioning, there is much work, and we suggest [17] for a good overview; a comprehensive review is given in [13], focusing mostly on partitioning, but also considering overlapping clusters.

2. THE SSDE-CLUSTER ALGORITHM

The input is a weighted graph, $G = (V, E)$. The algorithm can be broken down as follows.

1. Run SSDE (spectral embedding) to approximately embed the graph in $d \ll n$ dimensions.
2. Run a GMM algorithm to compute "soft clusters"; determine K, the number of clusters, by comparing the marginal value of adding a cluster on the real data with random data.
3. Use the GMM probabilities to construct overlapping clusters; determine the degree of cluster overlap by locally optimizing a cluster density.

2.1 Sampled Spectral Distance Embedding

Sampled Spectral Distance Embedding (SSDE) is an approximation to classical multidimensional scaling which was introduced in the context of fast graph drawing [5]. The distance matrix $\mathbf{D}$ is the symmetric $n \times n$ matrix containing all the pair-wise distances. Suppose we position vertex v_i at $\mathbf{x}_i \in \mathbb{R}^d$. We are seeking a positioning that approximates the graph theoretical distances with the Euclidean distances, i.e.,

$$||\mathbf{x}_i - \mathbf{x}_j|| \approx D_{ij}, i, j = 1, 2, \ldots, n \tag{1}$$

After squaring and some manipulation (see [5]), one obtains the MDS equation:

$$\mathbf{Y}\mathbf{Y}^T \approx -\frac{1}{2}\gamma\mathbf{L}\gamma = \mathbf{M} \tag{2}$$

where the embedding $\mathbf{Y}$ is an $n \times d$ matrix containing the coordinates of the points, $L_{ij} = D_{ij}^2$ (the squares of the distances) and the centering projection matrix $\gamma = \mathbf{I}_n - \frac{1}{n}1_n 1_n^T$. The spectral decomposition of $\mathbf{M}$ then gives $\mathbf{Y} = [\sqrt{\lambda_1}\mathbf{u}_1, \ldots, \sqrt{\lambda_d}\mathbf{u}_d]$, where $\lambda_1, \ldots, \lambda_d$ are the top d eigenvalues of $\mathbf{M}$ and $\mathbf{u}_1, \ldots, \mathbf{u}_d$ are the associated eigenvectors. One useful property of MDS is that when the distance matrix is nearly embedable in d dimensions, then MDS recovers such an approximate embedding (see [6, Theorem 3]). The drawback of MDS is that it is slow, $\Omega(n^2 d)$. The next two lemmas give the insight for speeding up MDS.

LEMMA 1 ([16]). *Any n-point finite metric can be embedded (to within ϵ) in $\mathbb{R}^d$ for $d = O(\ln n/\epsilon^2)$.*

This lemma says that from the point of view of distances, we can approximately treat the graph as n points in a (small) d-dimensional space. For practical purposes, $d \leq 100$ suffices.

LEMMA 2. *For any n points in d-dimensions, let $L_{ij} = \| \mathbf{x}_i - \mathbf{x}_j \|^2$. Then rank$(\mathbf{L}) \leq d + 2$.*

Note that the rank does not depend on n. Combining these two lemmas: any n-point finite metric is approximately embedable in $d = O(\ln n/\epsilon^2)$ dimensions, which means that the "numerical" rank of $\mathbf{L}$ is roughly $d + 2 = O(\ln n/\epsilon^2)$. Practically, this means that a small number (say 100) suitably chosen rows of $\mathbf{L}$ captures all the information in $\mathbf{L}$. SSDE works in three steps (see [5]).

1. Sample a subset $c \approx 100$ columns to represent L, we denote this by $\mathbf{C}$. These columns correspond to the nodes $v_{i_1}, \ldots, v_{i_c}$. We use the standard greedy 2-approximation to the c-center problem to select the nodes. This serves to "spread" out the nodes and give them large "numerical" rank. The first node is selected arbitrarily to compute the first column (SSSP); each subsequent node selected is farthest from the current set of selected nodes.
2. Construct a low rank approximation to $\mathbf{L}$, $\hat{\mathbf{L}} = \mathbf{C}\phi^+\mathbf{C}^{\mathrm{T}}$., where ϕ is the $c \times c$ matrix of the intersection of $\mathbf{C}$ and $\mathbf{C}^{\mathrm{T}}$. Some care may be needed in computing ϕ^+ in a stable way (see [5]).
3. Construct the embedding $\mathbf{Y}$ from the spectral decomposition of $-\frac{1}{2}\gamma\hat{\mathbf{L}}\gamma$.

We briefly comment of the running time of the SSDE phase. The first step involves c SSSP computations which can be performed in $O(c|E|\log|V|) = O(cn\log n)$. Since $\gamma = \mathbf{I}_n - \frac{1}{n}1_n 1_n^{\mathrm{T}}$, $\gamma\mathbf{C}$ can be computed in $O(nc)$ time. Let $\mathbf{Q} = \gamma\mathbf{C}$, and construct the singular value decompositions

$$\mathbf{Q} = \mathbf{U}_{\mathbf{Q}}\Sigma_{\mathbf{Q}}\mathbf{V}_{\mathbf{Q}}^{\mathrm{T}}, \qquad \phi = \mathbf{U}_\phi\Sigma_\phi\mathbf{V}_\phi^{\mathrm{T}},$$

where $\mathbf{U_Q}$ is $n \times c$, and all the other matrices are $c \times c$; this takes $O(nd^2 + d^3)$ time. Then,

$$-\frac{1}{2}\gamma\hat{\mathbf{L}}\gamma = -\frac{1}{2}\mathbf{U_Q}\Sigma_\mathbf{Q}\mathbf{V}_\mathbf{Q}^\mathrm{T}\mathbf{U}_\phi\Sigma_\phi^+\mathbf{V}_\phi^\mathrm{T}\mathbf{V_Q}\Sigma_\mathbf{Q}\mathbf{U}_\mathbf{Q}^\mathrm{T}.$$

The matrix product $\Sigma_\mathbf{Q}\mathbf{V}_\mathbf{Q}^\mathrm{T}\mathbf{U}_\phi\Sigma_\phi^+\mathbf{V}_\phi^\mathrm{T}\mathbf{V_Q}\Sigma_\mathbf{Q}$ and its eigen-decomposition $\mathbf{U}\Sigma\mathbf{U}^\mathrm{T}$ can be computed in $O(c^3)$ time; hence, the spectral decomposition $-\frac{1}{2}\gamma\hat{\mathbf{L}}\gamma = -\frac{1}{2}\mathbf{U_Q}\mathbf{U}\Sigma\mathbf{U}^\mathrm{T}\mathbf{U}_\mathbf{Q}^\mathrm{T}$ can be computed in $O(nc^2 + c^3)$. The embedding is $\mathbf{Y} = \mathbf{U_Q}\mathbf{U}(-\frac{1}{2}\Sigma)^{1/2}$. In practice, we may only need (say) the top 5 dimensions, and so a power iteration could be used in lieu of SVD (as was done in [5]).

2.2 Soft Clustering with a Gaussian Mixture Model (GMM)

After embedding the graph, our next task is to cluster the nodes. Many metric clustering methods exist, and we choose a standard GMM trained using the E-M algorithm. One advantage of the GMM over (say) straight K-means is that the cluster probabilities will be useful in constructing overlapping clusters. A GMM is a weighted sum of M component densities $p_1(\mathbf{x}), \ldots, p_M(\mathbf{x})$; each component density is a d variate Gaussian function with mean $\boldsymbol{\mu}_k$ and covariance matrix Σ_k:

$$p_k(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\mathrm{T}\Sigma_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)}.$$

The GMM is $p(\mathbf{x}) = \sum_{k=1}^M \pi_k p_k(\mathbf{x})$, where the mixture weights satisfy $\sum_{k=1}^M \pi_k = 1$. The model parameters are

$$\theta = \{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1,\ldots,M}. \tag{3}$$

We use a simple E-M procedure for training. We initialize the GMM by selecting $\boldsymbol{\mu}_k$ randomly without replacement, and setting $\pi_k = 1/M$ and $\Sigma_k = \mathbf{I}_d$. One might also consider a better initialization of the $\boldsymbol{\mu}_k$ (eg. using the greedy 2-approximation to the M-center problem, or more sophisticated methods as mentioned in the related work). In the expectation step, the probabilities for node i to belong to cluster k are calculated:

$$p_{ki} = \pi_k p_k(\mathbf{x}_i)$$

We then update the parameters using these probabilities during the maximization step:

$$\boldsymbol{\mu}_k \leftarrow \frac{\sum_i p_{ki}\mathbf{x}_i}{\sum_i p_{ki}}$$

$$\Sigma_k \leftarrow \frac{\sum_i p_{ki}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\mathrm{T}}{\sum_i p_{ki}}$$

$$\pi_k \leftarrow \frac{1}{n}\sum_i p_{ki}.$$

One iteration of E-M is $O(nMd^2)$; for small M and d this is acceptable. [18] discusses methods for reducing this to $O(nMd)$ using low rank perturbations to the covariance matrix. The output of the GMM phase is the set of probabilities $\{p_{ki}\}$, which will be used in computing the discrete *overlapping* clustering. For some applications, the probabilities themselves would suffice.

2.3 Determining the Number of Clusters

We describe a simple heuristic for determining the number of clusters by comparing the real data clusters to random data. Too many clusters results in over-fitting the data and breaking the true communities into several smaller communities.

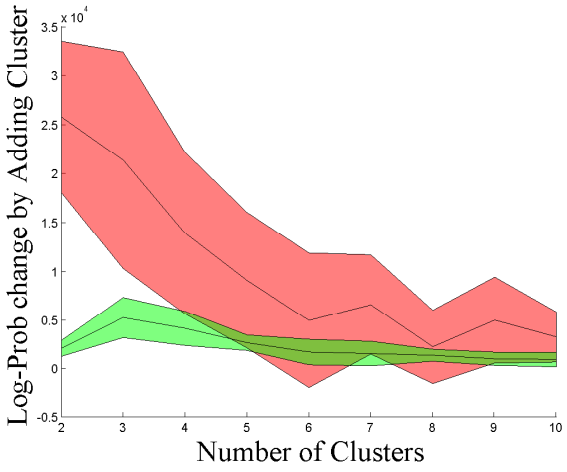


Figure 1: Setting # clusters

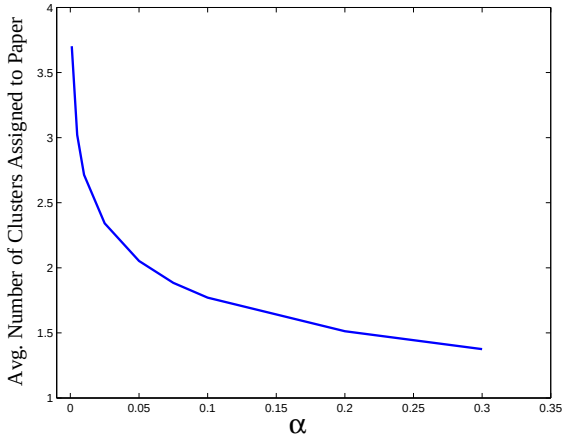


Figure 2: α vs. overlap.

We develop a random data set uniformly over the same space as the embedded data. Since there are no clusters in the random data, the increase in log-likelihood by adding an additional cluster is purely from over-fitting. When the benefit from adding a cluster to the real data is not significantly more than that of adding a cluster to random data, we argue that it is time to stop adding clusters. The figure to the right shows the successive gains of adding a cluster to the DBLP data versus to the random data; from Figure 1, we discern that about 7-11 clusters is appropriate (we used 7 in our experiments).

2.4 Determining the Extent of the Overlap

Every node i is assigned to its most likely cluster, $\mathrm{argmax}_k\, p_{ki}$. This constructs a partition of the nodes into clusters. We now extend this so that clusters may overlap. For node i, define $\alpha_{ki} = p_{ki}/\max_k p_{ki}$; α_{ki} measures how diffuse node i's membership is in its most likely cluster. Assume we have a cluster metric which measures the quality of a cluster. We use

$$E(C) = \lambda\frac{W_{\text{in}}(C)}{W_{\text{in}}(C) + W_{\text{out}}(C)} + (1-\lambda)\frac{W_{\text{in}}(C)}{|C|(|C|-1)};$$

W_{in} (resp. W_{out}) is the sum of similarities within the cluster (resp. from within to outside); $E(C)$ combines similarity internally and

to the outside with the average internal similarity. We used $\lambda = \frac{1}{2}$.

We use α_{ki} to define an ordering over node-cluster pairs, starting with high α. We propose two approaches to to constructing the overlaps. The first is to process the nodes according to the ordering, each time adding the node to its corresponding cluster if the metric for that cluster increases. This is similar in spirit to the local optimizations performed in [2, 3]. An alternative is to track how the average cluster metric varies as we progress along this ordering. When the average cluster metric starts to decrease noticeably, it defines an alpha threshold α^*. For all $\alpha_{ki} \geq \alpha^*$, we add the corresponding node i to cluster C_k. In our experiments, we set $\alpha^* = 0.3$ which gives about 1.2 clusters per paper – this average number of clusters per paper could also be a user input to compute α^*.

3. THE DBLP NETWORK

In general, the social network is constructed by defining the "agents" (nodes) and the interactions or relationships between them (communications, similarities, etc.). We apply our algorithm to the Digital Bibliography and Library Project (DBLP) data [**?**]; we choose papers as nodes, and two papers are related if they have common authors. The underlying assumption is that papers having similar authors are more likely to have similar content, which is reasonable, since most authors tend to have a focus area. We use the Jaccard index of the author sets to define similarity; for two papers i, j, let A_i, A_j be their respective author sets. Then

$$s_{ij} = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}; \qquad d_{ij} = \frac{1}{s_{ij}}$$

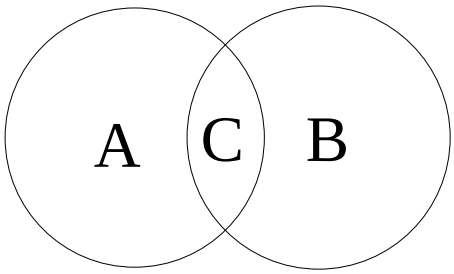
Most clustering algorithms for social networks work with the similarities $\{s_{ij}\}$ and try to optimize some measure of intra-cluster similarity versus inter-cluster similarity. Since our algorithms are metric based, we need a measure of difference, so we simply use the inverse-similarity; this allows us to construct our finite metric.

3.1 Validating Clusters

The recurring problem with applying a clustering algorithm to real data (where the "definition" of the cluster is the "result of the algorithm") is to validate these clusters as good. We use human judgement based on the title and venue information of the papers. We preprocess title texts by removing stop words and stemming [**?**]. For a cluster of papers C_k (remember, clusters can overlap), we construct a word probability distribution $h_k(w)$. Similarly, we can construct a background distribution $h(w)$ for the entire data set of titles. The words w for which $h_k(w) \gg h(w)$ are descriptive of the cluster; we can also identify the words w for which $h_k(w) \ll h(w)$, which are indicative of not being in the cluster. This breaks down the papers into "topics" based on title texts. Visually, we depict the descriptive words using word clouds, and compare with an LDA analysis of the title data. Note that in general networks, such an LDA analysis is not possible. We perform a similar analysis for the paper venues, which should hopefully identify the conferences corresponding to the various topics.

4. EXPERIMENTAL RESULTS

For our study, we constructed the DBLP network as described above and clstered the giant componend, which consisted of about 900K papers (nodes) and over 30 million (weighted) edges. We chose $c = 25$ for the SSDE phase and then took an the top 5 eigenvectors for the embedding. We clustered using the GMM into 7 clusters (as discussed earlier) and used a threshold $\alpha^* = 0.3$ to construct the overlapping clusters (a paper is in approximately 1.2 clusters on average). The process of embedding and clustering took under an 20 min on a modest single CPU machine.



Cluster and Overlap Quality. After clustering into 7 clusters, a typical pair of clusters is illustrated to the right. With respect to the paper-paper similarity, we can measure the average inside a set (A and B) as compared with the average between a set and the intersection(A, C and B, C) and the average between sets (A, B). We expect the within set similarity to be larger than that between a set and an intersection in which the set participates, which in turn should be larger than the between set similarity. This is indeed found to be true for our overlapping clusters (see Table 1), which lends credence to their validity.

Since the clustering was done with paper-paper similarities based on authorship overlap, we may validate the clusters by looking at the text-based and conference based topics they represent (as discussed earlier). The 7 word clouds representing the clusters and the 3 largest intersections are shown in Figure 3. Our algorithm roughly breaks down the papers into 7 intuitive topics, with corresponding representative conferences shown in Figure 5.

Qualitatively, we see that the results are similar. One main difference between LDA and SSDE-cluster is that SSDE-cluster seems to have merged Web and DB into one cluster and produced a new topic which combines robotics and bioinformatics.

We illustrate the word clouds for the intersections for the three largest intersections in Figure 6. The robotics and bioinformatics cluster did not really have any significant intersections.

5. CONCLUSION

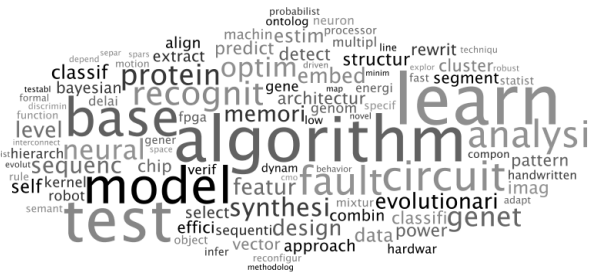
We present an algorithm to quickly find overlapping communities in very large social networks. Our algorithm involves two fast and linear-time phases (hiding a log factor in the SSSP task for weighted graphs). Further exploration of what distance matrices are captured well by the greedy node selection algorithm in SSDE, and further study on how to convert soft cluster probabilities into a discrete overlapping clustering woud be warranted. Also, our current algorithm puts every node into at least one cluster, and methods for outlier detection (i.e. allowing isolated nodes) could considerably help. In general, for efficient clustering on huge graphs, one cannot explore all the similarities between nodes $\Omega(n^2)$, and so such sampling based approximation approaches are likely to be the only feasible means of tackling such problems.

ACKNOWLEDGEMENTS. Research was sponsored by the Army Research Laboratory and was accomplished under Coopera-

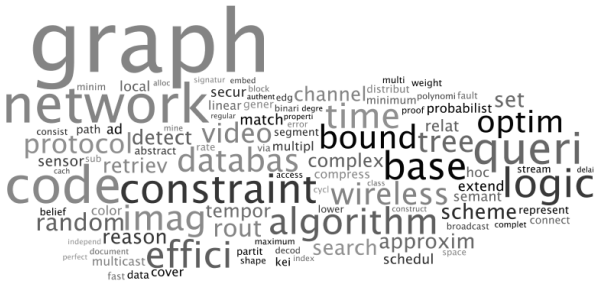
SysParSym

ImageTh

SysArch

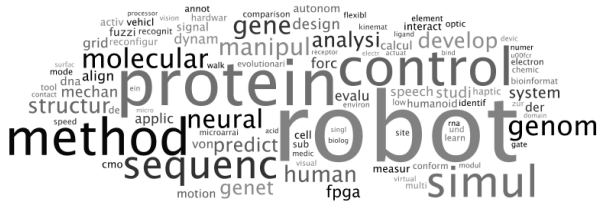


Learn



CombTh

WebDB



RoboBio

Figure 3: Keyword Clouds for the 7 clusters

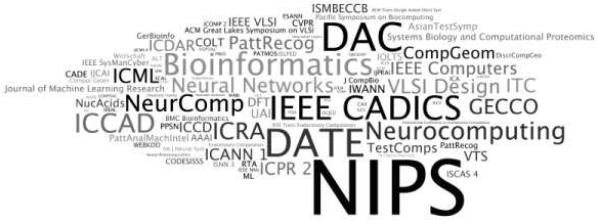
	SysParSim	ImageTh	SysArch	Learn	CombTh	WebDB	RoboBio
SysPS	**2.5**	2.4,1.7,1.3	2.4,1.7,1.3	2.4,1.7,1.3	2.4,1.6,1.3	2.5,1.4,1.0	2.4,1.7,1.2
ITh		**2.3**	2.4,1.8,1.5	2.4,1.7,1.3	2.4,1.7,1.5	2.4,1.6,1.3	2.3,1.7,1.5
SysA			**2.4**	2.4,1.7,1.5	2.4,1.8,1.5	2.4,1.8,1.4	2.4,1.8,1.5
Lrn				**2.4**	2.4,1.6,1.4	2.4,1.5,1.2	2.4,1.7,1.4
CTh					**2.4**	2.4,1.7,1.2	2.4,1.8,1.5
WbDB						**2.5**	2.4,1.9,1.4
RBio							**2.4**

Table 1: Cluster Quality

SysParSym

ImageTh

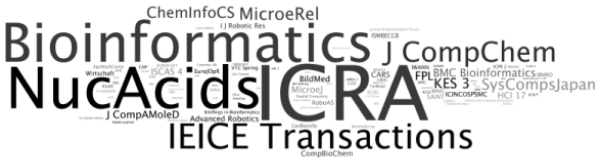
SysArch



Learn

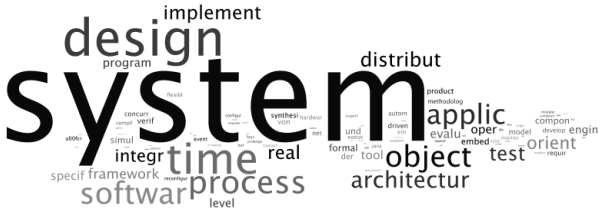
CombTh

WebDB

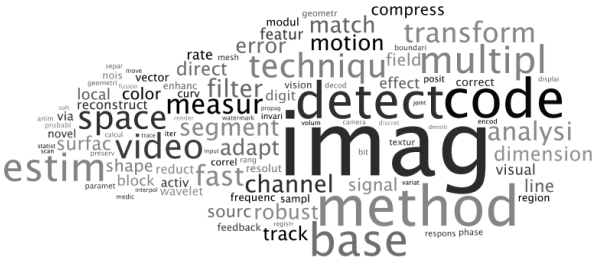


RoboBio

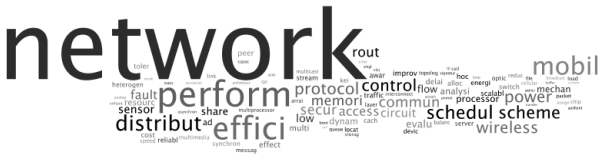
Figure 4: Conference Clouds for the 7 clusters



SysParSym



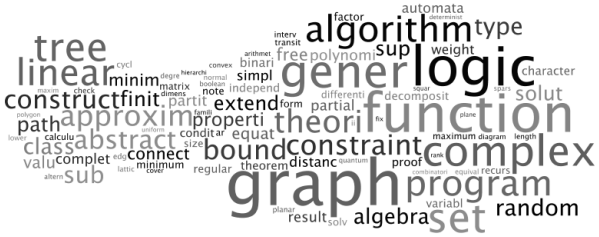
ImageTh



SysArch

Learn

CombTh



Web

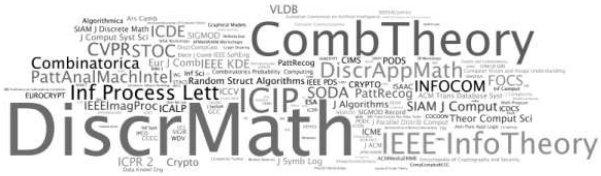
DB

Figure 5: LDA Keyword Clouds for 7 clusters

SysParSym ∩ SysArch

WebDB ∩ Learn

ImageTh ∩ CombTh



CombTh ∩ DiscrMath

Figure 6: Word/Venue Clouds for the three major intersections

tive Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

6. REFERENCES

[1] M. Al Hasan, Saeed Salem, Benjarath Pupacdi, and M.J. Zaki. Clustering with Lower Bound on Similarity. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings*, page 122. Springer Verlag, 2009.

[2] Jeffrey Baumes, Mark Goldberg, M. Krishnamoorthy, M. Magdon-Ismail, and N. Preston. Finding communities by clustering a graph into overlapping subgraphs. In *International Conference on Applied Computing (IADIS 2005)*, pages 97–104, 2005.

[3] Jeffrey Baumes, Mark Goldberg, and Malik Magdon-Ismail. Efficient Identification of overlapping communities. *IEEE Transactions on Neural Networks*, 1(2):19–328, March 2005.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, May 2003.

[5] A. Civril, M. Magdon-Ismail, and E. Bocek-Rivele. SSDE: Fast graph drawing using sampled spectral distance embedding. In *Graph Drawing*, pages 30–41. Springer, 2006.

[6] Ali Civril, Malik Magdon-Ismail, and Eli Bocek-Rivele. SDE: Graph drawing using spectral distance embedding. Technical Report 05-14, Rensselaer Polytechnic Institute, 2005.

[7] Trevor F. Cox and M.A.A. Cox. *Multidimensional Scaling*. Chapman Hall/CRC, 2nd edition, 2000.

[8] Sanjoy Dasgupta. Learning mixtures of gaussians. In *Proc. 40th Annual Symposium on Foundations of Computer Science*, page 634, 1999.

[9] Sanjoy Dasgupta and Leonard J. Schulman. A two-round variant of em for gaussian mixtures. In *Proc 16th Conference on Uncertainty in Artificial Intelligence*, pages 152–159, 2000.

[10] Petros Drineas and Michael W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.*, 6:2153–2175, 2005.

[11] Scott Epter, Mukkai Krishnamoorthy, and M. Zaki. Clusterability detection and cluster initialization. In *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications at the 2nd SIAM International Conference on Data Mining*, pages 47–58, 2002.

[12] Jonathan Feinberg. Wordle java applet, 2009.

[13] Santo Fortunato. Community detection in graphs. *Physics Reports 486, 75-174 (2010)*, 486:75–174, 2010.

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

[15] Eui-hong Sam Han. Clustering In A High-Dimensional Space Using Hypergraph Models. *Supercomputer*, 1994.

[16] W. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[17] R. Kannan, S. Vempala, and A. Veta. On clusterings-good, bad and spectral. In *Proc. 41st Annual Symposium on Foundations of Computer Science*, page 367, 2000.

[18] Malik Magdon-Ismail and Jonathan Purnell. Approximating the covariance matrix with low rank perturbations. In *Proc. 11th Int. Conf. on Intelligent Data Engineering and Automated Learning (IDEAL)"*, page to appear, Sept 2010.

[19] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):1–15, February 2004.

[20] M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B - Condensed Matter*, 38(2):321–330, March 2004.

[21] M.E.J. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[22] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[23] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–8, June 2005.

[24] Arora Sanjeev and Ravi Kannan. Learning mixtures of arbitrary gaussians. In *Proc 33rd annual ACM symposium on Theory of computing*, pages 247–257, 2001.

[25] Karlton Sequeira and Mohammed Zaki. SCHISM: a new approach to interesting subspace mining. *International Journal of Business Intelligence and Data Mining*, 1(2):137, 2005.

[26] Alex J. Smola and Bernhard Schökopf. Sparse greedy matrix approximation for machine learning. In *Proc 17th International Conference on Machine Learning*, pages 911–918, 2000.

[27] J. J. Verbeek, N. Vlassis, and B. Kröse. Efficient greedy learning of gaussian mixture models. *Neural Comput.*, 15(2):469–485, 2003.

[28] Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *Proc. 15th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2009.

[29] M Zaki, M Peters, I Assent, and T Seidl. Clicks: An effective algorithm for mining subspace clusters in categorical dataset. *Data & Knowledge Engineering*, 60(1):51–70, January 2007.

[30] M.J. Zaki and M. Peters. CLICKS: Mining Subspace Clusters in Categorical Data via K-Partite Maximal Cliques. *21st International Conference on Data Engineering (ICDE'05)*, pages 355–356, 2005.