Clustering with the DBLP Bibliography to Measure External Impact of a Computer Science Research Area

Laurel Orr and Jennifer Ortiz

1 Introduction

Digital bibliometric databases are rich sources of knowledge to understand the connections between authors and conferences. The DBLP Computer Science Bibliography is one of the main repositories for the computer science community. However, understanding and interpreting the bibliographic information is not intuitive and at times, controversial [14]. How should one determine a researcher's area? What can we learn from the interactions between authors in each area?

A natural method for answering these questions is to construct a network of authors and use community detection techniques to group authors into different research areas and analyze the areas. Organizing the data into a network graph rather than relational tables allows us to treat the areas as social groups and take advantage of graph analytics and visualizations, providing a unique insight and perspective to the data.

The goal of this project is to detect research communities from the DBLP bibliography in order to predict the various research areas an author contributes to. We evaluate different unsupervised clustering techniques by seeing how well they distinguish between research areas and place authors in their corresponding areas. We focus on unsupervised techniques because it may not be known apriori what field a researcher is in. The rest of this paper reviews related work, details our approach, and gives results of these clustering techniques.

2 Related Work

The problem of community detection in social networks has been studied extensively in computer science. Researchers have experimented with changing the structure of the network as well as the underlying algorithm and optimizations. Palla et. al. [9] create a bipartite network based on e-print archives where authors were nodes and edges were co-authorships. Palla also uses clique techniques to accurately represent real-world, overlapping groupings.

Zaïane et. al. [19] use a random walk technique to uncover communities through the DBLP data. They create a more complicated network by weighting the edges and creating nodes to represent authors' collaborations in each conference. Additionally, they extract topic words from paper titles to use in clustering. Similarly, Lin et. al. [15] also use title words in clustering, but instead, focus on clustering networks with incomplete data. They use distance metric learning in order to create separate clusters and evaluate the co-authorships of the DBLP data.

Gao et. al. [7] also explore the DBLP data by addressing the fact that different techniques may produce controversial results. Therefore, they propose a way to explore the data by combining and consolidating predictions from both supervised and unsupervised models.

Other authors focus on dynamic social network analysis which looks at how communities change over time such as Huang et. al. [11] and Babskova et. al. [4]. [11] creates a word association network from all the titles to evaluate the popularity of topics over time. On the other hand, [4]

looks at how the co-author network changes when taking a subset of papers relating to particular search terms.

Biryukov and Dong [6] specifically analyze computer science communities based on DBLP data. They form co-author communities and place authors in research areas based on which conferences they publish in. They evaluate different aspects of each area such as area relatedness, productivity, and age in order to gain a better understanding of the communities and conferences.

3 Approach

To compare different clustering algorithms, we need a "ground truth" for each author's research area. We are aware that an author's area is subjective, which is why we assign each author to a mixture of research areas.

3.1 Refining the Research Areas

Since the field of computer science is diverse, we only consider the 9 main research areas of AI/Robotics, Graphics/Vision, Hardware/Architecture, Databases, ML/Data Mining, Algorithms/Theory, HCI, Networks/Communication, and PL/Software Engineering. Similar to [6], we take the top conferences from each field by analyzing the rankings from [6], [5], [13], and [3].

Again, we are aware that these rankings are subjective and that the number of authors in each conference varies across research areas. Thus, to obtain a dataset that reflects the natural distribution of these areas, we focus on the top three conferences for each research area (see Table 1).

Area	Conferences
AI	AAAI, ICRA, IROS
DB	CIKM, ICDE, SIGMOD
GV	CVPR, ICCV, ICIP
HA	DAC, ICCAD, ISCA
HCI	CHI, CSCW, Mobile HCI
ML	ICPR, FSKD, KDD
NC	INFOCOM, IPSN, SIGCOM
PL	ICSE, JICSLP, OOPSLA
TH	FOCS, SODA, STOC

Table 1: Top conferences associated with each area

Once determining the top conferences and standardizing the abbreviations by cleaning up the dirty data (i.e. labelling "UbiCom" and "HUC Workshop", as "HUC"), we have a total of 136,736 authors and 126,142 articles.

Lastly, we find the "ground truth" of a researcher's area by calculating the percentage that author publishes in each research area, removing the areas that contribute less than 20 percent of publications, and recalculating the percentages. For example, if an author publishes a total of 30 articles in ICSE, 10 articles CVPR, and 1 article in CHI, we consider that author to be 75 percent in PL, 25 percent in GV, and 0 percent in HCI. We do this because if an author has only published a small percentage in one area, we do not want to classify her as being proficient in that area.

We specifically chose 20 percent to be the cut-off point by evaluating the number of areas authors publish in throughout different thresholds. We found that using a threshold greater than 20 percent left some authors without a research area, while using a lower threshold led authors to publish in as many as 7 areas. An author having a label of many areas seems to be an unrealistic description of an author.

3.2 Clustering Methods Overview

There are two types of unsupervised clustering we use: hard clustering and soft clustering. Hard clustering means an object is assigned to only one cluster ("all-or-nothing" approach) while soft

clustering means an object is assigned a strength of association with multiple clusters so it can reside in multiple groups.

An author should not necessarily belong to only one research area. Thus, when using a hard clustering technique, we account for this mixture of areas by using soft probabilistic labels [8]. When using soft clustering techniques, we stick to using hard labels to define each author.

It is important to point out our motivation behind the decision for assigning class labels. Assume there exists a group of authors that publish half in ML and half in AI. We want this distribution to be exposed through our results. If we use a hard clustering technique, then we must use soft labels so that those authors would be in one cluster and labeled as half in ML and half in AI. We describe this metric more in-depth in Section 4. On the other hand, if we use a soft clustering technique with hard labels, those authors would participate half in a ML cluster and half in an AI cluster.

Choosing to do soft clustering and soft labeling loses meaning behind our clusters. Consider an author being labeled as belonging 25 percent in a cluster that is 80 percent ML and 20 percent AI and 75 percent in a cluster that is 40 percent ML and 60 percent AI. This is confusing to interpret, complicates the model, and could easily lead to clusters that do not reveal underlying patterns in the data. Thus, we will use soft labels with hard clusters and hard labels with soft clusters.

To assign labels, assume there are N authors and k clusters. Each author i has a true label vector $\boldsymbol{\ell}^i = [\ell_1^i, \ldots, \ell_9^i]$ where each ℓ_a^i represents the percentage i publishes in area a (since we thresholded the areas, $\ell_a^i \geq 0.2$ for all a). For consistency, the areas are numbered alphabetically as in Table 1 (1 corresponds to AI and 9 corresponds to TH).

For hard clustering, $C(i) \in \{1, \ldots, k\}$ is the cluster assignment for author i, and for cluster j, the soft label vector $\mathbf{w}^j$ will be $\sum_{i:C(i)=j} \boldsymbol{\ell}^i / \text{sum}(\sum_{i:C(i)=j} \boldsymbol{\ell}^i)$. For soft clustering, $C(i) = [\pi_1^i, \ldots, \pi_k^i]$ where $0 \leq \pi_j^i \leq 1$ represents the percentage cluster j contributes to the classification of i. For cluster j, the label vector $\mathbf{w}^j$ will be all 0's except $w_a^j = 1$ for $a \in \{1, \ldots, 9\}$ representing the area corresponding to the maximum element of $\sum_{i=1}^N \pi_j^i \boldsymbol{\ell}^i$.

3.3 Evaluation Metrics

We use four external metrics: mistake rate, accuracy, precision, and recall. For these metrics, we first transform the predicted label $\hat{\boldsymbol{\ell}^i}$ and true author label $\boldsymbol{\ell}^i$. For author i, we create the 9 dimensional vector $\mathbf{y}^i$ where $y_a^i = 1$ if $\ell_a^i > 0$ and $= 0$ otherwise. For the classification, we create $\hat{\mathbf{y}}^i$ where $\hat{y}_a^i = 1$ if $\hat{\ell}_a^i \geq 0.2$ and $= 0$ otherwise. We only consider classification areas contributing 20 percent or more to the label to match the thresholding we determined as described in Section 3.1.

We simplify the error because we care more about a cluster getting a sense of the mixture of research areas an author is in rather than the specific percentages. In the real world when talking about an author, you would not say "she does 33 percent AI and 66 percent ML". Rather, you would say "she does AI and ML". Also, if a technique were to classify an author down to the percentage for some training data, the technique may be overfitting the data and may be too complex.

For research area a, $N_{a=1} = \sum_{i=1}^N y_a^i$ is the number of authors having research area a. We now define the metrics as

$$mistake\ rate = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(\mathbf{y}^i \neq \hat{\mathbf{y}}^i) \qquad prec = \frac{1}{\sum_{a=1}^9 N_{a=1}} \sum_{a=1}^{9} N_{a=1} \frac{\sum_{i=1}^N \mathbb{1}(y_a^i + \hat{y}_a^i = 2)}{\sum_{i=1}^N \mathbb{1}(\hat{y}_a^i = 1)}$$

$$acc = \frac{1}{9N} \sum_{i=1}^{N} \sum_{a=1}^{9} (1 - |y_a^i - \hat{y}_a^i|) \qquad recall = \frac{1}{\sum_{a=1}^9 N_{a=1}} \sum_{a=1}^{9} N_{a=1} \frac{\sum_{i=1}^N \mathbb{1}(y_a^i + \hat{y}_a^i = 2)}{\sum_{i=1}^N \mathbb{1}(y_a^i = 1)}$$

To find $\hat{\mathbf{y}}^i$, we need to classify each author i from $C(i)$. For hard clustering, $\hat{\mathbf{y}}^i = \mathbf{w}^{C(i)}$, and for soft clustering, $\hat{\mathbf{y}}^i = \left(\sum_{j=1}^k \pi_j^i \mathbf{w}^j\right) / \text{sum}\left(\sum_{j=1}^k \pi_j^i \mathbf{w}^j\right)$.

4 Clustering Implementations

4.1 Clique Finding Using Co-Authorships

First, we explore an unweighted and undirected graph based on co-authorships where the nodes represent authors, and two authors have an unweighted edge between them if they co-authored one or more times together.

With this representation, we use the clique-based clustering technique from [1] to extract co-author communities. This technique relies on the idea of k-cliques, which are complete subgraphs of size k. A k-clique-community is a group of k-cliques that are adjacent to each other where two k-cliques are adjacent if they share $k-1$ nodes. The algorithm finds k-clique-communities by first finding maximal complete subgraphs of the network, computing the clique-clique overlap matrix, and using it to locate cliques of all possible k values. Smaller values of k lead to more communities and larger communities that are less cohesive while larger values of k lead to fewer communities that are smaller yet strongly connected. Since authors can be part of zero or more communities, this is a soft clustering technique so we use hard labels. If an author is part of multiple communities, we assign equal weight to each community.

4.2 Random Walk on Co-Authorships

Using the same co-authorship graph representation as before, we utilize a random walk approach to find communities through a modularity metric [18]. Modularity is typically used for finding communities and measures the density of connections between nodes within a group (module). Specifically, we utilize an open source program called MapEquation [2] that finds communities in the following way: first, small communities are found by optimizing modularity locally for each node. Then, nodes in the same community are aggregated together to build a new network whose nodes represent those smaller communities. This process is repeated until a maximum modularity is reached.

Each author is assigned to one community (module) so we use soft labels for this method. Once the modules are found, the largest module has the highest amount of aggregated flow volume for all its nodes.

4.3 K-Means Based on Publications

Besides exploring the communities based on unweighted and undirected graphs, we explore finding areas based only on an author's publications. Specifically, we use k-means to first cluster articles based on similar title keywords and then use those clusters to classify an author. Even though k-means is hard clustering, the author can have articles in multiple clusters, so it acts like a soft clustering on the authors.

Once the articles are clustered into 9 groups, each cluster will consist of publications from potentially different areas but with similar keywords. The label of cluster j then becomes the majority research area of its articles. Before the clustering takes place, we use tf-idf to place less weight on common terms that may exist among the keywords and eliminate common stop words as well as generic computer science terms to reduce the possibility of articles containing these generic terms to be clustered together.

Once clustered, for each author, we find all her publications in each cluster and classify her based on the weighted sum of the cluster labels where the weights are the number of articles she published in each cluster. So, if an author has 2 papers in a HCI cluster and 5 papers in a DB cluster, then the author will get labeled as 29 percent HCI and 71 percent DB.

4.4 K-Means Based on Co-Authorships and Co-Conferenceships

The last technique uses two versions of k-means on an altered co-authorship graph. We did attempt these versions on the original co-authorship graph explained previously, but the results were poor, so we do not discuss them in this paper.

4.4.1 Modified Graph

Clustering just on co-authorships ignores the information about which conferences each author published at. Since we are using unsupervised techniques, we cannot associate conferences with research areas, but we can treat conferences as "black box" publishing venues. We want this information because authors are more likely in similar areas the more they publish at the same conferences (have co-conferenceships) and the more they co-author together. Inspired by [19], we create a graph representing which conferences an author published at and who her co-authors are. She is more related to authors she shares a conference with the more she publishes at that conference. For this representation, we consider an author to be a co-author with herself. We also use a scale factor $f \geq 0$ that weights how heavily co-authorships should be considered in relation to co-conferenceships.

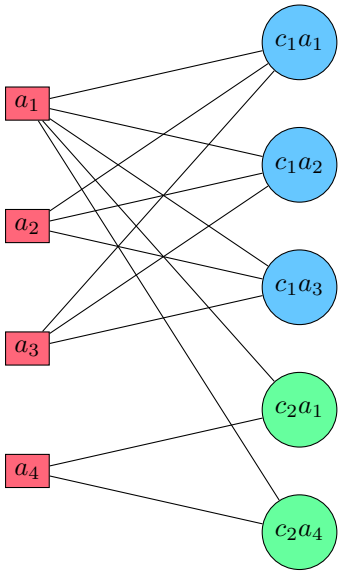
For this graph, assume there are N authors $\{a_1, \ldots, a_N\}$ and M conferences $\{c_1, \ldots, c_M\}$. There is one node for each a_i and one node for each author/conference pair $c_k a_i$ if author i published at conference k. As for weights, n_{ki} is the number of papers i published at k, including those i co-authored in. Also, n_{kij} is the number of papers i co-authored with j at k.

For each conference k author i published at, there is an edge between a_i and $c_k a_j$ for all a_j who published at k. If a_i and a_j did not co-author at k, the edge weight is n_{ki}. If $a_i = a_j$, the edge weight is $f * n_{ki}$, and if $a_i \neq a_j$ and they co-authored together, the edge weight is $n_{ki} + f * n_{kij}$.

For example, take the following conference and co-author data and n values which have been transformed into a graph for conferences c_1 and c_2 and authors a_1 through a_4. In the conference/publications data, for c_k, each sequence of authors followed by a number x in parenthesis represents that those authors, without other co-authors, published x papers together at c_k.

Conference	Publications
c_1	$a_1(2),\ a_1a_2(4),\ a_1a_2a_3(1),\ a_3(3)$
c_2	$a_1a_4(2),\ a_4(3)$

n_{11}	n_{12}	n_{13}	n_{112}	n_{113}	n_{123}	n_{21}	n_{24}	n_{214}
7	5	4	5	1	1	2	5	2



Node	Node	Weight
a_1	c_1a_1	$7f$
a_1	c_1a_2	$7+5f$
a_1	c_1a_3	$7+1f$
a_2	c_1a_1	$5+5f$
a_2	c_1a_2	$5f$
a_2	c_1a_3	$5+1f$
a_3	c_1a_1	$4+1f$
a_3	c_1a_2	$4+1f$
a_3	c_1a_3	$4f$
a_1	c_2a_1	$2f$
a_1	c_2a_4	$2+2f$
a_4	c_2a_1	$5+2f$
a_4	c_2a_4	$5f$

Figure 1: Co-authorship and co-conferenceship graph with edge weights in table to the left

To use this graph in k-means, the feature vector of each author i is the weight of the edges from a_i (square node) to each conference/author node (circle node). The weight is 0 if there is no edge between those two nodes.

4.4.2 Two Versions of K-Means

Since k-means is a hard clustering technique, we need to change the standard algorithm to allow authors to be associated with multiple areas. Our solutions are to separately use standard k-means with soft labels and fuzzy c-means, a soft clustering version of k-means that incorporates a fuzziness factor $m > 1$. As explained in Section 3, authors have a weight vector instead of a class label representing their weight of participation in each cluster. As m approaches 1, the algorithm acts like the standard k-means, and as m approaches infinity, each point becomes equally part of every cluster. In addition, we use cosine distance to represent objects being closer when they share a feature than when they do not because when using euclidean distance, the co-occurrence of a feature is handled similarly to the non-occurrence. (We did try using euclidean distance but got poor results). This means for standard k-means, we follow the update rules explained in [20], and for fuzzy c-means, we follow the ones given in [17].

5 Results

5.1 Clique Finding Using Co-Authorships

We use code provided by the authors of [9] found at [1] to run the k-clique-community finder. Due to the code's input size limitations, we restricted our co-author graph to include authors from 5,000 random publications from each area to still represent the natural distribution of authors per area. With that restriction, we had 65,993 authors.

The k values found ranged from 3 to 33 where there were 13,358 and 1 k-clique-communities found respectively. Although this visual community is helpful, to formally evaluate the algorithm, we chose k values based on the number of communities formed. Since our goal is to have one community per research area, we chose $k = 18$ and $k = 17$ because there were 8 and 10 communities formed respectively. Unfortunately, the majority of authors for these k values were not placed into communities, giving us an overall mistake rate of 0.99. If we just evaluate the authors and areas that were selected, for $k = 18$, with 178 total authors, we had a mistake rate of 0.05, accuracy of 99%, precision of 99%, and recall of 96%, and for $k = 17$, with 212 total authors, we had a mistake rate of 0.04, accuracy of 99%, precision of 99%, and recall of 95%. Note that for $k = 18$, no author from DB, HCI, and TH appeared in any cluster, and for $k = 17$, no author from HCI and TH appeared in any cluster.

Overall, this method performed extremely poorly and was unable to group authors into the 9 research areas. However, the communities found were strongly related to certain research areas. We believe this model performed poorly because it was based on a very simplistic representation of unweighted co-authorships, which is not complicated enough to group authors by research area given that computer science fields are interrelated and connected.

5.2 Random Walk on Co-Authorships

Similarly to the clique finding approach, the MapEquation method was helpful visually but found too many communities that were too small and not informative when using an unprocessed list of unweighted and undirected edges. These smaller communities consisted of author pairs that were linked together, but were not linked to larger modules. To solve this, we iterated through the authors and only kept those that were at least 3 degrees of separation away from another author. Specifically, we selected a set of 7,500 random publications from each area and then eliminated authors in small groups, giving us a total of 81,027 unique authors.

The MapEquation community finding application did not deliver optimal results for finding all the existing research areas from the author/co-author edges we provided. It found a total of 3,456 modules, and by examining the larger modules, their flow, and their labels, we learned which communities were highly connected. For example, we learned there exists a strong connection between TH and NC (networks) as well as TH and DB, which makes sense given the interdisciplinary nature of TH.

This method resulted in a total mistake rate of .43, accuracy of 92%, precision of 62%, and recall of 100%. We believe the recall is high since there existed thousands of modules, making each of

the 9 areas represented multiple times. In a sense, we overfitted each area since the small groups of authors are likely from the same area.

5.3 K-Means Based on Publications

We use WEKA [10] in order to cluster each of the publications based on title keywords. WEKA vectorizes the keywords into a binary vectors of words and removes stopwords and general computer science terms. Due to WEKA's limited speed when dealing with large data, we only used a random set of 1000 publications from the top conference in each area (similar to k-clique finding), giving us a total of 20,192 authors. In order to reduce the number of words in the feature vectors, we only use words that occur in more than one publication. This reduces the total number of features from 11,102 to 316. With the additional removal of stopwords and computer science terms, we are left with 307 features. Due to the sensitivity of initialization of the k-means algorithm, WEKA ran 500 iterations of the algorithm to find the optimal clusters.

This method had a mistake rate of 0.84, an accuracy of 80%, precision of 45% and recall of 20%. This method was not able to work well because many of the clusters contained a variety of areas, and was unable to effectively differentiate between the distinct research areas. We realized that many computer science terms are too general to help distinguish between our selected research areas.

5.4 K-Means Based on Co-Conferenceships

Since this representation resulted in a very large and sparse input matrix, to maintain a reasonable runtime, we selected authors from 500 random publications from each area, giving us a total of 10,806 authors. We also chose a weight factor of $f = 2$ to keep co-conferenceships important.

5.4.1 Hard Clusters, Soft Labels

We do not know what our ideal k value would be since clusters must be able to represent individual research areas as well as groups of research areas. However, larger k values will likely result in better accuracy and mistake rate due to overfitting, so we use internal metrics to chose an optimal k by calculating inter-cluster distance for various values of k [16], [12]. We could also calculate intra-cluster distance, but since that distance decreases as k increases towards N, it is not indicative of the best k value. Therefore, we just use inter-cluster distance where the average inter-cluster distance is the average cosine distance between each pair of cluster centers (shown in Figure 2).

Since we want to maximize inter-cluster distance, the best k value is 10, which gives us an accuracy of 90%, precision of 72%, recall of 86%, and mistake rate of 0.56. We further observed interesting trends in the groupings of research areas that appeared as labels such as ML and DB or ML and GV. These make intuitive sense when thinking about the areas and how there is a lot of overlap between those fields.

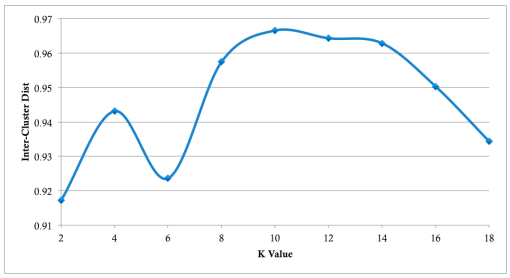


Figure 2: Standard k-means with soft labels for various values of k

5.4.2 Soft Clusters, Hard Labels

Our goal is to produce one cluster per research area, so our initial evaluation used $k = 9$. Figure 3 is a graph of mistake rate, accuracy, precision, and recall as a fraction from 0 to 1 for various values of m averaged over 5 trials. The mistake rate is lowest, accuracy and recall highest, and precision still around 90% when $m = 1.1$, indicating a lower value of m seems to perform better overall. This is further supported by the fact that when $m = 1.1$, some trials had each cluster represent one research area while that occurred less often when $m > 1.1$. This makes sense when considering that the majority of authors are in only one research area and would be misclassified if placed in multiple clusters. Although not shown, as m gets larger, more research areas are not represented and the cluster labels converge to the majority research area, AI.

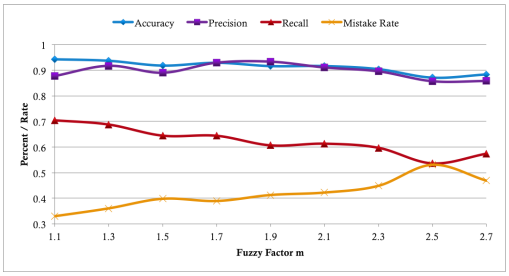


Figure 3: Fuzzy c-means evaluation metrics for $k = 9$

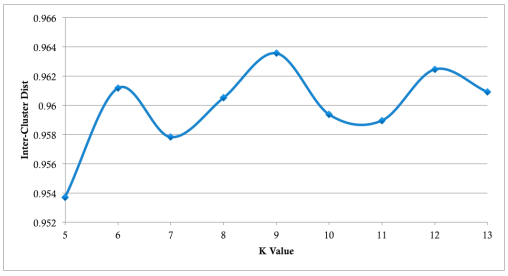


Figure 4: Inter-cluster distances plotted for $m = 1.1$ for various k values

We further evaluate the clusters by measuring the intra-cluster distance for various values of m where the average intra-cluster distance is

$$\frac{1}{k}\sum_{j=1}^{k}\left(\frac{\sum_{i=1}^{N} w_j^i \text{dist}_{\cos}(\mathbf{x}^i, \mathbf{c}^j)}{\sum_{i=1}^{N} w_j^i}\right).$$

As we saw with our external metrics, $m = 1.1$ performed best, having the lowest intra-cluster distance (around 0.3) for any $6 \leq k \leq 12$. Due to space limitations, we do not show the graph of these results. Lastly, fixing $m = 1.1$, Figure 4 shows the inter-cluster distances for various values of k averaged over 5 trials. The distance peaks at $k = 9$ which matches what we know to be the true number of areas.

6 Conclusion

Through this work, we used hard and soft clustering techniques to identify authors' research areas. Our best results came from fuzzy c-means where we represented not only co-authorships but also co-conferenceships. Some of the challenges included acquiring a data sample that accurately reflected the true distribution of authors from different areas, developing and modifying evaluation metrics to fit with multi-class labeling, and determining which author properties are the best features.

From our results, we noticed certain trends between research areas. For example, though the standard k-means and Infomap techniques, we learned that there exists a strong connection between TH/NC and ML/AI, implying there is a noticeable amount of collaboration between these areas. This points towards future work exploring hierarchical clustering, specifically a divisive (top-down) clustering, where we could learn about existing sub-groups within a cluster. For example, if we specifically hard labeled a cluster as AI, this cluster may have sub-groups of authors from DB or ML, indicating collaborations between these groups.

For other future work, we could experiment to better understand the relationship between the fuzzy factor m with respect to the number of clusters k. Additionally, we could add different information to an author's feature vector such as the keywords of her publication titles and information about the years she attended each conference and the years she co-authored with someone. From this temporal data, we could learn about the flow of authors migrating to different areas throughout the years.

Lastly, we want to point out how the techniques shown in this paper can be used. Two direct use cases would be developing a co-author recommendation system which uses clustering to find authors similar to you and developing a research area predictor for unlabelled authors. More abstractly, the co-authorship/co-conferenceship representation could be adapted to problems like finding subcategories of food ingredients or subcategories of newspaper ads. Ingredients occurring in the same meal is like a co-conferenceship while ingredients occurring in the same recipe is like a co-authorship. Similarly, newspaper ads occurring in the same paper is a co-conferenceship while ads occurring on the same page is a co-authorship because ads on the same page are working together to entice the reader. As shown through these examples, these techniques can be extended to real world problems.

References

[1] Clusters and communities. Avalaible at www.cfinder.org.

[2] Mapequation. Available at http://www.mapequation.org/.

[3] Microsoft academic search. Available at academic.research.microsoft.com.

[4] A. Babskova, P. Drazdilova, J. Martinovic, V. Svaton, and V. Snasel. Evolution of co-authors communities formed by terms on dblp. 2013.

[5] S. Bhowmik. Computer science conference rankings. Available at http://www.ntu.edu.sg/home/assourav/crank.htm.

[6] M. Biryukov and C. Dong. Analysis of computer science communities based on dblp. In *Proceedings of the 14th European conference on Research and advanced technology for digital libraries*, ECDL'10, Berlin, Heidelberg, 2010. Springer-Verlag.

[7] J. Gao, W. Fan, Y. Sun, and J. Han. Heterogeneous source consensus learning via decision propagation and negotiation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, New York, NY, USA, 2009. ACM.

[8] N. Gayar, F. Schwenker, and G. Palm. A study of the robustness of knn classifiers trained using soft labels. In F. Schwenker and S. Marinai, editors, *Artificial Neural Networks in Pattern Recognition*, volume 4087 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006.

[9] I. F. T. V. Gergely Palla, Imre Dernyi. Uncovering the overlapping community structure of complex networks in nature and society, 2005.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1).

[11] Z. Huang, Y. Yan, Y. Qiu, and S. Qiao. Exploring emergent semantic communities from dblp bibliography database. In *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining*, ASONAM '09, Washington, DC, USA, 2009. IEEE Computer Society.

[12] H. Khatri. Project c: Clustering report, 2004. Available at http://rakaposhi.eas.asu.edu/cse494/f05-projects/ProjC_Hemal.pdf.

[13] E.-K. Lee. Computer science conference rankings. Available at http://www.cs.ucla.edu/~eklee/paper/CS_conf_rank.htm.

[14] M. Ley. Dblp: some lessons learned. *Proc. VLDB Endow.*, 2(2), 2009.

[15] W. Lin, X. Kong, P. S. Yu, Q. Wu, Y. Jia, and C. Li. Community detection in incomplete information networks. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, New York, NY, USA, 2012. ACM.

[16] B. Liu. Unsupervised learning. Available at www.cs.uic.edu/~liub/teach/cs583.../CS583-unsupervised-learning.ppt.

[17] M. E. S. Mendes and L. Sacks. Evaluating fuzzy clustering for relevance-based information access. In *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on*, volume 1, pages 648–653, 2003.

[18] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci. USA*, page 1118, 2008.

[19] O. R. Zaïane, J. Chen, and R. Goebel. Dbconnect: Mining research community on dblp data, 2007.

[20] S. Zhong. Efficient online spherical k-means clustering. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5. IEEE.

7 Appendix - Extra Images

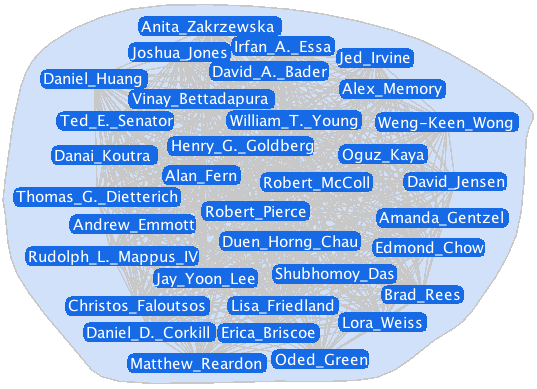


Figure 5: Sample k-clique finding ML cluster for k=21 consisting of 34 authors

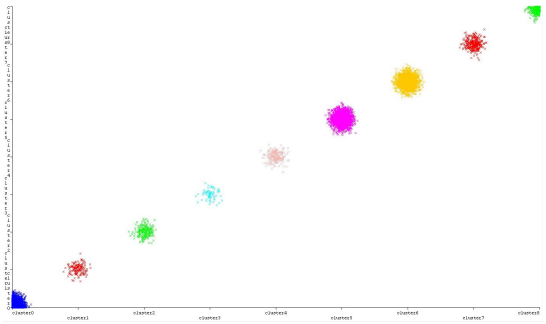


Figure 6: Publication Clusters found in WEKA when k = 9

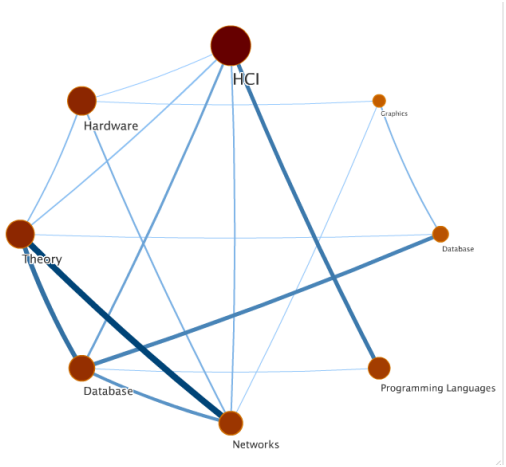


Figure 7: Sample of largest modules found in InfoMap